The Parameterized Complexity of Degree Constrained Editing Problems

MATHIESON, LUKE

Abstract

Luke Mathieson – The Parameterized Complexity of Degree Constrained Editing Problems

This thesis examines degree constrained editing problems within the framework of parameterized complexity. A degree constrained editing problem takes as input a graph and a set of constraints and asks whether the graph can be altered in at most k editing steps such that the degrees of the remaining vertices are within the given constraints. Parameterized complexity gives a framework for examining problems that are traditionally considered intractable and developing efficient exact algorithms for them, or showing that it is unlikely that they have such algorithms, by introducing an additional component to the input, the parameter, which gives additional information about the structure of the problem. If the problem has an algorithm that is exponential in the parameter, but polynomial, with constant degree, in the size of the input, then it is considered to be fixed-parameter tractable. Parameterized complexity also provides an intractability framework for identifying problems that are likely to not have such an algorithm.

Degree constrained editing problems provide natural parameterizations in terms of the total cost k of vertex deletions, edge deletions and edge additions allowed, and the upper bound r on the degree of the vertices remaining after editing. We define a class of degree constrained editing problems, WDCE, which generalises several well know problems, such as Degree r Deletion, Cubic Subgraph, r-Regular Subgraph, f-Factor and General Factor. We show that in general if both k and r are part of the parameter, problems in the WDCE class are fixed-parameter tractable, and if parameterized by k or r alone, the problems are intractable in a parameterized sense.

We further show cases of WDCE that have polynomial time kernelizations, and in particular when all the degree constraints are a single number and the editing operations include vertex deletion and edge deletion we show that there is a kernel with at most $O(kr(k+r))$ vertices. If we allow vertex deletion and edge addition, we show that despite remaining fixed-parameter tractable when parameterized by k and r together, the problems are unlikely to have polynomial sized kernelizations, or polynomial time kernelizations of a certain form, under certain complexity theoretic assumptions.

We also examine a more general case where given an input graph the question is whether with at most k deletions the graph can be made r-degenerate. We show that in this case the problems are intractable, even when r is a constant.

The Parameterized Complexity of Degree Constrained Editing Problems

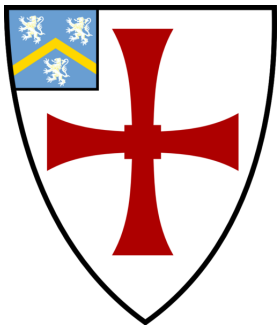
Luke Mathieson

Supervisor:

Dr. Stefan Szeider

Co-supervisor:

Prof. Hajo Broersma



School of Engineering and Computing Sciences

University of Durham

Thesis Submitted in Fulfilment of the Requirements of the Degree of Doctor of Philosophy

2009

Contents

List of Tables

List of Figures

Chapter 1

Preamble

1.1 Declaration

The work contained in this thesis represents original work by the author under the supervision of Dr. Stefan Szeider and Prof. Hajo Broersma. Portions of this thesis have been published in preliminary form [117, 118]. At the time of writing a two further papers [119, 115] have been submitted, but not yet reviewed.

No part of this thesis has been previously submitted for any degree at any institution.

1.2 Statement of Copyright

1.3 Acknowledgements

I would like to thank, in roughly chronological order, my parents for encouraging me in academic pursuits from an early age, Dr. Ljiljana Branković for first introducing me to computational complexity and theoretical computer science in general, Prof. Mike Fellows and Dr. Fran Rosamond, not just for introducing me to parameterized complexity and finding me a PhD fellowship, but for showing me the fun in complexity theory, and of course my supervisor, Dr. Stefan Szeider for his patience, guidance and support through this PhD.

I would also like to thank, in no particular order and for various reasons, Dr. Pablo Moscato, Dr. Regina Berretta, Dr. Michael Hannaford, Dr. Alexandre Mendes, Dr. Mario Inostroza, Triinu Tasa, Dr. Elena Prieto, Matt Skerritt, Andrew Hide, Tony Kemp, Bryan Paton, Dr. Liz Paton, Pim van 't Hof, James Gate, Dr. Barnaby Martin, Ioannis Lignos, Mark Rhodes, Dr. Berndt Farwer and no doubt many others.

I would further like to thank the University of Durham for both its support and environment during my doctoral studies, and my examiners Dr. Vadim Lozin and Prof. Iain Stewart.

Most of all I would like to thank Amanda.

1.4 Dedication

This thesis, and much more, is dedicated to Amanda Grech.

Chapter 2

Introduction

Graphs are key tools for the computational modelling of many real world problems. Consisting of elements (vertices) and relationships (edges) between pairs of elements (see Section 2.3 for formal definitions), graphs can be used to model problems and structures in networking, resource allocation, genetics, physics, silicon chip design, and many other areas. Furthermore graphs, as discrete structures, are easily handled by computers, and are thus an attractive abstraction tool in computer science.

Graph editing and modification plays a major role in algorithmic graph theory, and many problems can be viewed as an editing problem of some form. For example the well known Vertex Cover [80] problem, that asks for a (minimum) set of vertices of the graph where every edge is incident on at least one of these vertices, can be viewed as a vertex deletion problem where we are looking for a (minimum) set of vertices that can be deleted so that every vertex remaining in the graph is isolated. Similarly Dominating Set [80] can be viewed as an editing problem where we want to delete a set of vertices so that the degree of all the remaining vertices is lowered by at least one.

In fact any subgraph problem where we are given a graph G and asked to find a subgraph H with given structure or properties can be viewed as an editing problem. In this sense Clique, Subgraph Isomorphism, Maximum Cut, Maximum Leaf Spanning Tree, r-Regular Subgraph (all in [80]), General Factor [106] and many others can be viewed as editing problems.

Of course there are also many graph problems that are explicitly stated as editing problems. Cluster Editing [83], which asks for a minimum number of edge deletions and additions to create a set of disjoint cliques, is a particularly prominent

graph editing problem, thanks to its broad range of applications [15, 105, 110, 158].

Editing problems are also widely studied in the setting of parameterized complexity. Parameterized complexity extends the notion of tractability by including in the problem a special input, the parameter. If a problem can be solved in time bounded by a polynomial in the overall size of the input, but possibly exponential in the parameter, then it is tractable in a parameterized sense (see Chapter 3 for formal definitions). Many editing problems can be naturally parameterized by the number of editing steps allowed, for example Graph Bipartization [90, 142] and Cluster Editing [83] are both fixed-parameter tractable with the number of editing steps as the parameter.

The character of editing problems can also be varied by the editing operations available (see Section 2.3.1). If vertex deletion alone is allowed, then the graphs produced by editing are induced subgraphs, edge deletion alone produces spanning subgraphs. If edge addition is allowed then the result may no longer be a subgraph of the original graph. If vertex deletion, edge deletion and edge contraction are allowed, then we get the Minor Order Testing problem [144] (see Section 3.5.1).

In this thesis we study parameterizations of *degree constrained* editing problems, where rather than editing to produce a specific target graph, the goal of editing is to be left with a graph where the degrees of the vertices are within certain constraints, such as the Cubic Subgraph problem [80].

2.1 Known Results on the Complexity of Graph Modification and Editing Problems

The general Π Vertex (Edge) Deletion problem asks for a minimal set of vertices (edges) whose deletion results in a graph with property Π. Krishnamoorthy and Deo [97] classify the Π Vertex Deletion problem where Π defines one of 17 different graph classes including complete graphs, acyclic graphs and degree constrained graphs, showing that in all cases Π Vertex Deletion is NP-complete. Yannakakis [168] shows that if Π is hereditary on induced subgraphs, Π Vertex Deletion is NP-complete, and requiring that the graph remaining after deletion is connected does not alter the complexity. Yannakakis [169] also shows that the Π Edge Deletion problem is NP-complete for certain properties. Lewis [100] gives NP-completeness results for the Π Vertex Deletion problem with various properties Π (Lewis and Yannakakis give a combined presentation of results in [101]).

Yannakakis [170] also gives various cases where restricting the input graph to a bipartite graph results in polynomial time algorithms for hereditary properties.

Watanabe *at al.* [165, 166] give a more generalised result for Π Edge Deletion, showing that the problem is NP-complete if Π can be characterised by finitely many 3-connected graphs, their result also holds for the Π Edge Contraction problem. Asano and Hirata [11] generalise this result to hereditary properties determined by the 3-connected components. Later [12] they refine the result for the Π Edge Contraction problem, showing that it is NP-complete if Π is hereditary under edge contractions and determined by the biconnected components. They also show that if Π is determined by the 3-connected components, then the Π Edge Contraction remains hard when restricted to 3-connected graphs. Colbourn and El Mallah [40] consider further restrictions on the class of target graphs, and show that the H Edge Deletion problem where H is the class of target graphs is NP-complete when H is defined by a set of forbidden minors which are all 2-connected with minimum degree 3, when H is defined by K_4 as a forbidden minor and when H is defined by P_3 as a forbidden induced subgraph.

When edge deletion and edge addition are allowed we get the Π Edge Modification problem. Natanzon *et al.* [123] show that this is also NP-complete for certain target graph classes such as perfect graphs, chordal graphs and split graphs. They also give cases where bounded degree input graphs admit polynomial time algorithms.

Schoone *et al.* [146] show that it is NP-complete to decide whether it is possible to add at most k edges to reduce the diameter of the graph below a bound D, or to delete at most k edges to increase it above a bound D where k and D are part of the input.

As mentioned previously, Cluster Editing and its variants are well studied problems due to their wide practical applicability. Ben-Dor *et al.* [14] introduce the problem and demonstrate a stochastic algorithm. Shamir *et al.* [149] prove that Cluster Editing (specifically where both edge deletion and edge addition is allowed) is NP-complete. If we consider only clusters on two vertices (i.e. K_2) then Maximum Matching can be thought of as a form of Cluster Deletion problem. Hell and Kirkpatrick [89] take this view and generalise the problem to deletion with the aim of obtaining a collection of vertex disjoint subgraphs isomorphic to a given graph on at least 3 vertices. They show that despite Maximum Matching admitting a polynomial time algorithm their generalised problem is NP-complete.

2.1.1 Degree Constraint Problems

Degree constrained editing problems have a long history in the literature. Cubic Subgraph is one of the early NP-complete problems, with a proof attributed to Chvátal by Garey and Johnson [80]. If one considers matching problems as degree contrained editing problems, i.e. where we ask for a 1-regular spanning subgraph, then the first polynomial time algorithms date to the work of Kuhn [98, 99] and Edmonds [64, 65]. Matchings in their own right form a wide topic of research [81, 109].

The Cubic Subgraph problem naturally generalises to the r-Regular Subgraph problem, which can be seen as a degree constrained editing problem where we are allowed to delete edges and vertices to obtain an r regular graph. Plesník [134] establishes the NP-completeness of r-Regular Subgraph for every $r \geq 3$, even when the input graph is bipartite, planar and has maximum degree 4. Stewart [151, 152, 153] shows NP-completeness holds under a series of other constraints. Cheah and Corneil [33] show that r-Regular Subgraph remains NP-complete when the input graph has maximum degree $r + 1$. If we only allow vertex deletion (i.e. the subgraph must be induced), but demand that we delete a minimal number of vertices, the problem corresponds to the Maximum r-Regular Induced Subgraph problem. Cardoso *et al.* [30] show that this problem is NP-complete for every $r \geq 0$ (notably when $r = 0$ the problem is Maximum Independent Set). If the only editing operation allowed is edge deletion (i.e. the subgraph is spanning), then we have the r-Factor problem, which Tutte [160] reduces to a matching problem, later shown to be polynomial time solvable. In fact Tutte's proof is for the more general f-Factor problem [161], where each vertex may have a different degree, specified by the function f. Anstee [10] provides a direct algorithm and Liu and Zhu [104] give an improved algorithm when the input graph is bipartite. The Degree Constrained Subgraph problem generalises the degree constraints with each vertex having upper and lower bounds for the final degree. Urquhart [163] gives the earliest polynomial solution specifically for this problem, although an algorithm can be derived from Edmond and Johnson's work [66] according to Shiloach [150], who also provides alternative algorithms for the problem. The (Perfect) b-Matching problem [81] extends this further, by adding capacities to the edges. Tutte's [160, 161] algorithm extends to cover this problem [94]. Tutte [162] gives a discussion of related results.

Lovász [106, 107] introduces and studies the General Factor problem, which

generalises the f-FACTOR problem by giving each vertex a list of possible degrees and asks for a subset of the edges of the graph that satisfy these constraints. Similarly the GENERAL ANTIFACTOR problem [108] gives each vertex a list of forbidden degrees. Cornuéjols [43] gives a complete complexity classification of the GENERAL FACTOR problem; if the degree lists (excepting some trivial cases) contain gaps of length greater than 1, e.g. $\{2, 5, 6, 8\}$ the problem is NP-complete, otherwise the problem is polynomially solvable using matching techniques.

Lin and Sahni [103] look at a global degree constraint, where the question is whether at most k edge deletions can leave a graph where the number of edges incident on any vertex and the number of edges deleted from any vertex is bounded. In particular they show that if the resultant graph is acyclic then the problem is NP-hard if the input graph is undirected, but can be solved in linear time if the input graph is directed.

Bodlaender *et al.* [21] examine the problem of finding a Δ-regular supergraph of the input graph using edge addition and a minimum number of vertex additions (at most 2), where Δ is the maximum degree of the input graph. They show that this problem is solvable in polynomial time.

2.1.2 Parameterized Graph Editing

Parameterized complexity (see Chapter 3) examines the complexity of problems, typically NP-hard, taking into account the structure within the problem and the input. This structure is measured by an additional input, the parameter. The parameter can measure many aspects of a problem; natural parameters include solution size, structural measures of the input, and for editing problems in particular, the number of editing steps allowed. When the number of editing steps is taken as or included in the parameter, many editing problems can be shown to be fixed-parameter tractable (see Section 3.1). For the remainder of this section, k will be the number of editing steps allowed and G the input graph, unless otherwise specified.

One of the most general graph editing results, due to Cai [27], concerns editing a graph to obtain a graph with hereditary graph property Π. Given a limited number of edge additions, edge deletions and vertex deletions, if Π can be characterised by a finite set of forbidden induced subgraphs then the problem is fixed-parameter tractable. By controlling the number of editing operations of each type allowed, this result establishes the fixed-parameter tractability of several distinct subproblems

such as the Π Edge Deletion problem and the Π Edge Editing problem. This, for example, shows that Cluster Editing is fixed-parameter tractable with an $O(3^k \cdot |V(G)|^4)$ time algorithm where the forbidden graph is a path on three vertices. Conversely a property such as regularity is not hereditary, so Cai's result does not apply. Kratsch and Wahlström [96] answer an open question of Cai and show that Π Edge Deletion and Π Edge Editing do not in general have polynomial size kernelizations. Khot and Raman [93] examine the related problem of finding a Π subgraph with k vertices where Π is a hereditary property. They show that if Π contains all trivial graphs, but not all complete graphs, or vice versa, then the problem is $W[1]$-hard. Otherwise the problem is fixed-parameter tractable.

Cluster Editing problems are some of the most widely studied editing problems in parameterized complexity with not only extensive theoretical results, but numerous practical studies [17, 19, 51, 84, 138]. Gramm *et al.* [83] introduce the problems to the parameterized context giving a $O(2.27^k + |V(G)|^3)$ time algorithm for Cluster Editing, where k is the number of edge deletions and edge additions allowed, and a $O(1.77^k + |V(G)|^3)$ time algorithm for Cluster Deletion, where only edge deletion is allowed. Both cases have a kernel, an equivalent instance of size bounded by a function of the parameter (see Section 3.5.7), with $O(k^3)$ vertices. As they note, Cluster Completion, where only edge addition is allowed, is polynomial time solvable. Dehne *et al.* [51] investigate implementations based on Gramm *et al.*'s work. Protti *et al.* [137] improve the kernelization to $O(k^2)$ vertices and give an $O(4^k + |V(G)| + |E(G)|)$ time algorithm. Fellows *et al.* [69, 74] improve the kernel size to $24k$ vertices, and Guo [85] improves this further to $4k$. Böcker *et al.* [18] give an $O(1.82^k + |V(G)|^3)$ time algorithm which Hüffner *et al.* [91] claim can be combined with Protti *et al.*'s work to give a $O(1.82^k + |V(G)| + |E(G)|)$ time algorithm. Damaschke [46] shows that enumerating all minimal sets of edges to delete or add is also fixed-parameter tractable. Guo [85] also shows fixed-parameter tractability with a kernel with at most $(d+2)k + d$ vertices for the Cluster Editing[d] problem, which asks for at most k edge edits that give exactly d cliques. Hüffner *et al.* [91] examine the parameterized complexity of Cluster Vertex Deletion, where at most k vertex deletions are allowed, and no edge deletions or additions, and the d-Cluster Vertex Deletion problem, the vertex deletion analogue of Cluster Editing[d]. They show both are fixed parameter tractable, and in fact give three different algorithms for d-Cluster Vertex Deletion. Damaschke [47] introduces a generalisation of the Cluster Editing problem, the Twin Graph

EDITING problem in which we are asked to add or delete at most k edges to obtain a twin graph with at most t edges, where a twin graph is the graph of relationships between vertex equivalence classes. CLUSTER EDITING is the special case where $t = 0$. Damaschke shows that this problem is fixed-parameter tractable when parameterized by k and t together. If the problem is changed to ask for k changes that give a twin graph with at most t vertices, the new problem is still fixed-parameter tractable when parameterized by k and t together.

Marx and Schlotter [113] examine the PLANAR $+k$ VERTEX problem, which asks for a set of vertices whose deletion renders the graph planar. As this class of "almost planar" graphs is minor closed, Robertson and Seymour's graph minor theorem (see Section 3.5.1) immediately gives fixed-parameter tractability for the PLANAR $+k$ VERTEX problem. However this approach gives only a classification result. Marx and Schlotter give a more practical algorithm relying on reducing the input graph then applying treewidth based techniques (see Section 3.5.1). We show in Section 6.3 that the related general problem for degenerate graphs is $W[P]$-hard.

Given two graphs G and H the INDUCED SUBGRAPH ISOMORPHISM problem asks whether there is an induced subgraph of G isomorphic to H. Typically this problem is parameterized by $|V(H)|$. Marx and Schlotter [114] recast the problem as an editing problem with the parameterization $|V(G)| - |V(H)|$. With this parameterization the problem is fixed-parameter tractable when H is a tree or when both H and G are planar and H is 3-connected.

Nishimura *et al.* [130] study the following generalisation of VERTEX COVER: let $\mathcal{G}$ be a graph class that is minor closed, where each graph in $\mathcal{G}$ has bounded degree and all obstructions are connected; then given a graph G, is it possible to remove at most k vertices such that the resulting graph is in $\mathcal{G}$? They show that this general problem has an $O((g+k)|V(G)| + (fk)^k)$ time algorithm, where f and g are constants dependent on $\mathcal{G}$. If $\mathcal{G}$ is the class of graphs of maximum degree d, which is not minor closed, they show that they can still obtain an algorithm with time complexity $O((d+k)|V(G)| + k(d+k)^{k+3})$. Fellows *et al.* [70] concentrate on the BOUNDED DEGREE DELETION case where $\mathcal{G}$ is the class of graphs of maximum degree d, and give a kernel with at most $(d^3+4d^2+6d+4)k$ vertices, implying a single exponential algorithm for the problem. Interestingly they use a generalisation of Nemhauser and Trotter's [124] theorem that implies a $2k$ kernelization for VERTEX COVER.

The problem of adding at most k edges to a graph to obtain an interval graph,

k-Interval Completion, appears in [80] as an NP-complete problem. Heggernes *et al.* [88] show that it is fixed-parameter tractable with a bounded search tree algorithm running in $O(k^{2k} \cdot |V(G)|^3 \cdot |E(G)|)$ time.

Marx [112] shows that the Chordal Deletion problem of deleting at most k vertices and at most k' edges to obtain a chordal graph is fixed-parameter tractable with parameter $k+k'$ using iterative compression (see Section 3.5.4). Setting either k or k' to zero also shows that the Chordal Vertex Deletion and Chordal Edge Deletion problems are fixed-parameter tractable.

Moser and Thilikos [122] provide an examination of some parameterized editing problems that are explicitly degree constrained with the k-Size r-Regular Induced Subgraph and k-Almost r-Regular Induced Subgraph problems. k-Size r-Regular Induced Subgraph asks for an r-regular induced subgraph of the input graph with exactly k vertices. This problem is $W[1]$-hard for any $r \geq 0$ when parameterized by k. Conversely k-Almost r-Regular Induced Subgraph asks for an r-regular induced subgraph of the input graph which is obtainable by at most k vertex deletions. When parameterized by both k and r, this problem is fixed-parameter tractable. Stewart [154] shows how the fixed-parameter tractability of k-Almost r-Regular Induced Subgraph can be established using a logic based approach. In the conference version of the paper [121] Moser and Thilikos left the parameterized complexity of k-Almost r-Regular Induced Subgraph when parameterized by k alone as an open question. In Chapter 4 we answer this question, showing that it is $W[1]$-hard, along with several other variants and generalisations.

Amini *et al.* [9] show that finding a subgraph with k vertices and minimum degree d is $W[1]$-hard when parameterized by k; however the problem becomes fixed-parameter tractable on classes of graphs defined by excluded minors and classes of graphs of effectively bounded local treewidth.

2.2 Where to Look: A Guide for this Thesis

The remainder of the Introduction is devoted to providing preliminary definitions needed for the main body of the work (Section 2.3).

Chapter 3 formally introduces parameterized complexity theory, including an overview of the major techniques and tools of the field.

Chapter 4 introduces the WDCE* class of problems, which generalise a variety of degree constraint problems, particularly the r-Regular Subgraph problem,

and classifies several WDCE* problems according to their parameterizations and the available editing operations.

Chapter 5 extends the definition of WDCE* to the WDCE class of problems, which generalise the General Factor problem [106]. Chapter 5 completes the classification begun in Chapter 4 and extends these results to the more problematic case where both vertex deletion and edge addition are allowed as editing operations (see Section 2.3.1). We also examine various WDCE problems where the parameterization includes the treewidth of the input graph.

In Chapter 6 we move to a different style of degree constraint and examine parameterized problems in degenerate graphs (see Section 2.3.1). We give several complexity results for well known problems restricted to degenerate graphs, including both $W[1]$-hardness and fixed-parameter tractability results (defined in Sections 3.2 and 3.1 respectively). We conclude this with a proof that unlike regular graphs and general factors, it is hard to obtain degenerate graphs with a limited number of editing steps, even when the degeneracy is a constant.

2.3 Preliminaries

2.3.1 Graph Theory and Notation

We now present relevant graph theoretic definitions, for complete coverage any of the monographs of Diestel [53], Bondy and Murty [25], Chartrand and Lesniak [32] or West [167] are excellent.

A *graph* $G = (V, E)$ is a pair of finite sets V and E where E is a set of two element subsets of V. V is the set of *vertices* of G. E is the set of *edges* of G. Where the vertex and edge sets are not explicitly labelled, $V(G)$ denotes the vertex set and $E(G)$ the edge set. Unless otherwise stated we will only consider graphs where $E(G)$ is a set, that is there are no *parallel* edges. In general however $E(G)$ may be a multiset.

For two vertices $u, v \in V(G)$ we denote the edge $\{u, v\}$ as uv or equivalently vu. In general we will only consider graphs where given an edge $uv \in E(G)$, $u \neq v$.

Given two graphs $G = (V, E)$ and $G' = (V', E')$, if $V' \subseteq V$ and $E' \subseteq E$, we say G' is a *subgraph* of G. If $V' \subsetneq V$ or $E' \subsetneq E$, then G' is a *proper* subgraph of G. If G' is a subgraph of G and for every pair of vertices $u, v \in V(G')$ we have $uv \in E(G')$ if and only if $uv \in E(G)$, then G' is an *induced* subgraph of G. If G' is a subgraph of G and $V(G') = V(G)$, then G' is a *spanning* subgraph of G.

Given a graph G and a vertex v we denote the graph $G' = (V(G) \setminus \{v\}, E(G) \setminus \{uv | u \in V(G)\})$ by $G - v$. We extend this notation in the natural way to cover edges, and sets of vertices and/or edges.

Given two vertices $u, v \in V(G)$ and an edge $uv \in E(G)$ we say u is *adjacent* to v (or u and v are adjacent) and u and v are *incident* on uv. u and v are the *endpoints* of uv. If two vertices are adjacent, then they are *neighbours*. Given a vertex u the set of all neighbours of u is denoted $N_G(u)$ and is called the *open* neighbourhood of u, and $N_G[u] = N_G(u) \cup \{u\}$ is called the *closed* neighbourhood of u. Given a set V' of vertices, the open neighbourhood of V' is $N_G(V') = \bigcup_{u \in V'} N_G(u) \setminus V'$, and the closed neighbourhood of V' is $N_G[V'] = \bigcup_{u \in V'} N_G[u]$. If a vertex v has $|N_G(v)| = 0$, we call v *isolated*. Where context allows we omit the subscript and write $N(u)$, $N[u]$, $N(V')$ and $N[V']$.

A subgraph G' of a graph G is a *clique* if the vertices in $V(G')$ are pairwise adjacent, i.e. $uv \in E(G)$ for all $u, v \in V(G')$. A clique on k vertices is called a *k-clique* or a *complete graph* on k vertices, and is denoted K_k. A set V' of vertices is *independent* (or an *independent set*) if the vertices are pairwise non-adjacent, i.e. $uv \notin E(G)$ for all $u, v \in V'$.

We denote the set of edges that v is incident on as $E(v)$, i.e. $E(v) = \{\, uv \in E(G) \mid u \in V(G) \,\}$. The *degree* of a vertex v, denoted $d(v)$ is $|E(v)| = |N(v)|$. Given a set V' of vertices, the *degree of v into V'*, denoted $d_{V'}(v)$, is the number of vertices $u \in V'$ where $u \neq v$ and $uv \in E(v)$. In this thesis we also consider *weighted* graphs, where there is a function $\rho : V(G) \cup E(G) \to \mathbb{N}$, with $\mathbb{N}$ denoting the set $\{0, 1, 2, \ldots\}$ of natural numbers and $\mathbb{N}^+$ denoting the set $\{1, 2, \ldots\}$ of positive naturals. We then define the *weighted degree* of a vertex v, denoted $d^\rho(v)$, as $\sum_{uv \in E(v)} \rho(uv)$.

Given a graph G, a *path* on k vertices with $k \geq 1$, denoted P_k, is a subgraph of G with vertex set $\{v_1, \ldots, v_k\}$ and edge set $\{v_1v_2, \ldots, v_{k-1}v_k\}$. v_1 and v_k are the *endpoints* of the path. The *length* of a path on k vertices is $k - 1$. Given two vertices $u, v \in V(G)$ the *distance* between u and v, denoted $d(u, v)$ or $d(v, u)$, is the length of a shortest path in G with endpoints u and v. If for every distinct pair of vertices $u, v \in V(G)$ there exists a path in G with u and v as endpoints, then G is *connected*. A *cycle* on k vertices with $k \geq 3$, denoted C_k, in a graph G is a subgraph with vertex set $\{v_1, \ldots, v_k\}$ and edge set $\{v_1v_2, \ldots, v_{k-1}v_k, v_kv_1\}$. The length of a cycle on k vertices is k. If a graph G contains no cycles, then it is a *forest*. If G is a forest and connected, then G is a *tree*. Given a tree G and

a vertex $v \in V(G)$, if v has $d(v) = 1$ then v is a *leaf*. A *rooted tree* is a tree G with a distinguished vertex v called the *root*. Let v be the root of a tree G with leaf set L; if $\max_{u \in L}\{d(v,u)\} = k$ then G has *depth* k. Given a rooted tree G with root $v \in V(G)$, the *branching factor* of a vertex u is the number of neighbours of u that are further from the root than u. The branching factor of the tree G is then the maximum branching factor over all vertices of G. If every non-leaf vertex has branching factor r, then G is an *r-ary* tree. We denote the maximum number of vertices in an r-ary tree of depth k by $\mathrm{tr}(r,k)$. Given an r-ary tree G with root v and leaf set L, if $d(v,u) = k$ for all $u \in L$ then

$$|V(G)| = \mathrm{tr}(r,k) = \sum_{i=0}^{k} r^i = (r^{k+1} - 1)/(r-1).$$

Given a graph G and a set of vertices C, the *boundary* of C, denoted $B(C) \subseteq V(G)$, is the set of vertices not in C that are adjacent to some vertex in C, i.e. $B(C) = \{\, v \in V(G) \setminus C \mid$ there exists a vertex $u \in C$ such that $uv \in E(G) \,\}$.

Graph Editing

In this thesis we consider three *graph editing operations*: *vertex deletion*, *edge deletion* and *edge addition*.

Given a vertex $v \in V(G)$, we delete v from G by setting $V(G) := V(G) \setminus \{v\}$ and $E(G) := E(G) \setminus E(v)$. In an unweighted graph the cost of deleting a vertex is 1. In a weighted graph where $\rho(v)$ is the weight of vertex v, the cost of deleting v is $\rho(v)$.

In an unweighted graph G, given an edge uv, we delete uv from G by setting $E(G) := E(G) \setminus \{uv\}$. The cost of deleting an edge is 1.

In a weighted graph we allow "partial deletion" of an edge, that is, given an edge $uv \in E(G)$ of weight $\rho(uv)$ at a cost of $c \leq \rho(uv)$ we can reduce the weight of uv to $\rho(uv) := \rho(uv) - c$. If $c = \rho(uv)$ then the edge is completely deleted as before as in general we do not allow edges of weight 0.

Given an unweighted graph G, and two vertices u and v such that $uv \notin E(G)$, we add an edge by setting $E(G) := E(G) \cup \{uv\}$. This operation costs 1 to perform.

Given a weighted graph G we may add an edge of weight c between any pair of vertices u and v at a cost of c. If $uv \in E(G)$ then the weight of uv is increased by c, $\rho(uv) := \rho(uv) + c$. If $uv \notin E(G)$ then we set $E(G) := E(G) \cup \{uv\}$ and $\rho(uv) = c$.

Intuitively we may think of an edge of weight c as c parallel edges, then edge deletion in a weighted graph corresponds to removing some or all of the parallel

edges, and edge addition corresponds to adding new parallel edges.

Directed Graphs

The problems considered in this thesis will not involve directed graphs, however the definitions of circuits in Section 2.3.4 require them. A *directed* graph, or *digraph*, $D = (V, A)$ consists of a set V of vertices and a set A of ordered pairs of vertices called the *arc set*, or set of *arcs*, of D. When the vertex and arc sets are not explicitly labelled $V(D)$ denotes the vertex set of the digraph D, and $A(D)$ denotes the arc set of D.

We denote an element of $A(D)$ by (u, v) where $u, v \in V(D)$. Given an arc $(u, v) \in A(D)$, u is the *tail* vertex and v is the *head* vertex of (u, v). (u, v) is an *out* arc of u and an *in* arc of v. We exclude arcs of the form (u, u) from our definition. It no longer holds that $(u, v) = (v, u)$ in a directed graph.

The *in-neighbourhood* of a vertex $v \in V(G)$, denoted $N^-(v)$, is the set $\{\, u \in V(D) \mid (u, v) \in A(D) \,\}$. The *out-neighbourhood* of a vertex $v \in V(G)$, denoted $N^+(v)$, is the set $\{\, u \in V(D) \mid (v, u) \in A(D) \,\}$. The *in-degree* of a vertex v, denoted $d^-(v)$, is $|N^-(v)|$. The *out-degree* of a vertex v, denoted $d^+(v)$, is $|N^+(v)|$.

A *directed path* on k vertices, where $k \geq 1$, in a directed graph is a subdigraph with vertex set $\{v_1, \ldots, v_k\}$ and arc set $\{(v_1, v_2), \ldots, (v_{k-1}, v_k)\}$. A *directed cycle* on k vertices, where $k \geq 2$, is a subdigraph with vertex set $\{v_1, \ldots, v_k\}$ and arc set $\{(v_1, v_2), \ldots, (v_{k-1}, v_k), (v_k, v_1)\}$. If a digraph D contains no directed cycles, then D is *acyclic*.

2.3.2 Propositional Logic

Propositional logic formulae are constructed inductively from propositional variables, negations (NOTs), conjunctions (ANDs) and disjunctions (ORs). Let $\mathcal{X} = \{X_1, X_2, \ldots\}$ be a countably infinite set of propositional variables, let $\neg$ be the negation operator, let $\bigvee$ be the disjunction operator and let $\bigwedge$ be the conjunction operator. When dealing with binary operations we write $a_1 \vee a_2$ (resp. $a_1 \wedge a_2$) as shorthand for $\bigvee_{i \in \{1,2\}} a_i$ (resp. $\bigwedge_{i \in \{1,2\}} a_i$). A *literal* is a variable or negated variable. We write $[m, n]$ for the interval $\{m, \ldots, n\}$, if $m = 1$ we write $[n]$. The rules for constructing propositional formulae are:

1. Any element of $\mathcal{X}$ is a formula.

2. Let α be a formula. Then $\neg\alpha$ is a formula.

3. Let $\alpha_1, \alpha_2, \ldots, \alpha_d$ be formulae. Then $\bigvee_{i\in[d]} \alpha_i$ and $\bigwedge_{i\in[d]} \alpha_i$ are formulae.

4. Nothing else is a formula.

The class of all propositional formulae is denoted PROP. PROP will be important in the definition of the class W[SAT] (see Section 3.3).

For $t \geq 0$ and $d \geq 1$ we inductively define the classes $\Gamma_{t,d}$ and $\Delta_{t,d}$, which will form the basis of one definition of the W-hierarchy (see Section 3.2):

$$\Gamma_{0,d} = \{ \bigwedge_{i\in[c]} \lambda_i \mid 1 \leq c \leq d, \lambda_1, \ldots, \lambda_c \text{ are literals} \}$$

$$\Delta_{0,d} = \{ \bigvee_{i\in[c]} \lambda_i \mid 1 \leq c \leq d, \lambda_1, \ldots, \lambda_c \text{ are literals} \}$$

$$\Gamma_{t+1,d} = \{ \bigwedge_{i\in I} \alpha_i \mid I \text{ is a finite, non-empty index set, and for all } i \in I, \alpha_i \in \Delta_{t,d} \}$$

$$\Delta_{t+1,d} = \{ \bigvee_{i\in I} \alpha_i \mid I \text{ is a finite, non-empty index set, and for all } i \in I, \alpha_i \in \Gamma_{t,d} \}$$

2.3.3 First Order and Second Order Logic

The definitions for relational structures, first order logic and second order logic are drawn from Flum and Grohe [78]. For a dedicated coverage of these topics the monographs of Ebbinghaus and Flum [62] and Ebbinghaus, Flum and Thomas [63] are more complete references.

Relational Structures

A *relational vocabulary* (or just *vocabulary*) τ is a set of *predicates* (or *relation symbols*). The number of variables over which a predicate $R \in \tau$ operates, the *arity*, is denoted arity(R). All predicates $R \in \tau$ have arity(R) ≥ 1. A *τ-structure* $\mathcal{A}$ consists of a set A, the *universe*, and for every $R \in \tau$ a relation $R^{\mathcal{A}} \subseteq A^{\text{arity}(R)}$, the *interpretation* of R.

For the purposes of this thesis it is sufficient to consider only vocabularies that are non-empty and finite, and structures with a finite universe.

Graphs as Structures As will be seen in Section 5.4 it is important that we define how we treat graphs in terms of relational structures. Typically graphs are represented by a relational structure where the vocabulary τ consists of a single predicate E of arity 2 that defines the edge relation, and the universe is the set of vertices of the graph.

For Section 5.4 however it is useful to consider vertices and edges as elements of the universe. To this end we represent a graph G by its *incidence structure* $\mathcal{G}$, with universe $A = V(G) \cup E(G)$ and vocabulary $\tau = \{E, V, I\}$ where $E^{\mathcal{G}} = E(G)$, $V^{\mathcal{G}} = V(G)$ and $I^{\mathcal{G}} = \{ (x, y) \mid x \in V(G) \text{ is incident with } y \in E(G) \}$. In Section 5.4 we will introduce further predicates into τ that will be useful in the particular cases examined there.

First Order Logic

Let $X = \{x_1, x_2, \ldots\}$ be a countably infinite set of *variables*. Let τ be a vocabulary. *Atomic* formulae of vocabulary τ are of the form $x_i = x_j$ where $x_i, x_j \in X$ or $Ry_1 \ldots y_r$ where $y_i \in X$ for $i \in \{1, \ldots, r\}$ and $R \in \tau$ has arity$(R) = r$. *Formulae of vocabulary* τ are constructed inductively using the boolean connectives $\neg$, $\wedge$ and $\vee$, with $a \to b$ as shorthand for $\neg a \vee b$, and the existential and universal quantifiers $\exists$ and $\forall$ by the following rules:

1. Any atomic formula is a first order formula.
2. If ϕ and ψ are first order formulae, then $\phi \wedge \psi$ and $\phi \vee \psi$ are first order formulae.
3. If ϕ is a first order formula, then $\neg\phi$ is a first order formula.
4. If ϕ is a first order formula and x is a variable not quantified in ϕ, then $\exists x\phi$ and $\forall x\phi$ are first order formulae.

We defer the details of the semantic interpretation of first order formulae of vocabulary τ over τ-structures to Section 2.3.3, where it is treated as a special case of the semantics of second order logic.

The set of *free variables* of a formula ϕ is the set of variables x with an occurrence in ϕ that is not in the scope of any quantifier quantifying x. A formula with no free variables is a *sentence*. A formula is *quantifier free* if neither $\exists$ nor $\forall$ appear in the formula. A formula is in *negation normal form* if all negation symbols appear immediately in front of atomic formulae. A formula is in *prenex normal form* (or *prenex form*) if it is of the form $Q_1 x_1 \ldots Q_q x_q \phi$ where ϕ is in negation normal form and is quantifier free, and $Q_i \in \{\exists, \forall\}$ for all $i \in \{1, \ldots, q\}$.

Second Order Logic

Second order logic is an extension of first order logic that allows quantification over subsets of the universe and relations, rather than just elements of the universe. To

the definition of first order logic formulae we add a countably infinite set of *relation variables* $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots\}$ and allow new atomic formulae of the from $\mathcal{X}_i x_1 \ldots x_r$ where $\mathcal{X}_i \in \mathcal{X}$ has arity r, and $x_i \in X$ for all $i \in \{1, \ldots, r\}$. If a relation variable has arity 1, then we call it a *set variable*. If all relation variables are set variables, then the formula is *monadic*. The class of all monadic second order formulae is called *monadic second order logic* and denoted MSO. If a relation variable $\mathcal{X}$ is not quantified in a formula ϕ, then $\mathcal{X}$ is a *free relation variable* of ϕ. We add a new inductive rule; if ϕ is a formula and $\mathcal{X}$ is a free relation variable then $\exists\mathcal{X}\phi$ and $\forall\mathcal{X}\phi$ are formulae.

Given a second order formula ϕ where all relation variables are free, following Flum and Grohe [78] we slightly abuse the terminology and call ϕ a *first order formula with free relation variables*.

The Semantics of First Order and Second Order Logic

In the following we define the semantics of second order logic. The semantics for first order logic arises as a special case. When context allows, we omit τ and refer to a formula of vocabulary τ and a τ-structure simple as a formula and structure respectively.

Given a τ-structure $\mathcal{A}$ with universe A, and a second-order formula ϕ, an *interpretation* of ϕ is a mapping ι that

(a) assigns each free (individual) variable x of ϕ an element $\iota(x) \in A$, and

(b) assigns each free relation variable $\mathcal{Y}$ of ϕ of arity r a relation $\iota(\mathcal{Y}) \subseteq A^r$.

We define whether ϕ *is true in* $\mathcal{A}$ *under* ι as follows.

1. If ϕ is the atomic formula $x_i = x_j$, then ϕ is true in $\mathcal{A}$ under ι if $\iota(x_i) = \iota(x_j)$.
2. If ϕ is the atomic formula $Ry_1 \ldots y_r$ where $R \in \tau$ is a relation symbol of arity r, then ϕ is true in $\mathcal{A}$ under ι if $(\iota(y_1), \ldots, \iota(y_r)) \in R^{\mathcal{A}}$.
3. If ϕ is the atomic formula $\mathcal{Y}y_1 \ldots y_r$ where $\mathcal{Y}$ is a relation variable of arity r, then ϕ is true in $\mathcal{A}$ under ι if $(\iota(y_1), \ldots, \iota(y_r)) \in \iota(\mathcal{Y})$.
4. If $\phi = \neg\phi'$, then ϕ is true in $\mathcal{A}$ under ι if and only if ϕ' is not true in $\mathcal{A}$ under ι.
5. If $\phi = \phi_1 \wedge \phi_2$, then ϕ is true in $\mathcal{A}$ under ι if and only if ϕ_1 and ϕ_2 are true in $\mathcal{A}$ under ι.
6. If $\phi = \phi_1 \vee \phi_2$, then ϕ is true in $\mathcal{A}$ under ι if and only if ϕ_1 or ϕ_2 is true in $\mathcal{A}$ under ι.

7. If $\phi = \forall x \phi'$, then ϕ is true in $\mathcal{A}$ under ι if for all elements $a \in A$, ϕ' is true in $\mathcal{A}$ under the interpretation ι' obtained from ι by setting $\iota(x) = a$.

8. If $\phi = \exists x \phi'$ then ϕ is true in $\mathcal{A}$ under ι if there is at least one element $a \in A$, such that ϕ' is true in $\mathcal{A}$ under the interpretation ι' obtained from ι by setting $\iota(x) = a$.

If ϕ has no free variables then we say that ϕ is true in $\mathcal{A}$ if it is true in the empty interpretation $\iota = \emptyset$; in that case we also say that $\mathcal{A}$ is a *model* of ϕ.

Let ϕ be a first order formula with exactly one free relation variable X of arity s. We say that a set $S \subseteq A^s$ *is a* solution *for* ϕ over $\mathcal{A}$ if ϕ is true in $\mathcal{A}$ for an interpretation ι with $\iota(X) = S$. In most considered cases the free relation variable will be a set variable, and accordingly the solution S is a subset of the universe A.

Let both Σ_0 and Π_0 denote the class of all quantifier free formulae. For $t \geq 1$ we inductively define Σ_t as the class of all formulae of the form

$$\exists x_1 \ldots \exists x_p \phi$$

for all $p \in \mathbb{N}$ where $\phi \in \Pi_{t-1}$ and Π_t as the class of all formulae of the form

$$\forall x_1 \ldots \forall x_p \phi$$

for all $p \in \mathbb{N}$ where $\phi \in \Sigma_{t-1}$.

These classes will help form the basis of two definitions of the W-hierarchy (see Section 3.2).

2.3.4 Circuits

Circuit complexity has a key historical place in parameterized complexity; the W-hierarchy was original conceived in terms of the complexity of certain circuits (see Section 3.2). The definitions we give here are sufficient for the material presented in this thesis. For a full treatment of circuit complexity we refer to Vollmer's monograph [164], from which we derive our notation.

Let $B = \{\text{AND}, \text{OR}, \text{NOT}, \text{true}, \text{false}\}$ where AND, OR and NOT correspond to the familiar boolean operators and true and false to the boolean constants. A *(boolean) circuit* C with n inputs $x_1, \ldots, x_n$ and m outputs $y_1, \ldots, y_m$ consists of an acyclic directed graph $D = (V, A)$, a mapping $\beta : V \to B \cup \{x_1, \ldots, x_n\}$ and a mapping $\omega : V \to \{y_1, \ldots, y_m, \star\}$, where $\star$ indicates that a vertex is not an output

vertex, such that:

1. If $v \in V$ has $d^-(v) = 0$ then $\beta(v) \in \{\text{true}, \text{false}, x_1, \ldots, x_n\}$.
2. If $v \in V$ has $d^-(v) = 1$ then $\beta(v) \in \{\text{AND}, \text{OR}, \text{NOT}\}$.
3. If $v \in V$ has $d^-(v) \geq 2$ then $\beta(v) \in \{\text{AND}, \text{OR}\}$.
4. For all $i \in \{1, \ldots, n\}$ there is exactly one vertex $v \in V$ such that $\beta(v) = x_i$.
5. For all $i \in \{1, \ldots, m\}$ there is exactly one vertex $v \in V$ such that $\omega(v) = y_i$.

Let C be a boolean circuit and D be the associated directed graph. Here D is acyclic, but in Section 6.3 we consider a special case involving cyclic circuits. For $v \in V(D)$ we relax the notation and write $v \in C$. If $v \in C$ has $d^-(v) \geq 1$ then v is a *gate* with *fan-in* $d^-(v)$. If $v \in C$ has $d^+(v) \geq 1$ then v has *fan-out* $d^+(v)$. If $\beta(v) = x_i$ for some $i \in \{1, \ldots, n\}$ then v is an *input gate* or simply *input* for the circuit. If $\omega(v) = y_i$ for some $i \in \{1, \ldots, m\}$ then v is an *output gate* or *output* for the circuit. If $m = 1$ (i.e. the circuit has exactly one output) then C is a *decision circuit*.

For a gate $v \in C$ the *predecessor set* of v is $N^-(v)$, the *successor set* of v is $N^+(v)$.

The *value* of a circuit is computed inductively from its inputs. Let C be a circuit with n inputs and m outputs. Let $\{a_1, \ldots, a_n\} \in \{0,1\}^n$ be the set of input values. For each gate v the value $f(v)$ of v is computed as follows:

1. If $\beta(v) = x_i$ for some $i \in \{1, \ldots, n\}$, then $f(v) = a_i$.
2. If $\beta(v) = \text{true}$, then $f(v) = 1$.
3. If $\beta(v) = \text{false}$, then $f(v) = 0$.
4. If $\beta(v) = \text{NOT}$ and u is the predecessor of v, then $f(v) = 1 - f(u)$.
5. If $\beta(v) = \text{AND}$, then $f(v) = \min_{u \in N^-(v)}\{f(u)\}$.
6. If $\beta(v) = \text{OR}$, then $f(v) = \max_{u \in N^-(v)}\{f(u)\}$.

The value of the circuit is $f(C) = (f(v_1), \ldots, f(v_m))$ where $v_1, \ldots, v_m$ are the outputs of C.

Chapter 3

Parameterized Complexity Theory

In this chapter we introduce the framework of parameterized complexity theory, first developed by Downey and Fellows, and expanded in a series of papers (including [1, 2, 55, 56, 57, 73]) culminating in a formal founding in the form of Downey and Fellow's [59] monograph, collecting the known theory to that point. The area has since expanded greatly and aside from a substantial quantity of publications, has led to two further key monographs, Flum and Grohe's [78] and Niedermeier's [126].

3.1 Basic Definitions

Definition 3.1.1 (Parameterized Problem)**.** Let Σ be a non-empty finite alphabet. A *parameterized problem* is a pair $(\mathcal{P}, \kappa)$ where $\mathcal{P} \subseteq \Sigma^*$ is a language over Σ and κ is a polynomial time computable mapping $\kappa : \Sigma^* \to \mathbb{N}$. κ is called the *parameterization* (of $(\mathcal{P}, \kappa)$).

An *instance* of $(\mathcal{P}, \kappa)$ is a pair (x, k) where $x \in \Sigma^*$ and $k = \kappa(x)$. For decision problems if $(x, k) \in (\mathcal{P}, \kappa)$, then (x, k) is a YES-instance of $(\mathcal{P}, \kappa)$, otherwise it is a NO-instance. In this thesis we will be mostly concerned with decision problems; if a problem is not a decision problem, this will be clear from context. When the parameter is also part of the input we write the instance as (x, k), rather than the more correct $((x, k), k)$. Informally we will refer to the witness that verifies a YES-instance as a *solution*; for example the k vertices that make up the vertex cover for a YES-instance of VERTEX COVER. If we have a parameterized problem $(\mathcal{P}, \kappa)$, then $\mathcal{P}$ is the un-parameterized, traditional version of the problem. A parameterized

problem will normally be presented in this thesis in the following form:

$\mathcal{P}$-Problem
Instance: $x \in \Sigma^*$.
Parameter: $\kappa(x)$.
Question: Is $x \in \mathcal{P}$?

If the parameterization is undecided when the problem is defined or multiple parameterizations are involved, the problem may be presented in the traditional form with the particular parameterization specified later:

$\mathcal{P}$-Problem
Instance: $x \in \Sigma^*$.
Question: Is $x \in \mathcal{P}$?

Typically the mapping κ will also remain unspecified, with the parameter drawn from the input implicitly.

$\mathcal{P}$-Problem
Instance: $x \in \Sigma^*$, a positive integer k.
Parameter: k.
Question: Is $x \in \mathcal{P}$?

Definition 3.1.2 (Fixed-parameter Tractability)**.** Let $(\mathcal{P}, \kappa)$ be a parameterized problem and (x, k) an instance of that problem with $|x| = n$. $(\mathcal{P}, \kappa)$ is *fixed-parameter tractable* if and only if there exists an algorithm **A** that solves the problem in time

$$f(k) \cdot p(n)$$

where f is a computable function and p is a polynomial with degree independent of the parameter. If such an algorithm exists then $\mathcal{A}$ is called an *fpt-algorithm.*

FPT is the class of all fixed-parameter tractable problems.

If f is also a polynomial, then the problem is in P. Thus FPT generalises the traditional notion of tractability.

Note that Definitions 3.1.1 and 3.1.2 correspond to that of Flum and Grohe [78], rather than Downey and Fellows [59]. Downey and Fellows originally gave a more expansive, and less formal definition where the parameter was simply a second input drawn from Σ^*, and thus was not necessarily numeric. Furthermore Downey and Fellows do not require that the function f in the definition be computable.

However Flum and Grohe's definition corresponds to Downey and Fellows' *strongly uniform fixed-parameter tractability*, which suffices for all natural problems and parameterizations, and certainly does not affect the results in this thesis.

We now demonstrate an alternative definition of fixed-parameter tractability.

Proposition 3.1.3 (Additive FPT [59])**.** *Let $(\mathcal{P}, \kappa)$ be a parameterized problem and (x, k) an instance of that problem with $|x| = n$. $(\mathcal{P}, \kappa)$ is fixed-parameter tractable if and only if there exists an algorithm $\mathbf{B}$ that solves the problem in time*

$$g(k) + q(n)$$

where g is a computable function and q is a polynomial with degree independent of the parameter.

Proof. ($\Rightarrow$) Assume $(\mathcal{P}, \kappa) \in$ FPT. There exists an algorithm $\mathbf{A}$ that solves the problem in time $f(k) \cdot p(n)$. By the inequality $ab \leq a^2 + b^2$, this algorithm solves the problem in time $f(k)^2 + p(n)^2$.

($\Leftarrow$) Assume $(\mathcal{P}, \kappa)$ has such an algorithm $\mathbf{B}$. By the inequality $a+b \leq a \cdot (b+1)$ we have $g(k) + q(n) \leq g(k) \cdot (q(n) + 1) = g(k) \cdot q'(n)$ where $q'(n) = q(n) + 1$. □

Therefore the additive definition of fixed-parameter tractability is equivalent to the multiplicative definition.

Before introducing the first set of parameterized intractability classes we must define an adequate notion of reducibility.

Definition 3.1.4 (FPT Reductions)**.** Let $(\mathcal{P}_1, \kappa_1)$ and $(\mathcal{P}_2, \kappa_2)$ be parameterized problems. An FPT *many-one reduction* (or FPT *reduction* for short) from $(\mathcal{P}_1, \kappa_1)$ to $(\mathcal{P}_2, k_2)$ maps an instance (x, k) of $(\mathcal{P}_1, \kappa_1)$ to an instance (x', k') of $(\mathcal{P}_2, \kappa_2)$ such that:

1. x is a YES-instance of $(\mathcal{P}_1, \kappa_1)$ if and only if x' is a YES-instance of $(\mathcal{P}_2, \kappa_2)$.
2. $k' \leq g(k)$ for some computable function g.
3. The mapping can be computed in time bounded by $f(k)p(|x|)$ for some computable function f and polynomial p.

If such a mapping exists, then $(\mathcal{P}_1, \kappa_1)$ is (FPT) reducible to $(\mathcal{P}_2, \kappa_2)$. If f is also a polynomial, then the reduction is a *polynomial time* FPT reduction.

Given a parameterized problem $(\mathcal{P}, \kappa)$, the *closure* of $(\mathcal{P}, \kappa)$ under FPT-reductions is the set $[(\mathcal{P}, \kappa)]^{\text{FPT}}$ of all problems FPT reducible to $(\mathcal{P}, \kappa)$. Similar to

the closure of P under polynomial time many-one reductions, FPT is closed under FPT reductions, i.e. given two parameterized problems $(\mathcal{P}_1, \kappa_1)$ and $(\mathcal{P}_2, \kappa_2)$ such that $(\mathcal{P}_1, \kappa_1)$ is reducible to $(\mathcal{P}_2, \kappa_2)$, if $(\mathcal{P}_2, \kappa_2) \in$ FPT then $(\mathcal{P}_1, \kappa_1) \in$ FPT.

Given a parameterized complexity class X and a parameterized problem $(\mathcal{P}, \kappa)$, if there is an FPT reduction from every problem in X to $(\mathcal{P}, \kappa)$, then $(\mathcal{P}, \kappa)$ is X-hard. If $(\mathcal{P}, \kappa)$ is X-hard and also in X, then $(\mathcal{P}, \kappa)$ is X-complete.

Given two parameterized problems $(\mathcal{P}_1, \kappa_1)$ and $(\mathcal{P}_2, \kappa_2)$, and a parameterized complexity class X such that $(\mathcal{P}_2, \kappa_2)$ is X-hard, if there is an FPT-reduction from $(\mathcal{P}_2, \kappa_2)$ to $(\mathcal{P}_1, \kappa_1)$ then $(\mathcal{P}_1, \kappa_1)$ is also X-hard.

3.2 The W-Hierarchy

Just as NP-hardness forms the basis of the traditional idea of intractability, parameterized complexity has a notion of parameterized intractability, to demonstrate when a problem is unlikely to have an fpt-algorithm. However this notion is built not on one class, but a series of classes organised into hierarchies. We introduce the most fundamental, in a practical sense, of these, the W-hierarchy.

The W-hierarchy consists of a series of classes $W[t]$, $t \in \mathbb{N}^+$ such that $W[t] \subseteq W[t+1]$ for all t and two additional classes $W[\mathrm{SAT}]$ and $W[P]$, which will be discussed separately. It remains an open question as to whether the inclusions in the hierarchy are strict, though they are generally believed to be.

The W-hierarchy can be defined in a series of equivalent ways, the first we present is the *weighted Fagin definability* version.

Let ϕ be a first order formula with one free relation variable. The parameterized weighted Fagin definability problem for ϕ is:

> WD_ϕ
>
> *Instance:* A structure $\mathcal{A}$, a non-negative integer k.
>
> *Parameter:* k.
>
> *Question:* Is there a solution S for ϕ with $|S| = k$?

If Φ is a class of first order formulas, then the class of problems WD-Φ is the class of weighted Fagin definability problems where $\phi \in \Phi$. Recall from Section 2.3.3 that Π_1 is the class of prenex form first order formulae with only universal quantification.

Example 3.2.1. The following formula expresses the problem Vertex Cover:

$$vertexcover(X) := \forall x \forall y \forall e(Ixe \wedge Iye \wedge Vx \wedge Vy \rightarrow (Xx \vee Xy))$$

Thus Vertex Cover is in WD-Π_1.

Now we may define the W-hierarchy. Recall that Π_t is the class of prenex form first-order formulae with at most t alternations of existential and universal quantification, beginning with universal (Section 2.3.3).

Definition 3.2.2 (W-hierarchy)**.** For every $t \in \mathbb{N}^+$,

$$W[t] = [\text{WD-}\Pi_t]^{\text{FPT}}$$

Combining Definition 3.2.2 and Example 3.2.1, we can see that Vertex Cover is in $W[1]$.

It can also be shown [78] that WD-$\Sigma_{t+1} \subseteq$ WD-Π_t for $t \geq 1$, and that WD-$\Sigma_1 \subseteq$ FPT.

Downey and Fellows' [59] original formulation of the classes $W[t]$ is based on a family of circuit satisfiability problems where the decision circuit consists of *large* gates with unbounded fan-in, and *small* gates with bounded fan-in (the precise value of the bound is unimportant [59]). The *weft* of the circuit is the largest number of large gates on any path from the inputs to the ouput of the circuit. The W-hierarchy derives its name from circuit weft. The *depth* of the circuit is the largest number of gates of any size on any path from the inputs to the output of the circuit.

> Weighted Weft t Depth h Circuit Satisfiability (WCS(t, h))
> *Instance:* A weft t depth h decision circuit C.
> *Parameter:* A positive integer k.
> *Question:* Does C have a weight k satisfying assignment?

This problem gives the proposition:

Proposition 3.2.3 ([59])**.** *Let $t \in \mathbb{N}^+$. For every $h \in \mathbb{N}$:*

$$W[t] = [\text{WCS}(t, h)]^{\text{FPT}}$$

Downey and Fellows [59] also give a characterisation in terms of *weighted satisfiability* problems. Let F be a class of propositional formulae (see Section 2.3.2). A formula is k-satisfiable if there is a satisfying assignment where k variables are set

to TRUE. The weighted satisfiability problem for F is:

WSAT(F)

Instance: A formula $f \in F$ and a positive integer k.

Parameter: k.

Question: Is f k-satisfiable?

Recall from Section 2.3.2 that $\Gamma_{t,d}$ is the class of propositional formulae with at most t alternations of $\bigwedge$ and $\bigvee$, beginning with $\bigwedge$ and having at most d literals per clause.

Proposition 3.2.4 ([59]). *Let $t \in \mathbb{N}^+$. For every $d \in \mathbb{N}^+$:*

$$W[t] = [\mathrm{WSAT}(\Gamma_{t,d})]^{\mathrm{FPT}}$$

The equivalence of the definitions of the W-hierarchy from Propositions 3.2.4 and 3.2.3 is given by Downey and Fellows [56], and the equivalence of the definitions of the W-hierarchy from Definition 3.2.2 and Proposition 3.2.4 is given by Downey *et al.* [60].

Flum and Grohe [77] give a further definition of the W-hierarchy in terms of *model checking* problems. Let Φ be a class of (first order) formulae. Then the parameterized model checking problem for Φ is:

MC(Φ)

Instance: A structure $\mathcal{A}$ and a formula $\phi \in \Phi$.

Parameter: $|\phi|$.

Question: Is A a model of ϕ?

Recall from Section 2.3.3 that a prenex form first order formula is in Σ_t if it has at most t alternations of existential and universal quantification, beginning with existential. Let $\Sigma_{t,d}$ be the class of Σ_t sentences where all quantifier blocks after the leading existential block have length most d.

Proposition 3.2.5 ([77]). *Let $t \in \mathbb{N}^+$. For every $d \in \mathbb{N}^+$:*

$$W[t] = [\mathrm{MC}(\Sigma_{t,d})]^{\mathrm{FPT}}$$

If d is removed from the definition (i.e. is unbounded), then we obtain the definition for the A-hierarchy, an alternative parameterized hierarchy. It is known that $A[1] = W[1]$, and that $W[t] \subseteq A[t]$ for all t. However whether the containment is

strict for any level above the first is not known. Flum and Grohe [78] provide the following argument that it is unlikely that $A[t] \subseteq W[t]$ for $t \geq 1$. The classes $W[t]$ are defined by the problems WD-Π_t, which when viewed as non-parameterized problems are all in NP. The defining problems $\mathrm{MC}(\Sigma_t)$ for the classes $A[t]$ correspond to complete problems for the Polynomial Hierarchy [155], and it seems unlikely the two would be equivalent.

Of course to this point all these definitions have been relative to a collection of chosen complete problems which although practical may seem unsatisfying with regards to any underlying method of computation. However the W-hierarchy can also be defined in terms of machine models. Chen and Flum [37] give a characterisation using an alternating Turing machine model. Islam [92] gives a much more workable machine model. As this thesis does not deal with machine characterisations we refer to the cited literature for further discussion and definitions.

3.2.1 A Note on Containment

For every $t \geq 1$, $W[t] \subseteq W[t+1]$, by virtue of the similar containment of the defining classes Π_t. However it may not be immediately clear that FPT $\subseteq W[1]$.

Proposition 3.2.6. FPT $\subseteq W[1]$.

Proof. Let $(\mathcal{P}, \kappa) \in$ FPT be a parameterized problem. By Definition 3.1.2 there exists an algorithm $\mathbf{A}$ that solves each instance (x, k) of $(\mathcal{P}, \kappa)$ in time $f(k) \cdot p(n)$ where $|x| = n$, f is a computable function and p is a polynomial.

We construct a reduction that maps an instance (x, k) of $(\mathcal{P}, \kappa)$ to an instance of a problem in WD-Π_1 by running $\mathbf{A}$ on (x, k). If (x, k) is a YES-instance, the WD-Π_1 problem instance is the formula $\forall y(Xy \vee \neg Xy)$ where X is the free set variable. If it is a NO-instance, the WD-Π_1 problem instance is the formula $\forall y(Xy \wedge \neg Xy)$. In the first case both are YES-instances, and in the second both are NO-instances. Furthermore it is clear that reduction can be computed in time $O(f(k)p(n))$. Therefore $(\mathcal{P}, \kappa) \in W[1]$. □

3.3 W[SAT] and W[P]

So far the W-hierarchy classes considered have been defined in terms of problems for a series of fragments of logic. The next natural step is to consider the complexity classes obtained when the logic is less restricted.

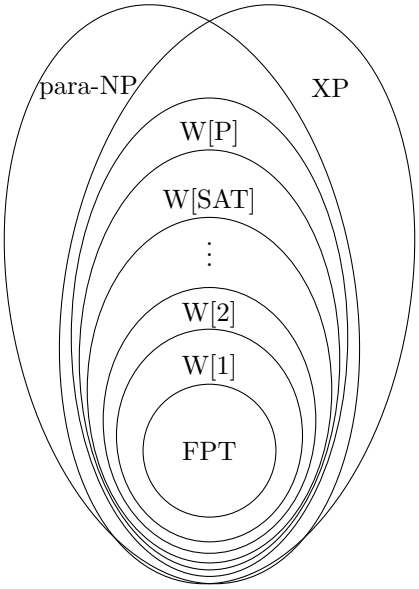


Figure 3.1: The relationships of major parameterized classes.

Consider the weighted satisfiability definition of the $W[t]$ classes (Proposition 3.2.4). If we choose F to be the class of all propositional logic formulae then we obtain the original definition of $W[\mathrm{SAT}]$ [56].

Definition 3.3.1. Let PROP be the class of all propositional logic formulae.

$$W[\mathrm{SAT}] = [\mathrm{WSAT}(\mathrm{PROP})]^{\mathrm{FPT}}$$

Papadimitriou and Yannakakis [132] give an alternative definition in terms of a parameterized model checking problem.

v-MC(Φ)
Instance: A structure $\mathcal{A}$ and a formula $\phi \in \Phi$.
Parameter: The number of variables in ϕ.
Question: Is A a model of ϕ?

Proposition 3.3.2 ([132])**.**

$$W[\mathrm{SAT}] = [v\text{-MC}(\Sigma_1)]^{\mathrm{FPT}}$$

Propositional formulae can be simulated by boolean circuits, thus it is natural to extend the definition of WSAT to boolean circuits. A decision circuit C is k-satisfiable if there exists an assignment to the inputs that gives an output of 1 with k inputs set to 1. Using the class of circuits in place of propositional logic classes

in the WSAT problem gives Downey and Fellows' [56] definition of $W[P]$.

Proposition 3.3.3 ([56]). *Let CIRC be the class of all decision circuits.*

$$W[P] = [\mathrm{WSAT}(\mathrm{CIRC})]^{\mathrm{FPT}}$$

Combining Definitions 3.2.4, 3.3.1 and 3.3.3 and Proposition 3.2.6 we can see the structure of the W-hierarchy:

$$\mathrm{FPT} \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[t] \subseteq \ldots \subseteq W[\mathrm{SAT}] \subseteq W[P]$$

Thus hardness of a problem for any class in the W-hierarchy gives evidence that the problem is not in FPT similar to NP-hardness indicating that a problem is unlikely to be in P.

Chen *et al.* [38] provide a particularly useful alternative definition of $W[P]$ based on Turing Machines. A nondeterministic Turing Machine M is κ-restricted if there are computable functions f and g and a polynomial p such that on input (x, k), M performs at most $f(k)p(n)$ steps, $g(k)\log n$ of them being nondeterministic, where $n = |x|$ and $k = \kappa(x)$.

Proposition 3.3.4 ([38]). *$W[P]$ is the class of problems $(\mathcal{P}, \kappa)$ that can be solved by a κ-restricted nondeterministic Turing Machine.*

This definition gives an intuitive method of showing that a problem is in $W[P]$ by a simple guess-and-check algorithm. Assuming the solution is of size k, each element of the solution can be encoded in $\log n$ bits, and the solution can be verified in time bounded by $f(k)p(n)$, where f is a computable function and p is a polynomial, it suffices to nondeterministically guess k elements as the solution and verify the correctness of the guesses. For example Dominating Set can be solved on such a Turing Machine by guessing k vertices then verifying in polynomial time that the vertices form a dominating set (i.e. that all vertices in the graph are either in the set or are adjacent to one of the vertices in the set).

3.4 Para-NP and XP

The W-hierarchy is contained in two more general classes. para-NP results from mimicking the relationship between P and NP by replacing 'algorithm' with 'non-deterministic' algorithm in the definition of FPT.

Definition 3.4.1 (Para-NP)**.** Let $(\mathcal{P}, \kappa)$ be a parameterized problem. $(\mathcal{P}, \kappa)$ is in para-NP if there is a nondeterministic algorithm $\mathbf{A}$ that on input (x, k) where $|x| = n$ and $k = \kappa(x)$ solves $(\mathcal{P}, \kappa)$ in time

$$f(k) \cdot p(n)$$

where f is a computable function and p is a polynomial.

Note that if $\mathcal{P} \in$ NP, then no matter what paramerization κ is chosen, $(\mathcal{P}, \kappa) \in$ para-NP. In fact the relationship between NP and para-NP can be strengthened further.

Proposition 3.4.2 ([78])**.** FPT = para-NP *if and only if* P = NP*.*

Consider the following parameterization of the colourability problem:

COLOURABILITY
Instance: A graph G, a positive integer k.
Question: Does G have a proper vertex colouring with k colours?

As COLOURABILITY is in NP, COLOURABILITY parameterized by k is in para-NP. As COLOURABILITY is NP-hard even when k is fixed at 3 we can see that COLOURABILITY cannot be in FPT with parameter k unless P = NP. This simple observation can be extended to a general statement. Let $(\mathcal{P}, \kappa)$ be a parameterized problem. The l^{th} *slice* of $(\mathcal{P}, \kappa)$ is the problem

$$(\mathcal{P}, \kappa)_l = \{\, x \in (\mathcal{P}, \kappa) \mid \kappa(x) = l \,\}.$$

It is clear that if $(\mathcal{P}, \kappa) \in$ FPT, then $(\mathcal{P}, \kappa)_l \in$ P. If $(\mathcal{P}, \kappa)_l \in$ P for each l then $(\mathcal{P}, \kappa)$ is *slice-wise polynomial.*

Proposition 3.4.3 ([76])**.** *If* $(\mathcal{P}, \kappa)_l$ *is* NP*-complete for some* $l \in \mathbb{N}$ *and* $\mathcal{P} \in$ NP*, then* $(\mathcal{P}, \kappa)$ *is* para-NP*-complete.*

Thus COLOURABILITY is para-NP-complete when parameterized by k. However Proposition 3.4.3 largely gives the intuition that para-NP is in fact a poor characterisation of parameterized intractability. Problems such as CLIQUE and DOMINATING SET which appear to have no fpt-algorithm for their natural parameterizations (and are in fact $W[1]$-complete and $W[2]$-complete respectively) are slice-wise polynomial.

The notion of slicewise polynomial however provides the basis for the class XP [59].

Definition 3.4.4. nonuniform-XP is the class of all parameterized problems $(\mathcal{P}, \kappa)$ such that $(\mathcal{P}, \kappa)_l \in \mathrm{P}$ for all $l \geq 1$.

From this we can derive the following observation:

Proposition 3.4.5. *If* $\mathrm{P} \neq \mathrm{NP}$ *then* para-NP $\not\subseteq$ nonuniform-XP.

Proof. If para-NP $\subseteq$ nonuniform-XP then Colourability with parameter k is slice-wise polynomial; in particular 3-Colourability is in P, therefore P = NP. □

However nonuniform-XP contains undecidable problems [78], therefore the following definition is generally more preferable:

Definition 3.4.6 (XP)**.** XP is the class of all parameterized problems $(\mathcal{P}, \kappa)$ such that given an instance (x, k) of $(\mathcal{P}, \kappa)$ with $|x| = n$, there is an algorithm that solves (x, k) in time $p_{f(k)}(n)$ where $p_{f(k)}$ is a polynomial of degree $f(k)$.

XP is one of the few classes about which a strict containment statement can be made.

Proposition 3.4.7 ([78])**.** FPT $\subsetneq$ XP.

The relationships of the classes discussed in Sections 3.1, 3.2, 3.3 and 3.4 are shown in Figure 3.1.

3.5 Techniques in Parameterized Complexity

The field of parameterized complexity is rich in techniques for classifying the complexity of parameterized problems and particularly for demonstrating fixed-parameter tractability. In this section, we will outline the main techniques and provide illustrative examples of their application.

3.5.1 Non-Constructive and Non-Practical Techniques

First we will examine techniques that demonstrate fixed-parameter tractability either without an accompanying algorithm, or with an algorithm with a running time that renders the algorithm infeasible to implement. However these techniques can still be readily used to obtain classifications.

Graph Minors

A graph H is a *minor* of a graph G if H can be obtained from G by a finite number of edge contractions, edge deletions and vertex deletions. An edge contraction is conducted by taking an edge uv, and creating a new vertex w such that $N(w) = (N(u) \cup N(v)) \setminus \{u, v\}$, and deleting u and v. Determining whether such a set of contractions and deletions exist for a given pair G and H is called *minor testing*.

In a series of twenty papers (starting with [143] and ending with [144]) Robertson and Seymour demonstrated one of the deepest results in modern graph theory. In brief it states that for any family $\mathcal{F}$ of (finite) graphs closed under taking minors (closed under the *minor order*), there exists a finite *obstruction set* of graphs such that a graph G belongs to $\mathcal{F}$ if and only if G does not contain any of the obstructions as a minor. Moreover, given a graph G, the problem of determining whether an obstruction is a minor of G is fixed-parameter tractable where the size of the obstruction is the parameter.

Thus any problem of determining whether a graph has a property that is closed under the minor order (i.e. given a graph G with the property, all minors of G also have the property) is fixed-parameter tractable by the Graph Minor Theorem where the parameter is the number of vertices in the largest obstruction graph. Vertex Cover is one such problem; if a graph G has a vertex cover of size at most k, then any minor of G will also have a vertex cover of size at most k. Dominating Set, which is $W[2]$-complete [59] clearly does not succumb to this approach. An induced cycle of length greater than $3k$ forms an obstruction for the existence of a dominating set of size at most k. Consider such a cycle, where each vertex on the cycle is connected to a central vertex. This central vertex forms a dominating set for the graph. If we remove the central vertex however, the size of the minimum dominating set increases (this increase can be made arbitrarily large). Therefore the property of having a dominating set of size at most k is not closed under the minor order, and there is no finite obstruction set.

The problems with using Graph Minor theory as a practical approach are twofold. First, although testing whether H with $|V(H)| = k$ is a minor of G with $|V(G)| = n$ has a running time of $O(f(k)n^3)$, f is a fast growing function and thus proves impractical even compared to naïve, brute force fixed-parameter algorithms. Second, Robertson and Seymour's proof is non-constructive. Thus it gives no way of determining the finite obstruction set. However some progress has been made on algorithms to compute excluded minors by Adler *et al.* [6] and Demaine *et al.* [52].

Graph Minor theory allows a quick determination of whether certain problems are fixed-parameter tractable, and whether it is possibly worthwhile pursuing a practical algorithm. However if a property is not closed under the minor order, then the associated problem may or may not be fixed-parameter tractable.

Bounded Treewidth and Courcelle's Theorem

It is often the case that when the input of a hard problem is restricted to trees, it becomes easy; for example Colouring can be decided in time $O(1)$ (all trees can be coloured with two colours) and a colouring can be found in time $O(|V|)$. Thus it is often useful to have a graph that is, in a certain sense, *almost* a tree.

Following Niedermeier's definition [126], a *tree decomposition* of a graph $G = (V, E)$ consists of a set $X = \{ X_i \mid i \in I \}$ where each element X_i (called a *bag*) is a set of vertices of G, and an accompanying tree where the nodes are elements of I with the following properties:

1. $\bigcup_{i \in I} X_i = V$.
2. For every $uv \in E$, there exists an i such that $\{ u, v \} \subseteq X_i$.
3. For every $i, j, k \in I$, if j lies on the path in T from i to k, then $X_i \cap X_k \subseteq X_j$.

The *width* of a decomposition is given by $w = \max\{ |X_i| \mid i \in I \} - 1$. Then the *treewidth* $\mathrm{tw}(G)$ of a graph G is given by the minimum k such that G has a tree decomposition of width k.

Thus a nontrivial tree may be decomposed into a series of bags of size 2 (effectively the edges), and has treewidth 1. If a graph has small treewidth, then it may be easier to deal with than a general graph. Practical approaches to using treewidth will be discussed later, however fixed-parameter tractability for graphs of bounded treewidth is often relatively simple to determine using Monadic Second-Order Logic (MSO), and Courcelle's Theorem.

Courcelle's Theorem, stated and developed over several publications (beginning with [44] and finishing with [45]) states that given a graph G of treewidth at most k, and an MSO formula ϕ, then it can be determined in time $O(f(k+|\phi|) \cdot (|V(G)| + |E(G)|))$ whether G is a model for ϕ (see Section 2.3.3 for the semantics of second order logic). In other words, if a graph property can be expressed in MSO logic, then it is fixed parameter tractable to determine whether a graph of bounded treewidth has that property or not where the parameter is the treewidth of the graph plus the length of the formula expressing the property.

For example, we may express the property of a graph being k-colourable:

$$\exists X_1 \dots X_k \forall x \forall y \forall e((Vx \wedge Vy \wedge x \neq y \wedge Ixe \wedge Iye \rightarrow \bigwedge_{i\in[k]} \neg(X_i x \wedge X_i y)) \wedge (Vx \rightarrow \bigvee_{i\in[k]} X_i x))$$

Thus Colourability parameterized by the treewidth of the input graph and the number of colour classes k is fixed-parameter tractable.

However, much like Graph Minors, the function f in the running time is unfeasibly large, with enormous hidden constants, thus Courcelle's Theorem is entirely of theoretical significance. However Courcelle's Theorem is at least constructive. Despite the algorithmic limitations, Courcelle's Theorem is an invaluable tool, as fixed-parameter algorithms for bounded treewidth generally rely on dynamic programming, and are complex and difficult. Courcelle's Theorem provides a quick and easy method of determining the complexity of a problem with respect to bounded treewidth.

This does not mean that treewidth cannot be used in a practical setting although a key problem here is producing a tree decomposition of small width. Bodlaender [22] gives a linear time fixed-parameter algorithm to decide if a graph G has treewidth k, and if so produce a tree decomposition of G with width k, where k is the parameter. However the hidden constant in the running time is prohibitively large. Most commonly Reed's [142] 4-approximation algorithm for tree decompositions is used, Bodlaender [20] gives some recent perspective on this. Once a tree decomposition is produced however, other (fixed-parameter) techniques are generally applied, most commonly dynamic programming.

Bounded Local Treewidth and First Order Logic

Frick and Grohe [79] give a meta-theorem thematically similar to Courcelle's Theorem. Given a τ-structure $\mathcal{A}$ with universe A we define the *Gaifman graph* of $\mathcal{A}$ as the graph $G(\mathcal{A}) = (V, E)$ where $V = A$ and ab with $a, b \in A$ and $a \neq b$ is an edge if and only if there exists $R \in \tau$ such that $a, b \in \{x_1, \dots, x_{\text{arity}(R)}\}$ for some $(x_1, \dots, x_{\text{arity}(R)}) \in R^{\mathcal{A}}$. Note that if a graph has maximum degree d then the Gaifman graph of its incidence structure has maximum degree d. The distance between two elements of A is their distance in $G(\mathcal{A})$. Let $N_r^G(v) = \{\, u \in V \mid d(u, v) \leq r \,\}$ be the *r-neighbourhood* of v, and let $[N_r^G(v)]$ be the graph induced by the r-neighbourhood of v. The *local treewidth at depth r* $\text{ltw}_r(\mathcal{A})$ of a structure $\mathcal{A}$ is $\max_{v\in V(G(\mathcal{A}))} \text{tw}([N_r^G(v)])$. The local treewidth of a class of structures C is *effec-*

tively bounded if there is a computable function f such that $\text{ltw}_r(\mathcal{A}) \leq f(r)$ for all $\mathcal{A} \in C$ and all $r \geq 1$.

Classes with effectively bounded local treewidth include the class of planar graphs, the classes of graphs of bounded treewidth, and notable for our purposes, the classes of graphs of bounded degree.

Let MC(C,FO) denote the model checking problem where the formula ϕ is a first order formula and the structure $\mathcal{A}$ is from the class C. Bounded local treewidth becomes useful when combined with the following theorem:

Theorem 3.5.1 ([79]). *Let C be a class of structures of effectively bounded local treewidth.* MC(C,FO) *is fixed-parameter tractable where the parameter is a function of the length of the input formula.*

As graphs of bounded degree are structures with effectively bounded local treewidth we obtain the following useful corollary:

Corollary 3.5.2 ([79]). *Let $\mathcal{G}_d$ be the class of graphs with degree at most $d \in \mathbb{N}$.* MC($\mathcal{G}_d$,FO) *is fixed-parameter tractable where the parameter is a function of the length of the input formula.*

Stewart [154] demonstrates that Frick and Grohe's proof of Theorem 3.5.1 establishes a stronger result. Let C be a class of pairs $(\mathcal{A}, \phi)$ where $\mathcal{A}$ is a structure and ϕ a first order formula, and assume that there exists a function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for $(\mathcal{A}, \phi) \in C$ we have $\text{ltw}_r(\mathcal{A}) \leq f(r, |\phi|)$. Then the problem of deciding if $\mathcal{A}$ is a model of ϕ for $(\mathcal{A}, \phi) \in C$ is fixed parameter tractable for parameter $|\phi|$. This generalisation of Theorem 3.5.1 gives the following corollary:

Corollary 3.5.3 ([154]). *Let C be a class of structures.* MC(C,FO) *is fixed-parameter tractable with parameter $d + l$ where d is the maximum degree of the structure and l is the length of the formula.*

3.5.2 Bounded Search Trees

Employing a search tree to solve a problem is by no means a new technique, and as a parameterized technique it has been recognised from the founding of the field. The key to developing a search tree algorithm is that of bounding the size of the tree by a function of the parameter, with each branching step performable in polynomial time. The *branching factor* of a bounded search tree is the maximum number of children of any vertex in the tree. Typically a tree is designed with a constant bound on the branching factor, along with a depth bound described by a function of the

parameter. However as long as the tree size is bounded entirely by a function of the parameter, and each step is performable in "FPT-time", the resulting algorithm will still be fixed-parameter tractable. Given a search tree G with branching factor r and depth k (see Section 2.3.1), we denote the maximum number of vertices of G by $\mathrm{tr}(r, k)$.

The classic example is that of Vertex Cover, which admits a very simple bounded search tree, but also has been studied extensively (Balasubramanian *et al.* [13], Downey *et al.* [54], Niedermeier and Rossmanith [127, 129], Chen *et al.* [34], Chen *et al.* [35] and Chandran and Grandoni [31] give significant contributions). If we consider an edge in the graph, then at least one of the endpoints must be in the vertex cover, giving a branching factor of 2. As there may only be at most k vertices in the cover, the depth is bounded by k. Thus the search tree has size $O(2^k)$. The current best, due to Chen *et al.* [35], stands at $O(1.2738^k)$, and uses more sophisticated branching rules and interleaved kernelization steps.

Many problems are amenable to a bounded search tree approach, and the technique is quite practical as the branching steps in the search tree are essentially a map for the recursive calls of the underlying algorithm, thus a bounded search tree approach leads directly to an algorithm, with running time inherently similar to the size of the search tree (perhaps plus some overhead or auxiliary processing). Therefore much effort is focussed on producing clever branching strategies that reduce the size of the tree, although sometimes an 'improvement' is a matter of closer analysis of the complex recurrences produced by the branching strategy.

Example problems that admit a bounded search tree based algorithm include Cluster Editing [83], d-Hitting Set [128] and Dominating Set for graphs of bounded genus [67]. For further examples of the application of bounded search trees consult the texts of Niedermeier [126] or Downey and Fellows [59].

3.5.3 Dynamic Programming

Dynamic programming is possibly one of the most widely applied algorithmic techniques in computer science. The basic idea is that by storing in a table the solutions to previously calculated sub-problems, the cost of conducting in effect an exhaustive search is reduced. For a general introduction one may consult any algorithms text (for example the popular text by Cormen *et al.* [42]). In the context of parameterized complexity, dynamic programming is still an important technique. Dynamic programming directly produces fixed-parameter tractable algorithms when the size

of the table (implicitly) generated is bounded by a function of the parameter alone, and each entry can be calculated in "FPT-time" (though typically it will be polynomial).

There are many fixed-parameter tractability results that employ dynamic programming algorithms for at least part of the process. For example Dreyfus and Wagner [61] present an algorithm for the Steiner Problem in Graphs that although not recognised at the time of publication, is a fixed-parameter algorithm. Mölle *et al.* [120] give a recent improvement to this. Guo and Niedermeier [87] give an algorithm for Multicommodity Demand Flow in Trees. An application to Vertex Cover is shown by Niedermeier and Rossmanith [129] and improved by Chandran and Grandoni [31].

3.5.4 Iterative Compression

Iterative compression is one of the newest techniques to enter the parameterized complexity toolbox. The technique was introduced by Reed *et al.* [141] for Graph Bipartization. Hüffner improves the algorithm and provides a clear explanation of its working [90].

The fundamental premise of iterative compression is to provide a fixed-parameter algorithm called the *compression step* that, given a solution of size $k + 1$, either produces a solution of size k, or correctly states that there is no such solution. In practice, the input is slowly reconstructed from an empty input (or some base case), where the optimal solution is easy or trivial to calculate, and each component of the input is added individually to the solution, then the compression step is run. This guarantees that at each step, the solution is optimal for that particular sub-instance.

Iterative compression has been applied in several interesting cases. Guo *at al.* [86] and Dehne *et al.* [49] independently provide similar algorithms for the Feedback Vertex Set problem. The long standing open problem Directed Feedback Vertex Set was shown to be fixed-parameter tractable using iterative compression [36], along with the subsidiary problem Ordered Multi-cut in DAGs [140], indicating that iterative compression may provide a new inroads to problems that have not yet proven amenable to other approaches.

3.5.5 Greedy Localization

Greedy localization is another relatively new technique, that is particularly suited to maximisation problems, and is in a sense, the reverse of iterative compression. The general technique is to greedily compute a maximal solution to the problem

being considered, and then either this solution is sufficient (i.e. large enough), or it provides a starting point from which we can construct an optimal solution in "FPT-time".

This technique was first applied to the Set Splitting problem by Dehne *et al.* [50]. Greedy localization also proves fruitful for Set Packing [71] and Star Packing [136].

3.5.6 Colour-Coding

Colour-coding is another long standing technique in parameterized complexity, but is also one of the most difficult to apply. Colour-coding relies on colouring the input with $f(k)$ colours for some f, where k is the parameter, such that at least one colouring allows the solution to be identified. Then the complexity of searching the input is reduced to searching for different coloured elements, of which there are at most $f(k)$ groups. The colouring in question may be generated randomly, with an expectation that a correct colouring will be generated after a finite number of iterations, bounded by some function of the parameter k. The colouring algorithm may be made into an exact algorithm using hashing, at a corresponding cost to the running time. This relies on the existence of a "k-perfect family of hash functions" $\mathcal{H}$, which is a set of functions mapping each element $\{1, \ldots, n\}$ to the set $\{1, \ldots, k\}$ such that for each k sized subset $S \subseteq \{1, \ldots, n\}$, there is a function $h \in \mathcal{H}$ such that h is one-to-one on S.

In fact, it turns out that not only do such families exist, it is possible to generate such a family in time $O(2^{O(k)} \log n)$ [8].

Alon *et al.* [8] give a colour-coding algorithm for the Longest Path problem. Other recent colour-coding algorithms are demonstrated by Marx [111], Scott *et al.* [147] and Bläser [16].

Although the technique provides reasonably elegant theoretical results, the cost of generating and searching the family of hash functions tends to be prohibitive.

3.5.7 Kernelization and Reduction

Kernelization is one of the most fundamental techniques of parameterized complexity, providing not only theoretical classifications, but also direct practical algorithms. Kernelizations also tend to be more easily understandable and intuitive than many other techniques, as kernelizations usually rely not on global properties, but small, local structure. Kernelizations can also provide integral parts of other

approaches, either in the form of reduction rules that can be used independently of the kernelization, or as a preprocessing step. In fact, in practice, most kernelization algorithms use only a selection of the (theoretical) reduction rules available (compare for example the rules used practically by Abu-Khzam *et al.* [4] and the range of Vertex Cover reductions available), and often interleave kernelization steps with other approaches, such as bounded search trees (for example Chen *et al.* [35] use such a process).

One of the key definitions in parameterized complexity theory, due to Downey and Fellows [59], is the following:

Definition 3.5.4 (Kernelization)**.** A kernelization for a parameterized problem $(\mathcal{P}, \kappa)$ is a polynomial time computable function that takes an instance (x, k) of $(\mathcal{P}, \kappa)$ and maps it to an instance (x', k') (the *kernel*) of $(\mathcal{P}, \kappa)$ such that $|x'| \leq g(k)$ for some computable function g, $k' \leq k$, and (x, k) is a Yes-instance if and only if (x', k') is a Yes-instance.

The mapping that kernelizations are based on may also be thought of as a polynomial time self reduction. Note that fixed-parameter tractable preprocessing that results in an instance of size bounded by a function of the parameter is not generally considered a kernelization (and certainly not a proper kernelization as defined by Abu-Khzam and Fernau [5]). Niedermeier [126] gives the following key result:

Proposition 3.5.5. *A decidable parameterized problem is fixed-parameter tractable if and only if it has a kernelization.*

Proof. Let $(\mathcal{P}, \kappa)$ be a decidable parameterized problem with instance (x, k).

($\Leftarrow$) Assume that $(\mathcal{P}, \kappa)$ has a kernelization with a kernel of size $f(k)$. Given the kernel, since the problem is decidable, we can decide whether the kernel is a Yes-instance in time bounded by some computable function g of the size of the kernel, i.e. $g(f(k)) = g'(k)$. As the kernel is obtained in polynomial time, we have an algorithm that solves the problem in time $p(n) + g'(k)$ where p is a polynomial and $|x| = n$. Therefore $(\mathcal{P}, \kappa) \in$ FPT.

($\Rightarrow$) Assume that $(\mathcal{P}, \kappa) \in$ FPT. Then there is an algorithm solving $(\mathcal{P}, \kappa)$ for every instance (x, k) in time bounded by $f(k) \cdot n^c$ where f is a computable function, $|x| = n$ and c is a constant independent of k and n. Assume that $f(k) < n$. Then in time bounded by n^{c+1} we can solve the problem and produce a constant size instance that encodes either Yes or No, depending on the solution, i.e. a kernelization. If

$f(k) \geq n$ then the size of the instance is already bounded by a function of the parameter, i.e. is already a kernel. □

For example, consider the following:

> HARMONIOUS COLOURING
> *Instance:* A graph $G = (V, E)$, a positive integer k.
> *Parameter:* k.
> *Question:* Is there a k-colouring of the vertices such that each pair of colours appears on at most edge?

Given k colours, there are only $\binom{k}{2}$ pairs of colours, thus if there are more than $\binom{k}{2}$ edges, there must be at least one repeated pair. Thus (assuming there are no isolated vertices), there can be at most $2\binom{k}{2}$ vertices in the graph. So for HARMONIOUS COLOURING there is a trivial kernelization, thus HARMONIOUS COLOURING $\in$ FPT.

Reduction Rules

Generally kernelizations are given in terms of reduction rules such as the following simple reduction for VERTEX COVER due to Buss and Goldsmith [26]. Given a vertex v of degree $k+1$, either v must be in the cover, or all of $N(v)$. As $|N(v)| > k$, the only possibility for obtaining a cover of size at most k requires that v be in the cover. Therefore v is added to the cover, and deleted from the graph, and k is reduced by 1. Then after this rule is exhaustively applied, if the instance has more than $k^2 + k$ vertices with degree at least 1, it cannot have a cover of size k or less, and is a NO-instance.

Kernelizations can be much more complex than this, particularly with regards to the rules applied to obtain the kernelization. Most rules exploit small, fixed structure to obtain a reduction. Some are parameter dependent, some are parameter independent. A rarer type of reduction rule is one that exploits variable structure, such as the so called "crown rule" for vertex cover (developed by Chor *et al.* [39], [68]).

Definition 3.5.6. Given a graph $G = (V, E)$, a crown is a tri-partition $C, H, X \subseteq V$, such that:

- C is an independent set.
- There is a matching from H into C.

- There are no edges between C and X.

It is easy to see that there exists a minimum vertex cover of G that includes all of H and none of C. Therefore we obtain an equivalent instance by removing C and H, and decreasing k by $|H|$. Note that the matching implies that $|H| \leq |C|$. This leads to a kernel with at most $3k$ vertices. This general approach has also been applied to other problems, such as Edge Disjoint Triangle Packing [116], Star Packing [136] and d-Hitting Set [3].

Kernelization by application of reduction rules is normally conducted in a somewhat ad hoc fashion, but Prieto [135] formalises the process of proving kernelization results as the 'Method of Extremal Structure'.

3.5.8 Compositional Problems and Polynomial Sized Kernels

As noted there a several techniques for demonstrating fixed-parameter tractability that do not produce practical algorithms, particularly applications of the graph minor theorem, and Courcelle's Theorem. For some problems classified via these techniques we may suspect that we cannot do much better. In particular they may not have kernels bounded by a polynomial in the parameter. Of course Proposition 3.5.5 guarantees that they have a kernelization of some form, however this may also be impractical. Bodlaender *et al.* [23] develop a tool aimed at showing that problems do not have a polynomially sized kernel, based on certain complexity theoretic assumptions. Note that the results we present here concern only OR-compositionality, so we quietly omit results concerned with AND-compositionality. For these results we refer to Bodlaender *et al.* [23, 24].

Definition 3.5.7 (Composition)**.** A *composition algorithm* for a parameterized problem $(\mathcal{P}, \kappa)$ is an algorithm that receives as input a sequence $((x_1, k), \ldots, (x_t, k))$ of instances of $(\mathcal{P}, \kappa)$ and outputs in time bounded by a polynomial in $\sum_{i=1}^{t} |x_i| + k$ an instance (x', k') where:

1. (x', k') is a Yes-instance of $(\mathcal{P}, \kappa)$ if and only if (x_i, k) is a Yes-instance of $(\mathcal{P}, \kappa)$ for some $i \in \{1, \ldots, t\}$.

2. $k' = p(k)$ where p is a polynomial.

This definition is then accompanied by the key lemma:

Lemma 3.5.8 ([23])**.** *Let $(\mathcal{P}, \kappa)$ be a parameterized problem with a composition algorithm where the non-parameterized version of the problem $\mathcal{P}$ is* NP*-complete.*

Then if $(\mathcal{P}, \kappa)$ has a polynomially sized kernel, the Polynomial Hierarchy collapses to the third level.

For details of the Polynomial Hierarchy we refer to Stockmeyer [155], and note that a collapse in the Polynomial Hierarchy seems unlikely [131].

Therefore any demonstration of the existence of a composition algorithm for a fixed-parameter tractable problem indicates that the problem is unlikely to have a polynomially sized kernel.

Bodlaender *et al.* [23] also make the following useful observation:

Lemma 3.5.9 ([23])**.** *Let $(\mathcal{P}, \kappa)$ be a parameterized graph problem, let G_1 and G_2 a pair of graphs and let k be the parameter. Suppose that $(G_1, k) \in \mathcal{P}$ or $(G_2, k) \in \mathcal{P}$ if and only if $(G_1 \uplus G_2, k) \in \mathcal{P}$. Then $(\mathcal{P}, \kappa)$ has a composition algorithm.*

$A \uplus B$ denotes the disjoint union of A and B.

This lemma then immediately shows that problems such as Longest Path, k-Cycle and k-Exact Cycle have composition algorithms, and are therefore unlikely to have polynomially sized kernels unless there is a collapse in the Polynomial Hierarchy.

Given two parameterized problems $(\mathcal{P}, \kappa)$ and $(\mathcal{P}', \kappa')$ a *polynomial time and parameter* FPT *reduction* from $(\mathcal{P}, \kappa)$ to $(\mathcal{P}', \kappa')$ is a polynomial time FPT reduction that maps an instance (x, k) of $(\mathcal{P}, \kappa)$ to an instance (x', k') of $(\mathcal{P}', \kappa')$ such that $k' \leq p(k)$, where p is a polynomial. Bodlaender *et al.* [24] show that this notion of reduction can be used to demonstrate lower bounds for kernel sizes.

Theorem 3.5.10 ([24])**.** *Let $(\mathcal{P}, \kappa)$ and $(\mathcal{P}', \kappa')$ be parameterized problems where the unparameterized problem $\mathcal{P}$ is in* NP*, $\mathcal{P}'$ is* NP*-complete and there is a polynomial time and parameter* FPT *reduction from $(\mathcal{P}, \kappa)$ to $(\mathcal{P}', \kappa')$. If $(\mathcal{P}', \kappa')$ admits a polynomially sized kernel, then $(\mathcal{P}, \kappa)$ also admits a polynomially sized kernel.*

Kratsch and Wahlström [96] use this approach to show that the Π Edge Deletion problem in unlikely to have a polynomial size kernel when parameterized by the number of deletions where Π is characterised by a finite set of forbidden induced subgraphs.

Chapter 4

Regular and Chosen Degree Graphs

4.1 Introduction

A graph G is *r-regular* if for every vertex $v \in V(G)$, we have $d(v) = r$; a graph is *regular* if it is r-regular for some $r \geq 0$.

The problem of finding a regular subgraph of a given graph has a long history. The Cubic Subgraph problem is one of Garey and Johnson's [80] original NP-complete problems, with the NP-hardness proof attributed to Chvátal. Stewart [151, 152, 153] refines this result, showing that r-Regular Subgraph is NP-complete for every $r \geq 3$, even when restricted to graphs of maximum degree 7, planar graphs of maximum degree 4 and bipartite graphs. In general results concerning similar problems are negative, with the only nontrivial polynomial results concerning spanning subgraphs [160].

From a parameterized viewpoint we may apply Proposition 3.4.3 immediately to state the following:

Proposition 4.1.1. *r-Regular Subgraph is* para-NP*-complete for parameter r.*

Therefore such a parameterization of the problem is not very interesting. However if we consider an editing version of the problem where the number of changes made to obtain the regular graph is limited we obtain more interesting parameterizations. For example, we can ask whether it is possible to obtain a 3-regular graph from an input graph by deleting at most k vertices. In order to study a wide range of such problems where the type and number of editing steps is restricted

we define a general template, the Weighted Degree Constrained Editing, or WDCE*, class of problems. We denote the editing operations vertex deletion, edge deletion and edge addition defined in Section 2.3.1 as v, e and a respectively. Then for $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$ we define WDCE$^*(S)$ as:

> WDCE$^*(S)$
> *Instance:* A graph $G = (V, E)$, two integers k and r, a weight function $\rho : V \cup E \to \{1, 2, \dots\}$, and a degree function $\delta : V \to \{0, \dots, r\}$.
> *Question:* Can we obtain from G a graph $G' = (V', E')$ using editing operations from S only, such that for all $v \in V'$ we have $\sum_{uv \in E'} \rho(uv) = \delta(v)$, with total editing cost at most k?

We write WDCE$^*_1(S)$ to indicate that the given graph is unweighted. If for all vertices v the degree constraint is $\delta(v) = \{r\}$ then we write WDCE$^r(S)$. Furthermore, we write ${}_\infty$WDCE$^*(S)$ to indicate that the editing cost is not restricted (or set to a value that exceeds the sum of weights of all vertices and edges). We omit set braces whenever the context allows, and write, for example, WDCE$^*(\mathsf{v})$ instead of WDCE$^*(\{\mathsf{v}\})$. For example the problem of deleting at most k vertices to obtain a 3-regular graph can be conveniently expressed as WDCE$^3_1(\mathsf{v})$. In Chapter 5 we will introduce a more general form of these problems, WDCE.

Chapters 4 and 5 constitute a complete classification of the complexity of WDCE(S) for $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$. In this Chapter we establish the fixed-parameter tractability of the cases WDCE$^*_1(S)$ and WDCE$^*(S)$ with $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}\}$, the $W[1]$-hardness of WDCE$^r_1(S)$ with $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$ and the polynomial time computability of WDCE$^*(\mathsf{e}, \mathsf{a})$ and the various implications. In Chapter 5 we examine the remaining cases.

4.1.1 Relationship to Existing Problems

Chvátal [80], as mentioned above, shows NP-completeness for the Cubic Subgraph = ${}_\infty$WDCE$^3_1(\mathsf{v}, \mathsf{e})$ problem. Stewart shows that the problem r-Regular Subgraph = ${}_\infty$WDCE$^r_1(\mathsf{v}, \mathsf{e})$ remains NP-complete for any $r \geq 3$, even for graphs with maximum degree 7, and planar graphs with maximum degree 4 [151, 152]. With WDCE$^r_1(\mathsf{v})$ we can express the problem of finding an induced r-regular subgraph with the largest number of vertices which includes several known NP-complete problems: Independent Set = WDCE$^0_1(\mathsf{v})$ [80], Induced Matching = WDCE$^1_1(\mathsf{v})$ [29, 156], and Induced r-Regular Subgraph = WDCE$^r_1(\mathsf{v})$ [30, 121].

The problem r-Factor $= {}_\infty\mathrm{WDCE}_1^r(\mathsf{e})$ asks whether a given graph has a *spanning* r-regular subgraph; this problem is well-known to be solvable in polynomial time as it can be reduced to the problem Perfect Matching $= {}_\infty\mathrm{WDCE}_1^1(\mathsf{e})$ as Tutte [160] shows. In fact Tutte's result holds for the more general problems f-Factor $= {}_\infty\mathrm{WDCE}_1^*(\mathsf{e})$ and Perfect b-Matching $= {}_\infty\mathrm{WDCE}^*(\mathsf{e})$ [94].

Lovász [106, 107] considers the NP-hard generalization of this problem, General Factor $= {}_\infty\mathrm{WDCE}_1(\mathsf{e})$, where each vertex is given with a list of possible degrees. By a result of Cornuéjols [43], General Factor is NP-complete, apart from certain trivial cases, if the lists may contain gaps of length > 1, such as in $\{2, 3, 6, 7\}$, but is otherwise polynomially solvable.

Moser and Thilikos [121] establish the fixed-parameter tractability of k-Almost r-Regular Graph $= \mathrm{WDCE}_1^r(\mathsf{v})$ with combined parameter $k + r$ but leave the parameterized complexity of the same problem with parameter k as an open question.

4.2 NP-Completeness and para-NP-Completeness

Although the problem r-Regular Subgraph $= {}_\infty\mathrm{WDCE}_1^r(\mathsf{v}, \mathsf{e})$ is NP-complete for $r \geq 3$, r-Regular Subgraph is in P for $r \in \{0, 1, 2\}$ [41]. Cardoso *et al.* [30] show that the Maximum r-Regular Induced Bipartite Subgraph problem is NP-complete for every $r \geq 0$. We use a variant of their reduction to show the following:

Lemma 4.2.1. $\mathrm{WDCE}_1^r(\mathsf{v})$ *is* NP*-complete for all* $r \geq 0$.

Proof. $\mathrm{WDCE}_1^0(\mathsf{v})$ (i.e. $r = 0$) corresponds to the Vertex Cover problem where the size of the vertex cover is k, which is a fundamental NP-complete problem [80]. For $r > 0$, the reduction is from $\mathrm{WDCE}_1^0(\mathsf{v})$. First we construct a bipartite r-regular graph H. Let $t \geq r$, $V(H) = V_1 \uplus V_2$ where $V_1 = \{x_1, \ldots, x_t\}$ and $V_2 = \{y_1, \ldots, y_t\}$. $E(H) = \{\, x_i y_j \mid r > j - i \mod (t),\, x_i \in V_1,\, y_j \in V_2 \,\}$.

Let (G, k) be an instance of $\mathrm{WDCE}_1^0(\mathsf{v})$ and choose $t \geq r \cdot |V(G)|$. We construct an instance (G', k') of $\mathrm{WDCE}_1^r(\mathsf{v})$ with a copy H_v of H in G' for every vertex v in $V(G)$, and with all possible edges between the vertices of every H_v and H_u if $uv \in E(G)$. Following Cardoso *et al.* we say that H_u is adjacent to H_v if $uv \in E(G)$. Let $k' = 2tk$.

Assume (G, k) is a Yes-instance of $\mathrm{WDCE}_1^0(\mathsf{v})$. Then there exists a set $S \subseteq V(G)$ of size at most k such that the deletion of S from G leaves all other vertices

isolated. For each $v \in S$ we can delete H_v from G'. This results in $2t \cdot |S| \leq 2tk = k'$ vertex deletions. G' is now r-regular as each H_u is now no longer adjacent to any other H_v, otherwise S would not have been a solution for (G, k), and the graph H was r-regular to begin with. Therefore (G', k') is a YES-instance of $\mathrm{WDCE}_1^r(\mathsf{v})$.

Assume (G', k') is a YES-instance of $\mathrm{WDCE}_1^r(\mathsf{v})$. Assume that Q is a set of vertices inducing an r-regular induced subgraph of G' and that Q' induces a connected component of Q. We may assume that $|Q|$ is maximised by the following argument. Assume that there is some H_v that is not wholly included in Q'. Some vertices of H_v must have been deleted and the remaining vertices of H_v must be adjacent to vertices of some other H_u, but then all the remaining vertices of H_u must be adjacent to all the remaining vertices of H_v by the construction. As each vertex of H_u has degree r there can be at most r vertices remaining of H_v. Q' can at most contain fragments of each H_w in G' therefore $|Q'| \leq r \cdot |V(G)|$, but by choosing H_v to remain in the graph and deleting all other fragments in Q', we have more than $r \cdot |V(G)|$ vertices as $t \geq r \cdot |V(G)|$. This new solution, as it contains more vertices, must require less deletions, therefore it is still a YES-instance of $\mathrm{WDCE}_1^r(\mathsf{v})$. Furthermore this argument shows that we can assume that H_v components are either completely deleted or remain whole. Any remaining H_v component cannot be adjacent to any other, as the graph would not be r-regular. Therefore there is a solution to (G, k) where the remaining vertices correspond to the remaining H_v components. Thus (G, k) is a YES-instance of $\mathrm{WDCE}_1^0(\mathsf{v})$. □

By subproblem containment we may also immediately claim the following:

Corollary 4.2.2. $\mathrm{WDCE}_1^*(\mathsf{v})$ *and* $\mathrm{WDCE}^*(\mathsf{v})$ *are* para-NP*-complete for parameter* r*.*

4.3 A Bounded Search Tree Algorithm for $\mathrm{WDCE}_1^r(\mathsf{v})$ and $\mathrm{WDCE}^*(\mathsf{v}, \mathsf{e})$

Apart from a kernelization, Moser and Thilikos [122] give an elegant bounded search tree algorithm for $\mathrm{WDCE}_1^r(\mathsf{v})$.

Lemma 4.3.1 ([122])**.** $\mathrm{WDCE}_1^r(\mathsf{v})$ *is fixed-parameter tractable for parameter* $k+r$*.*

Proof. Let (G, k, r) be an instance of $\mathrm{WDCE}_1^r(\mathsf{v})$. As the only operation available is vertex deletion, we may preprocess the instance by deleting any vertices with

degree less than r and reducing the parameter accordingly. Thus we may assume that all vertices have degree at least r. The algorithm then proceeds as follows:

1. If $k \geq 0$ and G is r-regular, answer YES. If $k < 0$ or $k = 0$ and G is not r-regular, answer NO.
2. Choose a vertex v with $d(v) > r$.
3. Arbitrarily pick a set $M \subseteq N(v)$ of size $r + 1$.
4. Branch on deleting one vertex of $M \cup \{v\}$, reduce k by 1.
5. Return to step 1.

The search tree has branching factor $r + 2$, and depth at most k, thus the tree has at most $\mathrm{tr}(r + 2, k) = ((r + 2)^{k+1} - 1)/(r + 1)$ vertices (see Section 3.5.2 for details regarding the size of search trees). □

In this thesis we consider several variants of the bounded search tree algorithm given above, the key difference between them being the choice of branching set at step 4. Therefore to avoid repetition and to make the presentation more compact, we present a generic version of the bounded search tree algorithm, which can then be applied with appropriate branching sets for the variants of the problems examined later.

Let G be a weighted graph with weight function $\rho : V(G) \cup E(G) \to \mathbb{N}$ and a series of constraints $f_1, f_2, \ldots, f_h$ such that $f_i : X \to 2^{\{0,\ldots,r_i\}}$ for each $i \in [h]$ where X is either the vertices or edges of G. Let $r = \max_{i\in[h]}\{r_i\}$. Let $g_1, g_2, \ldots, g_h$ be a series of functions $g_i : X \to \mathbb{N}$, where X is either the vertices or edges of G, such that g_i is associated with f_i and has the same domain for each $i \in [h]$. An element $x \in V(G) \cup E(G)$ of a graph G is *clean* if for every $i \in [h]$ we have $g_i(x) \in f_i(x)$. If every element of G is clean, then we say that G is clean.

Consider the generic problem CONSTRAINED DELETION(v, e) of rendering a weighted graph G with constraints $f_1, f_2, \ldots, f_h$, functions $g_1, g_2, \ldots, g_h$ and weight function $\rho : V(G) \cup E(G) \to \{1, \ldots, k + 1\}$ clean with at most k deletions. The following bounded search tree algorithm, an extension of the algorithm of Moser and Thilikos [122], solves the problem.

1. If G is clean and $k \geq 0$, answer YES. If $k < 0$ or G is not clean and $k = 0$, answer NO.
2. Choose an element $x \in V(G) \cup E(G)$ that is not clean. If x is an edge, let u and v be its endpoints.

3. If $g_i(x) < \min\{f_i(x)\}$ for some $i \in [h]$ and x is a vertex, delete x, reduce k by $\rho(x)$ and return to step 1. If x is an edge branch on deleting x, u or v, reduce k by $\rho(x)$, $\rho(u)$ or $\rho(v)$ as appropriate and return to step 1.

4. Arbitrarily select a set B of $r+1$ vertices from $N(x)$ if $x \in V(G)$, or $(N(u) \cup N(v)) \setminus \{u, v\}$ if $x \in E(G)$. Let E_B be the set of edges between a vertex of B and x if x is a vertex, or u and v if x is an edge.

5. Branch on the following options and reduce k as specified:

   - Delete x and reduce k by $\rho(x)$.
   - Delete a vertex $b \in B$ and reduce k by $\rho(b)$.
   - Reduce the weight of an edge $e \in E_B$ by 1 and reduce k by 1.
   - If x is an edge, delete u and reduce k by $\rho(u)$.
   - If x is an edge, delete v and reduce k by $\rho(v)$.

6. Return to step 1.

Step 3 has branching factor at most 3. In the case where the elements of E_B are adjacent to both u and v step 5 has branching factor at most $|B|+|E_B|+3 = 3r+6$. At each branching step k is reduced by at least 1, therefore the tree has depth at most k. Therefore the tree has at most $\mathrm{tr}(3r+6, k) = ((3r+6)^{k+1} - 1)/(3r+5)$ vertices. Hence we have shown the following:

Lemma 4.3.2. Constrained Deletion(v, e) *is fixed-parameter tractable with parameter* $k + r$.

If we restrict the operations available, the branching factor can be reduced. If only one of vertex deletion and edge deletion is allowed, the branching factor of step 4 can be reduced to $r + 2$. In this case, the tree has at most $\mathrm{tr}(r+2, k) = ((r+2)^{k+1} - 1)/(r+1)$ vertices.

If both vertex deletion and edge deletion are allowed, but all constraints apply only to vertices then the branching factor of step 5 is reduced to $2r + 3$ and the maximum number of vertices in the tree to $\mathrm{tr}(2r+3, k) = ((2r+3)^{k+1})/(2r+2)$.

We may now instantiate this generic algorithm by specifying the branching set chosen in step 4.

Lemma 4.3.3. WDCE$^*(\mathsf{v}, \mathsf{e})$ *is fixed-parameter tractable for parameter* $k + r$.

Proof. The branching set consists of a vertex v with $\delta(v) < d^\rho(v)$ and at most $r+1$ neighbours of v, and the edges between v and the $r + 1$ vertices. With this set

we can apply the algorithm described above to obtain a search tree with at most $\mathrm{tr}(2r+3,k) = ((2r+3)^{k+1}-1)/(2r+2)$ vertices. □

4.4 A Kernelization for WDCE$^*(\mathsf{v},\mathsf{e})$

We now present a kernelization for WDCE$^*(\mathsf{v},\mathsf{e})$. Although the kernelization is for the weighted version of the problem, it produces correct kernels for the unweighted version, thus applies also to WDCE$_1^*(S)$, WDCE$^r(S)$ and WDCE$_1^r(S)$ for $\emptyset \neq S \subseteq \{\mathsf{v},\mathsf{e}\}$, although it should be noted that if $S = \{\mathsf{e}\}$ this is less interesting, as this variant is in P (see Section 4.5). It is however interesting to note that even though the kernelization is for a more general version of the problem, the weighting allows a smaller kernel than Moser and Thilikos [122] achieve.

4.4.1 Reduction Rules

For all reduction rules considered it is obvious that they can be applied in polynomial time and one can check in polynomial time if they are applicable.

Reduction Rule 1: Let (G,k,r) be an instance of WDCE$^*(S)$. If there is a vertex v in G such that $d^\rho(v) > k+r$, then replace (G,k,r) with (G',k',r) where $G' = G - v$ and $k' = k - \rho(v)$.

Lemma 4.4.1. *Reduction Rule 1 is sound for* WDCE$^*(S)$ *with* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v},\mathsf{e}\}$.

Proof. Assume there is such a vertex v, then at least $k+1$ vertices or edges must be deleted if we do not delete v, but we may only perform at most k deletions.

Thus (G,k,r) is a YES-instance of WDCE$^*(S)$ if and only if (G',k',r) is a YES-instance of WDCE$^*(S)$. □

Consider an instance (G,k,r) of WDCE$^*(S)$. A vertex v is *clean* if $d^\rho(v) \in \delta(v)$. Extending a notion of Moser and Thilikos [121], we define a *clean region* C of G as a maximal connected subgraph of G where each vertex $v \in V(C)$ is clean. See Section 2.3.1 for the definition of a boundary.

Reduction Rule 2: Let (G,k,r) be an instance of WDCE$^*(S)$, let C be a clean region of G with empty boundary $B(C) = \emptyset$, and let $G' = G - V(C)$. Then replace (G,k,r) with (G',k,r).

Lemma 4.4.2. *Reduction Rule 2 is sound for* WDCE$^*(S)$ *with* $\emptyset \neq S \subseteq \{\mathsf{v},\mathsf{e}\}$.

Proof. As $B(C) = \emptyset$, C is not connected to the rest of the graph and therefore does not affect the solution. □

Reduction Rule 3: Let (G, k, r) be an instance of WDCE$^*(S)$. Let C be a clean region of G containing more than one vertex. Then we replace (G, k, r) with (G', k, r) where G' is obtained from G by contracting C to a single vertex v as follows:

1. Add a new vertex v.
2. For each $b \in B(C)$ we add the edge bv to G' of weight $\rho(bv) = \sum_{bc \in E(G), c \in V(C)} \rho(bc)$.
3. Set $\rho(v) = \min\{k+1, \sum_{u \in V(C)} \rho(u)\}$ and $\delta(v) = d^{\rho}(v)$ (i.e. v is clean in G').
4. Delete all vertices that belong to C.

Claim 4.4.3. *Reduction Rule 3 is sound for* WDCE$^*(S)$ *with* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}\}$.

Proof. Let (G, k, r) be an instance of WDCE$^*(S)$ and let C be a clean region of G with boundary $B(C)$. Let (G', k, r) be the new instance obtained by Reduction Rule 3, contracting C to a new vertex v as illustrated in Figure 4.1.

If we delete a vertex $u \in V(C) \cup B(C)$ or an edge incident with a vertex $u \in V(C)$, then any neighbour $u' \in V(C)$ of u will no longer be clean, and subsequently must also be deleted (edge addition is not available to make u' clean again). Clearly this cascades until the entire clean region is removed. Thus any solution for (G, k, r) either deletes all or none of the vertices in $V(C)$. Hence we can represent $V(C)$ by a single vertex v as described in Reduction Rule 3. We can limit the weight of v to $k+1$ as any weight larger than k prevents deletion. □

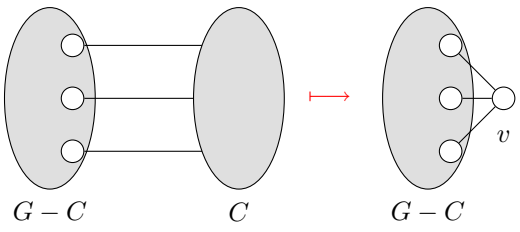


Figure 4.1: Reduction Rule 3.

4.4.2 Kernelization Lemma

We may now state the kernelization lemma. We omit the case WDCE$^*(\mathsf{e})$ as it is solvable in polynomial-time and thus admits a kernel of constant size (see Section 4.5).

Lemma 4.4.4. *Let* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}\}$. *Let* (G, k, r) *be a* YES*-instance of* WDCE$^*(S)$ *that is reduced under Reduction Rules 1, 2, and 3. Then* $|V(G)| \leq k(1 + (k + r)(1 + r)) = O(kr(k + r))$.

Proof. Let (G, k, r) be a YES-instance of WDCE$^*(S)$ with $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}\}$. We define three disjoint sets D, H and X where $D \subseteq V(G) \cup E(G)$ is the set of vertices and edges deleted from G to obtain the solution. $H \subseteq V(G)$ is the set of vertices adjacent to vertices of D or incident to edges of D. $X \subseteq V(G)$ contains all vertices that are in neither D nor H. $|D| \leq k$ by definition, and $D \subseteq V(G)$ if $S = \{\mathsf{v}\}$. Observe that H separates D from X. By definition, for all vertices $v \in H \cup X$ we have $d^{\rho}_{H \cup X}(v) \in \delta(v)$, otherwise D would not be a solution. Furthermore there are no independent clean regions in G, by Reduction Rule 2. Figure 4.2 gives an example of such a partition.

Claim 4.4.5. $|H| \leq |D| \cdot (k + r)$.

By Reduction Rule 1, for any vertex $v \in V(G)$, $d(v) \leq k + r$. In particular every element of D is adjacent to at most $k + r$ vertices in H. Moreover every vertex in H is adjacent or incident to an element of D. The claim follows directly.

Claim 4.4.6. $|X| \leq |H| \cdot r$.

All vertices in X are clean, and thus X consists entirely of clean regions. By Reduction Rule 3 all clean regions have been reduced to a single vertex each. Furthermore as $G - D$ is also clean, the vertices of H can have at most r neighbours in X.

As $|V(G)| \leq |D| + |H| + |X|$ and $|D| \leq k$, by Claims 4.4.5 and 4.4.6 we have $|V(G)| \leq k + k(k + r) + kr(k + r)$. □

Proposition 4.4.7. *Let* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}\}$. WDCE$^*(S)$ *has a kernelization with a kernel of size* $O(kr(k + r))$.

By recasting instances of WDCE$^*_1(S)$, WDCE$^r(S)$ and WDCE$^r_1(S)$ as instances of WDCE$^*(S)$ we may apply this kernelization to all these problems.

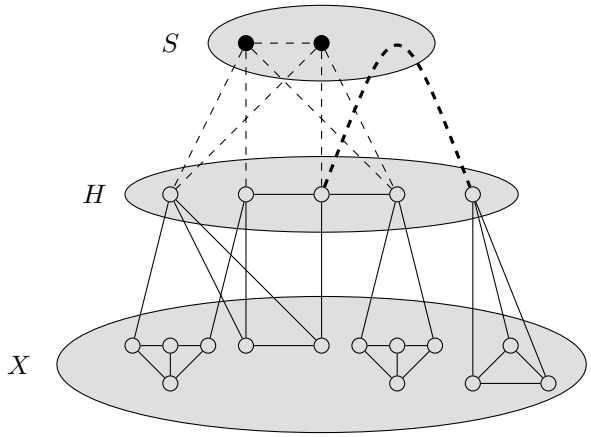


Figure 4.2: Example of the partitioning described in the proof of Lemma 4.4.4 with $r = 3$.

Corollary 4.4.8. *Let* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}\}$. $\mathrm{WDCE}_1^*(S)$, $\mathrm{WDCE}^r(S)$ *and* $\mathrm{WDCE}_1^r(S)$ *have kernelizations with kernels of size* $O(kr(k+r))$.

Proof. The case for $\mathrm{WDCE}^r(S)$ is clear. For $\mathrm{WDCE}_1^*(S)$ and $\mathrm{WDCE}_1^r(S)$ it is clear that Reduction Rules 1 and 2 hold. For Reduction Rule 3 we note that although the definitions $\mathrm{WDCE}_1^*(S)$ and $\mathrm{WDCE}_1^r(S)$ do not allow weights greater than 1, the clean regions still behave as detailed in the rule. Thus treating the clean regions as a single element represented by a weighted vertex does not introduce incorrect solutions, nor invalidate correct ones. Once the kernelization has been performed and a solution obtained, any clean regions remaining in the graph can be restored without loss of information. □

4.5 Polynomial Time Cases

In this section we show that the problems $\mathrm{WDCE}^*(S)$ and $\mathrm{WDCE}_1^*(S)$, and hence $\mathrm{WDCE}^r(S)$ and $\mathrm{WDCE}_1^r(S)$, can be solved in polynomial time for $\emptyset \neq S \subseteq \{\mathsf{e}, \mathsf{a}\}$. If $S = \{\mathsf{e}\}$ then it is not difficult to apply standard methods of matching theory [109]. The case $S = \{\mathsf{a}\}$ can be reduced to the case $S = \{\mathsf{e}\}$ by a complementation technique which will be used in the proof of Theorem 5.5.1. However it is not immediately apparent that matching techniques can be directly applied to the case where we allow both edge addition and edge deletion. Hence we give a general construction for solving all variants of the problem with $\emptyset \neq S \subseteq \{\mathsf{e}, \mathsf{a}\}$.

Given a graph G, a *matching* is a set $E' \subseteq E(G)$ such that for every vertex $v \in V(G)$ we have $E(v) \cap E' \leq 1$. If $E(v) \cap E' = 1$ for every vertex $v \in V(G)$, then E' is a *perfect matching*. The Minimum Weight Perfect Matching problem is

defined as:

> MINIMUM WEIGHT PERFECT MATCHING
> *Instance:* A graph $G = (V, E)$ and a weight function $\rho : E(G) \to \mathbb{N}$.
> *Question:* Find a perfect matching $E' \subseteq E(G)$ such that $\sum_{e \in E'} \rho(e)$ is minimised or answer NO if no perfect matching exists.

Edmonds' blossom algorithm gives the following result:

Theorem 4.5.1 ([64, 65])**.** *The* MINIMUM WEIGHT PERFECT MATCHING *is solvable in polynomial time.*

We will use this theorem to demonstrate the following:

Theorem 4.5.2. *The problems* $\mathrm{WDCE}^*(S)$ *and* $\mathrm{WDCE}^*_1(S)$ *can be solved in polynomial time for* $\emptyset \neq S \subseteq \{\mathsf{e}, \mathsf{a}\}$.

Proof. First we consider the problem $\mathrm{WDCE}^*(\mathsf{e}, \mathsf{a})$; we will explain later how the approach can be modified for the other versions of the problem.

Let (G, k) be an instance of $\mathrm{WDCE}^*(\mathsf{e}, \mathsf{a})$ with $G = (V, E)$. For the scope of this proof it is convenient to allow edges of weight 0, so we may assume that G is a complete graph. Solving the problem is clearly equivalent to finding an edge weight function $\rho' : E(G) \to \{0, 1, 2, \dots\}$ of G such that for each $v \in V(G)$ we have $\sum_{vv' \in E(G)} \rho'(vv') = \delta(v)$ and the cost of ρ', $\sum_{vv' \in E(G)} |\rho(vv') - \rho'(vv')|$, is at most k.

We construct a graph H with edge-weight function η as follows: For each vertex v of G we introduce in H a set $V(v)$ of $\delta(v)$ vertices. For each edge $vv' \in E(G)$ we add the following vertices and edges to H.

1. We add two sets $V_{\mathrm{del}}(v, v')$ and $V_{\mathrm{del}}(v', v)$ of vertices, each of size $\rho(vv')$.
2. We add two sets $V_{\mathrm{add}}(v, v')$ and $V_{\mathrm{add}}(v', v)$ of vertices, each of size $\min\{\delta(v), \delta(v')\}$.
3. We add all edges uw for $u \in V(v)$ and $w \in V_{\mathrm{del}}(v, v') \cup V_{\mathrm{add}}(v, v')$, and all edges uw for $u \in V(v')$ and $w \in V_{\mathrm{del}}(v', v) \cup V_{\mathrm{add}}(v', v)$.
4. We add edges that form a matching $M_{vv'}$ between the sets $V_{\mathrm{del}}(v, v')$ and $V_{\mathrm{del}}(v', v)$. We will refer to these edges as *deletion edges*.
5. We add edges that form a matching $M'_{vv'}$ between the sets $V_{\mathrm{add}}(v, v')$ and $V_{\mathrm{add}}(v', v)$ and subdivide the edges of $M'_{vv'}$ twice; that is, we replace $xy \in M'_{vv'}$

by a path x, x_y, y_x, y where x_y and y_x are new vertices. We will refer to the edges of the form $x_y y_x$ as *addition edges*.

For all addition and deletion edges e in $E(H)$ we set $\eta(e) = 1$. All other edges e' have $\eta(e') = 0$.

Claim 4.5.3. *G has a solution ρ' of cost at most k if and only if H has a perfect matching of weight at most k.*

($\Rightarrow$) Let ρ' be a solution of cost $k' \leq k$. We define a perfect matching M of H as follows. Consider an edge $e = vv' \in E(G)$.

First we consider the case $\rho(e) \leq \rho'(e)$. Let $d = \rho'(e) - \rho(e)$. We match the vertices in $V_{\text{del}}(v, v')$ to $\rho(e)$ vertices in $V(v)$, and similarly, we match the vertices in $V_{\text{del}}(v', v)$ to $\rho(e)$ vertices in $V(v')$. Now we choose d vertices $x_1, \ldots, x_d$ in $V_{\text{add}}(v, v')$; let $y_1, \ldots, y_d$ be the corresponding vertices in $V_{\text{add}}(v', v)$. We match the vertices x_i to vertices in $V(v)$, the vertices y_i to vertices in $V(v')$, and the vertices $(x_i)_{(y_i)}$ and $(y_i)_{(x_i)}$ to each other, $1 \leq i \leq d$. We match the remaining vertices $x \in V_{\text{add}}(v, v') \setminus \{x_1, \ldots, x_d\}$ to x_y and the remaining vertices $y \in V_{\text{add}}(v', v) \setminus \{y_1, \ldots, y_d\}$ to y_x. Observe that the defined matching has weight d (all edges used for the matching have weight 0 except for d addition edges).

Second we consider the case $\rho'(e) \leq \rho(e)$. Let $d = \rho(e) - \rho'(e)$. We choose d vertices $x_1, \ldots, x_d$ from $V_{\text{del}}(v, v')$ and match them to the corresponding vertices $y_1, \ldots, y_d$ in $V(v', v)$. We match the remaining vertices in $V_{\text{del}}(v, v') \setminus \{x_1, \ldots, x_d\}$ to vertices in $V(v)$, and the remaining vertices in $V_{\text{del}}(v', v) \setminus \{y_1, \ldots, y_d\}$ to vertices in $V(v')$. We match all vertices $x \in V_{\text{add}}(v, v')$ to the corresponding vertices x_y, and symmetrically, all vertices $y \in V_{\text{add}}(v', v)$ to the corresponding vertices y_x. Observe that the defined matching has weight d as the only edges of weight 1 are d deletion edges.

In both cases we have defined a matching that covers exactly $\rho'(e)$ vertices of $V(v)$ and $\rho'(e)$ vertices of $V(v')$. By assumption $\sum_{vv' \in E(G)} \rho'(vv') = \delta(v)$, hence we can proceed in this way for all edges $vv' \in E(G)$, and we end up with a perfect matching M of H of weight $k' \leq k$.

($\Leftarrow$) Conversely, let M be a perfect matching of H of minimum weight $k' \leq k$.

First, we make the observation that for the sets of vertices corresponding to an edge vv', the matching M must be symmetric. That is, if a vertex x from $V_{\text{del}}(v, v')$ is matched to a vertex in $V(v)$, then the adjacent vertex y in $V_{\text{del}}(v', v)$ must be matched to a vertex in $V(v')$, as there is no other possibility if M is a perfect matching. Similarly if a vertex x in $V_{\text{add}}(v, v')$ is matched to a vertex in $V(v)$, then

its corresponding vertex y in $V_{\text{add}}(v', v)$ must be matched to a vertex in $V(v')$, and x_y must be matched to y_x, as again there is no other possibility if M is a perfect matching. Thus a perfect matching of H must correspond to some solution for G.

Second, as M is of minimum weight, there can be no edge vv' in G where the matching in H has vertices from $V_{\text{add}}(v, v')$ that are matched to vertices in $V(v)$ and vertices in $V_{\text{del}}(v, v')$ matched to vertices in $V_{\text{del}}(v, v')$. Assume for contradiction that $a \in V_{\text{add}}(v, v')$ is matched to $x \in V(v)$ and $d \in V_{\text{del}}(v, v')$ is matched to $c \in V_{\text{del}}(v', v)$. By symmetry we must have $b \in V_{\text{add}}(v', v)$ matched to $y \in V(v')$, and the vertices a_b and b_a must also be matched to each other. Then we could take the following alternate matching M' where we match d to x, c to y, a to a_b and b to b_a. Then M' is still a perfect matching, but of weight two less than M.

Then we can construct a solution ρ' of weight k. For each addition edge used in M, we increase the weight of the edge between the corresponding vertices in G by one. For each deletion edge used in M, we reduce the weight of the corresponding edge in G. As the vertices in $V(v)$ are matched, we know that $\sum_{vv' \in E(G)} \rho'(vv') = \delta(v)$.

Therefore a minimum weight perfect matching for H corresponds to a minimum weight solution for G. This completes the proof of Claim 4.5.3.

The graph H can be obtained from G in polynomial time, and we can find a minimum weight perfect matching for H in polynomial by Theorem 4.5.1. Consequently $\text{WDCE}^*(\mathsf{a}, \mathsf{e})$ can be solved in polynomial time.

For the unweighted case $\text{WDCE}_1^*(\mathsf{e}, \mathsf{a})$ we need to modify the above construction of H to ensure that $\alpha'(e) \leq 1$ holds for all $e \in E(G)$. This, however, can be accomplished by using sets $V_{\text{add}}(v, v')$, $vv' \in E(G)$, of size at most 1 only: $|V_{\text{add}}(v, v')| = 1$ if and only if $\rho(vv') = 0$. Furthermore, the above construction can be modified to deal with the (simpler) case where only edge deletion or edge addition is available: For $S = \{\mathsf{e}\}$ we remove all addition edges from H; for $S = \{\mathsf{a}\}$ we remove all deletion edges from H. It is easy to check that Claim 4.5.3 also holds for these variants of $\text{WDCE}^*(\mathsf{a}, \mathsf{e})$ with H modified as described. □

4.6 $W[1]$-Hardness of WDCE for Parameter k

4.6.1 A Useful Construction: The Fixing Gadget

Before proceeding to the main results of this section we introduce a construction that we will later use in several proofs. For any $r \geq 2$, this construction produces an

almost r-regular graph G_r, where all vertices have degree r except two with degree $r-1$. We will refer to an instance of G_r as a *fixing gadget*. The vertex set of G_r consists of the vertices x^j, y_i^j, z_i^j for $1 \leq j \leq 2$ and $1 \leq i \leq r$. The edge set consists of all edges $x^j y_i^j$ and $y_i^j z_{i'}^j$ for $1 \leq j \leq 2$ and $1 \leq i, i' \leq r$, $i \neq i'$, and all edges $z_i^1 z_i^2$ for $1 \leq i \leq r-1$.

Thus each vertex has degree r, except z_r^1 and z_r^2 which have degree $r-1$. The two vertices of degree $r-1$ will be used as *attachment points*.

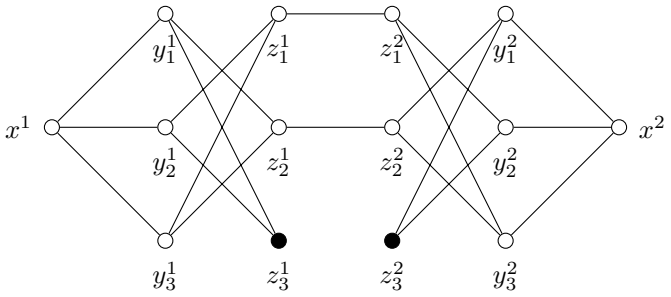


Figure 4.3: Fixing gadget for $r = 3$.

4.6.2 Preliminary Hardness Reductions

We now demonstrate $W[1]$-hardness for $\mathrm{WDCE}_1^r(S)$ where $\mathsf{v} \in S$ when the problem is parameterized by k alone. This also answers an open question posed by Moser and Thilikos [121].

Ultimately, all our reductions are from the Clique problem, parameterized by the number of vertices that form the clique, a fundamental $W[1]$-complete problem [59].

> Clique
> *Instance:* A graph $G = (V, E)$ and an integer k.
> *Question:* Does G contain a k-clique (i.e. a complete subgraph on k vertices)?

Without parameterization, Clique is NP-complete [80], and all our reductions are in fact polynomial-time FPT reductions. Therefore, all problems that we show to be $W[1]$-hard are NP-hard if considered without parameterization. This holds in particular for the next lemma; we shall use its NP-hardness part explicitly in Section 5.3.

First we show that Regular Clique (the problem Clique restricted to regular graphs) is $W[1]$-complete.

Lemma 4.6.1. *The problem* Regular Clique *is* NP*-complete and* $W[1]$*-complete for parameter* k*.*

Proof. Membership in NP of the non-parameterized version and membership in $W[1]$ for the parameterized version of the problem follow immediately as the problem is a special case of Clique. To prove hardness we devise a polynomial-time FPT reduction from Clique. Let (G, k) be an instance of Clique. We construct an instance (G', k) of Regular Clique by modifying G as follows. Let Δ be the maximum degree of G, and let $r = \Delta + (\Delta \mod 2)$. We will now demonstrate how to make the graph r-regular. We use fixing gadgets (see Section 4.6.1) to increase the degree of each vertex by attaching as many fixing gadgets as necessary by the two attachment vertices. This attachment is made between a vertex v and an instance of the fixing gadget by adding the edges between each attachment vertex and v (or perhaps only one of these edges, as below). If the degree of v is initially even, then we use an integral number of fixing gadgets; if the degree of v is initially odd, then will reach degree $r - 1$ by this method, and we will have to take another degree $r - 1$ vertex and attach one fixing gadget attachment vertex to the first, and the other attachment vertex to the second. There is always some pairing of such vertices as necessary as every graph contains an even number of vertices of odd degree, and so there is an even number of vertices requiring an odd increase of degree.

The fixing gadgets added to create G' contain no non-trivial cliques and do not introduce any new non-trivial cliques since the two attachment vertices in a fixing gadget are not adjacent. Thus G and G' have exactly the same non-trivial cliques. Clearly G' can be constructed from G in polynomial time, and since the parameter remains the same we have a polynomial-time FPT reduction. □

The reduction for several of our hardness results will be from the following variant of Regular Clique.

> Strongly Regular Multi-Coloured Clique (SRMCC)
> *Instance:* A graph $G = (V, E)$, vertex-coloured with k colours, where each vertex has exactly d neighbours in each of the k colour classes (thus G is kd-regular and each colour class has the same size)
> *Question:* Does G contain a properly coloured k-clique (i.e. a k-clique whose vertices have all different colours)?

The proof of the next lemma follows closely the $W[1]$-completeness proof of the problems Multi-Coloured Clique and Partitioned Clique in [72, 133], respectively.

Lemma 4.6.2. *The problem* SRMCC *is $W[1]$-complete for parameter k.*

Proof. Again $W[1]$ membership follows as the problem is a special case of Clique. We reduce from Regular Clique. Given an instance (G, k) of Regular Clique where G is d-regular, we let G' be the union of k vertex disjoint copies $G_1, \ldots, G_k$ of G, assigning the vertices of G_i the colour i. Then for every pair of vertices u, v in G, if uv is an edge, we add the edges $u_i v_j$, for all i, j, where a_i denotes the vertex in G_i which corresponds to vertex a in G. Clearly every vertex of G' has exactly d neighbours in each colour class, and there is a k-clique in G if and only if there is a properly coloured k-clique in G'. □

4.6.3 Main Hardness Results

We may now move to the key hardness results for this chapter.

Theorem 4.6.3. *The problem* $\mathrm{WDCE}_1^r(S)$ *is $W[1]$-hard for parameter k and $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$,*

Proof. Consider an instance (G, k) of SRMCC. Let $G = (V, E)$ be kd-regular, thus each vertex has exactly d neighbours in each colour class. We denote the set of vertices of colour i by V_i $(1 \leq i \leq k)$. Then $V = \bigcup_{i=1}^{k} V_i$ forms a partition of V. Let $|V_i| = s$ for all $1 \leq i \leq k$.

We construct an instance (G', k') of $\mathrm{WDCE}_1^r(S)$, $G' = (V', E')$, by first defining k sets V_i' $(1 \leq i \leq k)$ such that for each vertex $v \in V_i$ we add a vertex v' to V_i'. We add all possible edges between pairs of vertices in the same set V_i'. We call each of these k subgraphs induced by V_i' a (colour) *class gadget*.

For each edge uv in G where $u \in V_i$ and $v \in V_j$ with $i \neq j$, we add to G' two vertices $u'_{v'}$ and $v'_{u'}$, with the edges $u'u'_{v'}$, $u'_{v'}v'_{u'}$ and $v'_{u'}v'$. For each pair V_i' and V_j' (where $i \neq j$) of class gadgets, we denote the set of these new vertices and edges as P_{ij}. We denote by P_{ij}^i the set of all vertices $u'_{v'} \in P_{ij}$ where $u' \in V_i'$. Furthermore, for each pair of vertices u_v and $u'_{v'}$ in the same P_{ij}^i we add the edge $u_v u'_{v'}$ to P_{ij}^i if u and u' belong to the same class gadget and $u \neq u'$. We call each such P_{ij} a *connection gadget*, and each P_{ij}^i a *side* of the connection gadget. There are $\binom{k}{2}$ connection gadgets in total. Figure 4.4 gives a sketch of the structure of a connection gadget.

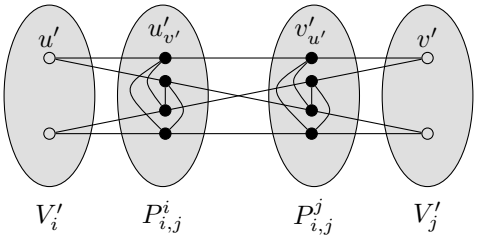


Figure 4.4: An illustration of the gadget construction in the proof of Theorem 4.6.3.

At this point we have k class gadgets corresponding to the k colour classes in the original graph, each with s vertices of degree $(s-1)+d(k-1)$, and $\binom{k}{2}$ connection gadgets corresponding to the "inter-colour-class" edges, each with $2sd$ vertices of degree $2+(s-1)d$ (sd vertices in each half). Now we choose r for the instance such that $r \geq \max\{(s-1)+d(k-1), 2+(s-1)d\}$, and $r \equiv s+1$ modulo 2 (i.e. r is of opposite parity to s). In particular we may choose the smallest r such that this is true.

Now we add for each class gadget V'_i a gadget V''_i that contains $r+1-((s-1)+d(k-1))$ vertices with s edges per vertex, such that each vertex in V''_i is adjacent to every vertex in the class gadget V'_i. We refer to V''_i as a *degree gadget*. We then add a further set of fixing gadgets as before to complete the degree of each vertex in the degree gadget to $r+1$. Note that by choosing r to have opposite parity to s, we guarantee that this is possible (if s is odd, r will be even and each vertex will require $r+1-s$ additional edges, which is even, and thus achievable; if s is even, r will be odd, then $r+1-s$ is again even, and we can complete the construction). Thus each vertex in each class gadget and degree gadget has degree one too many, but the vertices in the fixing gadgets attached to each degree gadget have the correct degree.

We similarly adjust the connection gadgets by adding two degree gadgets, each with $r+1-2+(s-1)d$ vertices, one for each side of the connection gadget. Every vertex in the degree gadget is connected to every vertex in its associated side of the connection gadget. Again we complete the degree of vertices in the degree gadgets to $r+1$ by adding fixing gadgets, and as before, by the choice of r we can guarantee that this can be done (if s is even, r is odd and $r+1-sd$ is even, if s is odd, r is even and $r+1-sd$ is even). Thus each vertex in the connection gadgets has degree $r+1$, as does each vertex in the degree gadgets. Each vertex in each fixing gadget has degree r.

We can construct G' from G in polynomial time, as we are adding only $(4r+$

$3)(2r+2s-s-sd-dk+1)$ vertices, where $r, s, d \leq n$. Thus, by the following claim we have a polynomial-time FPT reduction from SRMCC to $\mathrm{WDCE}_1^r(S)$, and the result follows.

We set $k' = k + 2\binom{k}{2}$.

Claim 4.6.4. *The following statements are equivalent:*

1. (G, k) *is a* YES*-instance of* SRMCC.
2. (G', k') *is a* YES*-instance of* $\mathrm{WDCE}_1^r(\mathsf{v})$.
3. (G', k') *is a* YES*-instance of* $\mathrm{WDCE}_1^r(\mathsf{v}, \mathsf{e})$.
4. (G', k') *is a* YES*-instance of* $\mathrm{WDCE}_1^r(\mathsf{v}, \mathsf{e}, \mathsf{a})$.

$(1 \Rightarrow 2)$ Assume that (G, k) is a YES-instance of SRMCC. Then there exist k vertices $v_1, \ldots, v_k$, one from each colour class, that form a properly coloured clique. Assume without loss of generality that $v_i \in V_i$. Then we can delete from G' the corresponding vertices v_i' from V_i', and the pairs of vertices $(v_i')_{v_j'}$ and $(v_j')_{v_i'}$ from P_{ij} that correspond to the edges in the clique. Then each remaining vertex in each class gadget has had precisely one incident edge removed from it, as have the vertices in each degree gadget associated with the class gadget. So the components corresponding to the colour classes and their immediate extension are now r-regular. Similarly each vertex in every connection gadget and their associated connection gadgets has had exactly one incident edge removed, either by the vertex removed from the connection gadget, or from the parent vertex in the class gadget (but never both). Now each vertex in these gadgets has degree precisely r. We have chosen one vertex from each V_i', and two vertices from each P_{ij}, giving a total of $k' = k + 2\binom{k}{2}$ vertices, thus (G', k') is also a YES-instance of $\mathrm{WDCE}_1^r(\mathsf{v})$.

$(2 \Rightarrow 3,\ 3 \Rightarrow 4)$ These implications are trivial.

$(4 \Rightarrow 1)$ Assume that (G', k') is a YES-instance of $\mathrm{WDCE}_1^r(\mathsf{v}, \mathsf{e}, \mathsf{a})$. Then there are $k' = k + 2\binom{k}{2}$ deletions that can be made to make G' r-regular. Obviously we cannot delete any vertices from the fixing gadgets in the graph. Further we cannot delete any vertices from the degree gadgets, as this would reduce the degree of their attached fixing gadgets. Thus the deleted vertices must come from class and connection gadgets. However, there must be precisely one vertex from each such component: if no vertex is deleted then the degree of some vertex in that component will remain $r+1$; if more than one vertex is deleted, then the degree of some vertices in the component will drop below r. Also note that for each vertex

u_v deleted from one side of a connection gadget, the vertex deleted in the other side must be the vertex v_u. If it were not, then at least one vertex in each side would have degree $r-1$. Also, the vertex deleted from each side of each connection gadget must be attached to the vertex deleted from the adjacent class gadget, otherwise the vertices attached to vertices deleted from the class gadget will have degree at most $r-1$. Thus we can see that if (G', k') is a YES-instance, the set of vertices to be deleted is very precise and restricted. In fact, if we are to use only the allotted budget of $k+2\binom{k}{2}$, we must choose precisely one vertex from each class gadget, and two vertices from each connection gadget, where the vertices from the connection gadget component are connected to the vertices deleted from the two class gadgets it is associated with. An edge deletion can only reduce the degree of two vertices, leaving us with too many edges to delete, or vertices of degree less than r. Similarly, the tight budget implies that no edge addition can be used either. Thus (G', k') is a YES-instance of $\mathrm{WDCE}_1^r(\mathsf{v})$. Furthermore, the vertices $v_1, \ldots, v_k$ corresponding to the k vertices deleted from the class gadgets induce a properly coloured clique in G; the edges of the clique correspond to the $2\binom{k}{2}$ vertices deleted from the connection gadgets. Hence (G, k) is a YES-instance of SRMCC. □

Corollary 4.6.5. $\mathrm{WDCE}_1^*(S)$ *is* $W[1]$*-hard for parameter* k *and* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$.

As the proof of Theorem 4.6.3 ensures that no edge addition is used in the reduction, the $W[1]$-hardness results hold also for the weighted versions of the problems.

Corollary 4.6.6. *The problems* $\mathrm{WDCE}^r(S)$ *and* $\mathrm{WDCE}^*(S)$ *are* $W[1]$*-hard for parameter* k *and* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$.

Let $\mathrm{WDCE}^=(S)$ denote the variant of $\mathrm{WDCE}^r(S)$ where we ask for a regular graph of unspecified regularity (i.e. when r is not given).

Corollary 4.6.7. *The problems* $\mathrm{WDCE}_1^=(S)$ *and* $\mathrm{WDCE}^=(S)$ *are* $W[1]$*-hard for parameter* k *and* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$.

Proof. We reduce from $\mathrm{WDCE}_1^r(S)$ and $\mathrm{WDCE}^r(S)$, respectively. Given an instance (G, k) of one of these problems, we construct a new instance (G', k) by adding one r-regular connected component with more than k vertices. This can be done by taking, for example, k fixing gadgets and connecting them in a ring. We clearly cannot alter this component within the budget k, thus the only possible solution is the same as that for (G, k). Thus if (G', k) is a YES-instance of $\mathrm{WDCE}_1^=(S)$ or

$\text{WDCE}^{=}(S)$, then it is also a YES-instance of $\text{WDCE}_1^r(S)$ or $\text{WDCE}^r(S)$, respectively. The converse direction holds trivially. □

4.7 Conclusion

In this chapter we have seen the key results for the WDCE^* class of problems.

We have demonstrated a polynomially sized kernelization for $\text{WDCE}^*(\mathsf{v}, \mathsf{e})$ parameterized by $k+r$ that extends to $\text{WDCE}_1^r(\mathsf{v}) =$ k-ALMOST r-REGULAR GRAPH $=$ MAXIMUM INDUCED r-REGULAR SUBGRAPH, and thus improves Moser and Thilikos's [122] kernelization.

We have given a polynomial time algorithm for $\text{WDCE}^*(\mathsf{v}, \mathsf{e})$ which complements previously known results for r-FACTOR $= {}_\infty\text{WDCE}_1^r(\mathsf{e})$, PERFECT MATCHING $= {}_\infty\text{WDCE}_1^1(\mathsf{e})$, f-FACTOR $= {}_\infty\text{WDCE}_1^*(\mathsf{e})$ and PERFECT b-MATCHING $= {}_\infty\text{WDCE}^*(\mathsf{e})$.

We have also answered an open question of Moser and Thilikos [121], and shown that $\text{WDCE}_1^r(\mathsf{v})$ is $W[1]$-hard for parameter k. Subsequently any variant of the problem which contains this as a subproblem is also $W[1]$-hard.

In Chapter 5 we will continue the generalisation of these results, moving to variants that resemble Lovász's GENERAL FACTOR problem, and also examining the case where both vertex deletion and edge addition are allowed.

Chapter 5

General Factors of Graphs

5.1 Introduction

In this chapter we will generalise the WDCE* class of problems, and classify the complexity of the cases not already covered by the WDCE* cases. The cases examined in this chapter generally relate to the General Factor $= {}_\infty\mathrm{WDCE}_1(\mathsf{e})$ problem, introduced by Lovász [106, 107] and studied by many others [10, 29, 43, 108, 109, 148, 161, 162].

We define the generalisation of WDCE* as WDCE.

> WDCE(S)
> *Instance:* A graph $G = (V, E)$, two integers k and r, a weight function $\rho : V \cup E \to \{1, 2, \dots\}$, and a *degree list function* $\delta : V \to 2^{\{0,\dots,r\}}$.
> *Question:* Can we obtain from G a graph $G' = (V', E')$ using editing operations from S only, such that for all $v \in V'$ we have $\sum_{uv \in E'} \rho(uv) \in \delta(v)$, with total editing cost at most k?

By a result due to Cornuéjols [43], General Factor is NP-complete, apart from certain trivial cases, if the lists may contain gaps of length > 1, but is otherwise polynomially solvable. This subsumes the polynomial time results for r-Factor $= {}_\infty\mathrm{WDCE}_1^r(\mathsf{e})$ (which can be reduced to Perfect Matching $= {}_\infty\mathrm{WDCE}_1^1(\mathsf{e})$ as shown by Tutte [160]), f-Factor $= {}_\infty\mathrm{WDCE}_1^*(\mathsf{e})$ [160] and Perfect b-Matching $= {}_\infty\mathrm{WDCE}^*(\mathsf{e})$ [66, 94, 150, 163].

We also examine the cases where vertex deletion and edge addition are both allowed with parameter $k + r$. It appears that this combination of editing operations increases the complexity significantly, and we show that although the problem is still fixed-parameter tractable there is unlikely to be a polynomial time kernelization of

a form similar to that of Lemma 4.4.4, unless P = NP. Moreover we show that in the general case this combination of editing operations is unlikely to allow a polynomially sized kernel by application of Lemma 3.5.9. However by application of Stewart's [154] extension (Corollary 3.5.3) of the logical meta-theorem (Section 3.5.1) due to Frick and Grohe [79], we show that WDCE(S) with $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$ is fixed parameter tractable for parameter $k + r$.

5.1.1 A Note on Some Immediate Results

Chapter 4 gives some immediate results that we may apply to the more general problems in this chapter.

Bound Search Tree Algorithm

The bounded search tree approach from Section 4.3 can be applied essentially immediately to WDCE(S) where $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}\}$.

Lemma 5.1.1. WDCE(v, e) *is fixed-parameter tractable for parameter* $k + r$.

Proof. The branching set consists of a vertex v with $d^{\rho}(v) \notin \delta(v)$, at most $r + 1$ neighbours of v and the edges between v and the $r + 1$ neighbours. Therefore the branching set has size at most $2r + 3$. By instantiating the generic algorithm described in Section 4.3, we obtain a search tree with at most $\mathrm{tr}(2r + 3, k) = ((2r + 3)^{k+1} - 1)/(2r + 2)$ vertices. □

If we only allow vertex deletion, then the branching set consists only of vertices, and is of size at most $r + 2$, giving a search tree with at most $\mathrm{tr}(r + 2, k) = ((r + 2)^{k+1} - 1)/(r + 1)$ vertices.

Corollary 5.1.2. WDCE(v) *is fixed-parameter tractable for parameter* $k + r$.

Hardness

Theorem 4.6.3 established the $W[1]$-hardness of WDCE$^*(S)$, WDCE$^r(S)$ and WDCE$^*_1(S)$ for parameter k with $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$ for parameter k. As these are all subproblems of the more general form, we may immediately conclude the following:

Theorem 5.1.3. *The problems* WDCE$_1(S)$ *and* WDCE(S) *with* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$ *are* $W[1]$*-hard for parameter* k.

Similarly the para-NP-completeness results of Corollary 4.2.2 carry over immediately.

Proposition 5.1.4. $\mathrm{WDCE}_1(\mathsf{v})$ *and* $\mathrm{WDCE}(\mathsf{v})$ *are* para-NP*-complete for parameter* r*.*

5.2 Kernelizations for WDCE(v), WDCE(v, e) and WDCE(e)

5.2.1 Reduction Rules

Consider an instance (G, k, r) of $\mathrm{WDCE}(S)$ with $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}\}$.

Observe that Reduction Rules 1 and 2 from Section 4.4.1 still apply in this case. However Reduction Rule 3 is no longer valid. We introduce the following new rule.

We extend the definition of clean vertices and regions to the case where vertices have degree list functions. A vertex v is clean if $d^{\rho}(v) \in \delta(v)$. A clean region C of graph G is a maximal connected subgraph of G such that every vertex $v \in C$ is clean. Let C be a clean region of G with boundary $B(C)$. Let the *i-th layer* of C be the subset $C_i = \{\, c \in V(C) \mid \min_{b \in B(C)}\{d_G(c, b)\} = i \,\}$ where $d_G(c, b)$ denotes is the distance between c and b in G. Note that all the neighbours of a vertex of layer C_i belong to $C_{i-1} \cup C_i \cup C_{i+1}$.

Reduction Rule 4: Let (G, k, r) be an instance of $\mathrm{WDCE}(S)$ and let C be a clean region of G such that $C_{k+2} \neq \emptyset$. Then replace (G, k, r) with (G', k, r) as follows:

1. For each vertex $u \in C_{k+1}$ reduce each entry of $\delta(u)$ by $d^{\rho}_{C_{k+2}}(u)$.
2. Delete all layers C_i for $i \geq k + 2$.

Lemma 5.2.1. *Reduction Rule* 4 *is sound for* $\mathrm{WDCE}(S)$ *with* $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}\}$*.*

Proof. Let D be the set of vertices and edges deleted in a minimal solution of (G, k, r); thus $D \subseteq V(G)$ if $S = \{\mathsf{v}\}$ and $D \subseteq V(G) \cup E(G)$ if $S = \{\mathsf{v}, \mathsf{e}\}$. Let $G(D)$ be the subgraph of G induced by all vertices that belong to D or are incident to an edge in D. Each connected component X of $G(D)$ that contains a vertex of a clean region C must also contain a vertex of the boundary $B(C)$ of C, since otherwise we could obtain a solution that is smaller than D by removing those vertices and edges from D that induce X in $G(D)$. Consequently, each vertex $v \in D \cap V(C)$ must be of distance at most $|D|$ from a vertex in the boundary of C, i.e. $v \in C_i$ for $i \leq |D|$. Similarly, each endpoint of an edge $e \in D \cap E(C)$ must belong to some C_i for $i \leq |D| + 1$. If $|D| \leq k$ then D is also a solution of (G', k, r) as D does not

touch the part deleted from G. On the other hand, each solution D' of (G', k, r) is trivially a solution of (G, k, r). □

5.2.2 Kernelization Lemmas

Now we can state our kernelization lemmas.

Lemma 5.2.2. *Let* $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}\}$. *Let* (G, k, r) *be a* YES*-instance of* WDCE(S) *that is reduced under Reduction Rules* 1, 2 *and* 4. *Then* $|V(G)| \leq k(1 + (k + r)(1 + r^{k+1})) = O(k^2 r^{k+1} + k r^{k+2})$.

Proof. Assume that (G, k, r) is a YES-instance of WDCE(S) and is reduced under Reduction Rules 1, 2 and 4. Similarly to the proof of Lemma 4.4.4, we define three disjoint sets D, H, and X where $D \subseteq E(G) \cup V(G)$ is the set of vertices and edges whose deletion provides a solution, $H \subseteq V(G)$ is the set of vertices adjacent to vertices in D or incident to edges in D, and $X \subseteq V(G)$ contains the remaining vertices of G that are neither in D nor in H. $|D| \leq k$ by definition, and $D \subseteq V(G)$ if $S = \{\mathsf{v}\}$. Observe that H separates D from X. Furthermore, observe that there are no independent clean regions in G (otherwise the graph would not be reduced under Reduction Rule 2), and $d^{\rho}(x) \in \delta(x)$ for all $x \in X$ (otherwise D would not be a solution).

Claim 5.2.3. $|H| \leq |D| \cdot (k + r)$.

Since the instance is reduced under Reduction Rule 1, the maximum weighted degree of every vertex is at most $k + r$, and each vertex in H is adjacent or incident to some element of D; hence the claim follows.

Claim 5.2.4. $|X| \leq |H| \cdot r^{k+1}$.

As all vertices in X are clean, all boundary vertices for all clean regions are contained in $H \cup D$. Then there is no vertex in X of distance greater than $k + 1$ from H, otherwise the graph is not reduced under Reduction Rule 4. Thus X is the disjoint union of the sets $X_1, \ldots, X_{k+1}$ where $X_i = \{\, x \in X \mid \min_{h \in H}\{d_G(x, h)\} = i \,\}$. We have $|X_1| \leq |H| \cdot r$ as vertices in H may have all their neighbours in X. For $i > 1$ we have $|X_i| \leq |X_{i-1}|(r - 1) \leq |X_1|(r - 1)^{i-1}$ as each vertex in X_i has at least one neighbour in X_{i-1} and at most r in neighbours in total. For $r \geq 1$ we assume inductively that $\sum_{i=1}^{m} |X_i| \leq |X_1| \cdot r^{m-1}$. Then $\sum_{i=1}^{m+1} |X_i| \leq |X_1| \cdot r^{m-1} + |X_1| \cdot (r - 1)^{m-1}(r - 1) \leq |X_1| \cdot r^m$. Therefore with $k + 1$ layers, $|X| \leq |H| \cdot r^{k+1}$. For $r = 0$, H is an independent set after removal of D, therefore $X = \emptyset$, and the claim holds trivially.

Since $|V(G)| \leq |D| + |H| + |X|$ and $|D| \leq k$, the lemma follows from Claims 5.2.3 and 5.2.4. □

If we restrict the allowed editing operations to edge deletion alone, then we can obtain a better kernel by observing that if the graph contains any vertex of degree greater than $k+r$, then it is a NO-instance, as this vertex cannot be fixed with only k edge deletions. Reduction Rules 2 and 3 still apply. Thus WDCE(e) allows an improved kernelization lemma.

Lemma 5.2.5. *Let (G, k, r) be a* YES*-instance of* WDCE(e) *that is reduced under Reduction Rules 2 and 4. Then $|V(G)| \leq 2k(1 + r^{k+1}) = O(kr^{k+1})$.*

Proof. We define D, H, and X as in the proof of Lemma 5.2.2. Since D consists only of edges and is of size at most k, we have $|H| \leq 2k$. The rest of the proof follows as for Lemma 5.2.2. □

We combine the results of Lemmas 4.4.4, 5.2.2 and 5.2.5 into the following theorem.

Theorem 5.2.6. *For $\{\mathsf{v}\} \subseteq S \subseteq \{\mathsf{v}, \mathsf{e}\}$, the problem* WDCE($S$) *admits a kernel with $O(k^2 r^{k+1} + kr^{k+2})$ vertices, and the problem* WDCE*(S) *admits a kernel with $O(kr(k+r))$ vertices. The problem* WDCE(e) *admits a kernel with $O(kr^{k+1})$ vertices.*

5.3 Kernelization and Edge Addition

It appears that the kernelization approach in Section 4.4 does not work under the presence of edge addition. If we apply our approach to a variant of WDCE that includes WDCE*(v, a) as a subproblem, it becomes necessary to consider deleting a set of vertices from clean regions so that the edges that become available may be used to complete the degree of a vertex of insufficient degree. This gives rise to the following subproblem:

> EDGE REPLACEMENT SET
> *Instance:* A graph $G = (V, E)$, two positive integers k and t.
> *Question:* Does there exist a set $X \subseteq V$ such that $|X| \leq k$ and there are exactly t edges between vertices in X and vertices in $V \setminus X$?

Unfortunately, EDGE REPLACEMENT SET is NP-complete, thus making the possibility of obtaining a kernel in polynomial time by somehow identifying all relevant sets in the clean regions unlikely.

Proposition 5.3.1. Edge Replacement Set *is* NP*-complete and* $W[1]$*-hard for parameter* k*.*

Proof. We give a polynomial-time FPT reduction from Regular Clique (see Lemma 4.6.1).

Let (G, k) be an instance of Regular Clique where $G = (V, E)$ is r-regular. We may assume that $r > k^2$ since we can use the fixing gadgets (see Section 4.6.1) to increase the degree of each vertex arbitrarily without introducing non-trivial cliques. For a set $X \subseteq V$ let $d(X)$ denote the number of edges $uv \in E$ with $u \in X$ and $v \in V \setminus X$. If X forms a k-clique in G then $d(X) = k(r - k + 1)$. Therefore we put $t = k(r - k + 1)$ and consider (G, k, t) as an instance of Edge Replacement Set.

Let $X \subseteq V$ with $|X| \leq k$ and $d(X) = t$. We show that X has exactly k elements and forms a clique in G. Assume for the sake of contradiction that $|X| < k$. It follows that $d(X) \leq |X|r \leq r(k-1) < rk - k^2 \leq t$, a contradiction; hence $|X| = k$. Each vertex $x \in X$ has at most $k-1$ neighbours in X and at least $r-k+1$ neighbours in $V \setminus X$. Therefore, if at least one $x \in X$ had fewer than $k - 1$ neighbours in X, then $d(X) > k(r - k + 1) = t$, again a contradiction. Hence X indeed induces a k-clique in G.

Membership in NP is clear, establishing the completeness requirement of the proposition. □

Thus this proof demonstrates that a polynomial time kernelization which relies upon identifying such candidate sets for deletion is unlikely to exist unless P = NP. Note also that the proof holds if we also demand that the set X is connected.

In fact we go further and note that Lemma 3.5.9 applies to Regular Clique. Therefore Regular Clique has a composition algorithm, and thus does not have a polynomially sized kernel unless the Polynomial Hierarchy collapses to the third level. As the reduction for Proposition 5.3.1 is a polynomial time and parameter FPT reduction, by Theorem 3.5.10 we can infer the following:

Lemma 5.3.2. Edge Replacement Set *has no kernel of size polynomial in* k *and* r *unless the Polynomial Hierarchy collapses to the third level.*

This can be extended further by noting that it is trivial (in the sense of a polynomial time and parameter FPT reduction) to produce an instance of $\text{WDCE}^*(\mathsf{v}, \mathsf{a})$ where a deletion in the style of Edge Replacement Set is required.

Lemma 5.3.3. *For* $\{\mathsf{v},\mathsf{a}\} \subseteq S \subseteq \{\mathsf{v},\mathsf{e},\mathsf{a}\}$, $\mathrm{WDCE}^*(S)$, *and hence* $\mathrm{WDCE}(S)$, *has no kernel of size polynomial in* k *and* r *unless the Polynomial Hierarchy collapses to the third level.*

Proof. We will show that there is a polynomial time and parameter FPT reduction from Edge Replacement Set to $\mathrm{WDCE}^*(S)$ where $\{\mathsf{v},\mathsf{a}\} \subseteq S \subseteq \{\mathsf{v},\mathsf{e},\mathsf{a}\}$. The non-parameterized $\mathrm{WDCE}^*(S)$ is NP-complete by Theorem 4.6.3 and the non-parameterized Edge Replacement Set is clearly in NP. Hence we can use Theorem 3.5.10 to extend Lemma 5.3.2 to $\mathrm{WDCE}^*(S)$.

Let (G,k,t) be an instance of Edge Replacement Set. By the proof of Proposition 5.3.1 we may assume that $t = k(r-k+1)$ and that G is r-regular for some $r > k^2$, and therefore that the instance is a Yes-instance if and only if G contains a k-clique. Furthermore we assume that $k \geq 3$. We construct an instance of (G',k',r') of $\mathrm{WDCE}^*(S)$ with $\{\mathsf{v},\mathsf{a}\} \subseteq S \subseteq \{\mathsf{v},\mathsf{e},\mathsf{a}\}$ where G' is G with an additional vertex v added. We set $k' = k+t$, $\rho(v) = k+1$ and $\delta(v) = \{t\}$. For all vertices $u \in V(G') \setminus \{v\}$ we set $\delta(u) = \{r\}$ and $\rho(u) = 1$. We choose r' to be t.

Assume (G,k,t) is a Yes-instance of Edge Replacement Set. Then there is a set X of vertices of size at most k such that $\sum_{x\in X} d_{V(G)\setminus X}(x) = t$. Let $X' \subseteq V(G) \setminus X$ be the set of endpoints of edges with one endpoint in X. Deleting the vertices of X from G' and suitably adding t edges between v and the vertices of X' satisfies all degree constraints of vertices remaining after editing. The cost of the deletions and additions is at most $t+k = k'$. Therefore (G',k',r') is a Yes-instance of $\mathrm{WDCE}^*(S)$ with $\{\mathsf{v},\mathsf{a}\} \subseteq S \subseteq \{\mathsf{v},\mathsf{e},\mathsf{a}\}$.

Assume (G',k',r') is a Yes-instance of $\mathrm{WDCE}^*(S)$ with $\{\mathsf{v},\mathsf{a}\} \subseteq S \subseteq \{\mathsf{v},\mathsf{e},\mathsf{a}\}$. As the weight of v is $k+1$ it cannot have been deleted, so there must be a set of added edges of total weight t incident on v. Let k_e be number of edges deleted. We first assume that $k_e = 0$ (and will later show that $k_e > 0$ is not possible). By the construction $G' - v$ is clean. We also have that $t = k(r-k+1)$ and $r > k^2$. By the same argument as in the proof of Proposition 5.3.1, the only possibility is that there is a k-clique in $G' - v$ which can be deleted to allow t edges to be added. This would give a total cost of $t + k = k'$. Assume that $k_e > 0$, then we may only delete at most $k - k_e$ vertices, which would give at most $r(k-k_e) + 2k_e = rk - (r-2)k_e \leq rk - (r-2) < rk - k^2 + 2 \leq t$ edges to add and we derive a contradiction. Thus $k_e = 0$. Therefore (G,k,t) is a Yes-instance of Edge Replacement Set.

The reduction is clearly a polynomial time and parameter FPT reduction. □

Extending this result to $\mathrm{WDCE}_1^r(\mathsf{v},\mathsf{a})$ seems more difficult, as it is not clear that Edge Replacement Set is a subproblem. Conversely however there does not seem to be a good kernelization for $\mathrm{WDCE}_1^r(\mathsf{v},\mathsf{a})$, but perhaps some progress can be made here with new structural insights.

5.4 General Fixed-Parameter Tractability for WDCE

In this section we establish the fixed-parameter tractability of WDCE for parameter $k+r$ where we allow edge addition:

Theorem 5.4.1. *The problems* $\mathrm{WDCE}(S)$ *and* $\mathrm{WDCE}_1(S)$ *are fixed-parameter tractable for parameter* $k+r$ *and* $\emptyset \neq S \subseteq \{\mathsf{v},\mathsf{e},\mathsf{a}\}$.

For establishing Theorem 5.4.1 we apply the meta-theorem of Frick and Grohe [79] (see Section 3.5.1). In particular we use Corollary 3.5.3, due to Stewart [154], which states that that the parameterized model checking problem for first-order logic is fixed-parameter tractable for relational structures of bounded degree where the degree bound depends on the parameter. In fact Stewart [154] indicated how this can be used to show that r-Regular Subgraph $= \mathrm{WDCE}_1^r(\mathsf{v},\mathsf{e})$ with parameter $k+r$ is fixed-parameter tractable. In the following we extend this approach to $\mathrm{WDCE}(S)$ and $\mathrm{WDCE}_1(S)$ for $S \subseteq \{\mathsf{v},\mathsf{e},\mathsf{a}\}$.

We must now ensure that the maximum weighted degree of the graph is bounded by a function of the parameter, which is not guaranteed by the problem. However if the graph has a vertex v such that $d^\rho(v) > k+r$ then, if there is a solution of cost at most k, v must be part of the solution, i.e. v must be deleted (i.e. by Reduction Rule 1, from Section 4.4). Thus we can eliminate all such vertices in polynomial time and decrease the parameter k accordingly. Further, we can assume that all vertices are of weight at most $k+1$ and all edges are of weight at most $r+k+1$ (this or any higher weight renders deletion infeasible). Hence we can assume that the weighted degree and all edge and vertex weights are bounded in terms of the parameters k and r.

We associate with a weighted graph G its incidence structure S_G as described in Section 2.3.3. To the vocabulary of the structure we add the unary relations W_i, $1 \leq i \leq k+1$, and D_j, $0 \leq j \leq r$ (where $W_i x$ expresses that vertex x has weight $\rho(x) = i$, and $D_j x$ expresses that $j \in \delta(x)$). We represent an edge uv of weight w by w distinct "parallel" elements $x_1,\ldots,x_w$ with Iux_i and Ivx_i, $1 \leq i \leq w$. Note

that the maximum degree of the structure S_G (in particular the maximum degree of the Gaifman graph as defined in Section 3.5.1) is bounded in terms of k and r.

We now define the formulae $\phi_{k,r}$ expressing the problem. In the following we write $[n] = \{1, \ldots, n\}$. We define

$$\phi_{k,r} = \bigvee_{k',k'',k'''\in[k] \text{ such that } k'+k''+k'''\leq k} \exists u_1, \ldots, u_{k'}, \\ \exists e_1, \ldots, e_{k''}, \exists a_1, \ldots, a_{k'''}, \exists b_1, \ldots, b_{k'''}(\phi'_{k,r} \wedge \forall v\ \phi''_{k,r})$$

where $\phi'_{k,r}$ and $\phi''_{k,r}$ are given below. The subformula $\phi'_{k,r}$ is the conjunction of the clauses (1)–(3) and ensures that $u_1, \ldots, u_{k'}$ represent deleted vertices, $e_1, \ldots e_{k''}$ represent deleted edges, a_i, b_i, $1 \leq i \leq k'''$ represent end points of added edges, and the total editing cost is at most k. Note that since added edges are not present in the given structure we need to express them in terms vertex pairs. For the unweighted case we must also include subformulae (4) and (5) to ensure that the addition of edges does not produce parallel edges. By restricting k', k'' or k''' to zero as appropriate we can express which editing operations are available.

(1) $\bigwedge_{i\in[k']} Vu_i \wedge \bigwedge_{i\in[k'']} Ee_i$ "u_i is a vertex, e_i is an edge;"

(2) $\bigwedge_{i\in[k''']} Va_i \wedge Vb_i \wedge a_i \neq b_i \wedge \bigwedge_{j\in[k']}(u_j \neq a_i \wedge u_j \neq b_i)$ "a_i and b_i are distinct vertices and not deleted;"

(3) $\bigvee_{w_1,\ldots,w_{k'}\in[k'] \text{ such that } \sum_{i\in[k']} w_i+k''+k'''\leq k} \bigwedge_{i\in[k']} W_{w_i}u_i$ "the weight of deleted vertices is correct;"

(4) $\bigwedge_{1\leq i<j\leq k''}(a_i \neq b_j \vee a_j \neq b_i) \wedge (a_i \neq a_j \vee b_i \neq b_j)$ "the pairs of vertices are mutually distinct;"

(5) $\bigwedge_{i\in[k''']} \forall y(\neg Ia_iy \vee \neg Ib_iy)$ "a_i and b_i are not adjacent."

The subformula $\phi''_{k,r}$ ensures that after editing each vertex v has degree $l \in \delta(v)$.

$$\phi''_{k,r} = (Vv \wedge \bigwedge_{i\in[k']} v \neq u_i) \rightarrow \bigvee_{l\in[r]} D_lv \wedge \bigvee_{\substack{l',l''\in[l] \\ l'+l''=l}} \exists x_1, \ldots, x_{l'}, y_1, \ldots, y_{l''}\ \phi'''_k,$$

where $\phi'''_{k,r}$ is the conjunction of the clauses (6)–(12).

(6) $\bigwedge_{i\in[l']} Ivx_i$ "v is incident with l' edges;"

(7) $\bigwedge_{1\leq i<j\leq l'} x_i \neq x_j$ "the edges are all different;"

(8) $\bigwedge_{i\in[l'],j\in[k'']} x_i \neq e_j$ "the edges have not been deleted;"

(9) $\bigwedge_{i\in[l'],j\in[k']} \neg I u_j x_i$ "the ends of the edges have not been deleted;"

(10) $\forall x(Ivx \to \bigvee_{i\in[l']} x = x_i \vee \bigvee_{i\in[k'']} x = e_i \vee \bigvee_i Ixu_i)$ "v is not incident with any further edges except deleted edges;"

(11) $\bigwedge_{i\in[l'']} \bigvee_{j\in[k''']}(y_i = a_j \wedge v = b_j) \vee (y_i = b_j \wedge v = a_j)$ "v is incident with at least l'' added edges;"

(12) $\bigwedge_{j\in[l'']}(v = a_j \to \bigvee_i y_i = b_j) \wedge (v = b_j \to \bigvee_{j\in[l'']} y_i = a_j)$ "v is incident with at most l'' added edges."

The formulae $\phi_{k,r}$ give us the following results:

Lemma 5.4.2. *For all $k, r \geq 0$ there exists a first order formula $\phi_{k,r}$ such that for every instance (G, k, r) of* WDCE(S) *(resp.* WDCE$_1(S)$*) with $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$ and associated incidence structure S_G, we have S_G is a model of $\phi_{k,r}$ if and only if (G, k, r) is a* Yes*-instance of* WDCE(S) *(resp.* WDCE$_1(S)$*).*

Then by Corollary 3.5.3, Lemma 5.4.2 proves Theorem 5.4.1.

5.5 $W[1]$-Hardness for WDCE$_1(\mathsf{e})$, WDCE$_1(\mathsf{a})$ and WDCE$_1(\mathsf{e}, \mathsf{a})$

We now show $W[1]$-hardness for the remaining cases where the only operations are edge deletion and edge addition.

Theorem 5.5.1. *The problem* WDCE$_1(S)$ *is $W[1]$-hard for parameter k and $\emptyset \neq S \subseteq \{\mathsf{e}, \mathsf{a}\}$.*

Proof. Let (G, k) be an instance of Regular Clique where G is r-regular and has n vertices. We construct a graph H by adding to G additional vertices $v_1, \ldots, v_k$ and all edges uv_i for $u \in V(G)$ and $1 \leq i \leq k$. We set $\delta(v_i) = \{n-k\}$, $1 \leq i \leq k$, and $\delta(u) = \{r-k+1, r+k\}$ for all $u \in V(G)$. We set $k' = \binom{k}{2} + k^2$. Furthermore let $\bar{H}$ be the complement graph of H (i.e. $V(\bar{H}) = V(H)$ and $E(\bar{H}) = \{\, uv \mid u, v \in V(H)$, $u \neq v$, and $uv \notin E(H)\,\}$) and label the vertices v of $\bar{H}$ with $\bar{\delta}(v) = \{\, |V(H)| - 1 - d \mid d \in \delta(v)\,\}$.

The theorem follows from Lemma 4.6.1 and the following claim.

Claim 5.5.2. *The following statements are equivalent.*

1. *(G, k) is a* Yes*-instance of* Regular Clique*.*

2. (H, k') *is a* YES*-instance of* $\mathrm{WDCE}_1(\mathsf{e})$.

3. (H, k') *is a* YES*-instance of* $\mathrm{WDCE}_1(\mathsf{e}, \mathsf{a})$.

4. $(\bar{H}, k)$ *is a* YES*-instance of* $\mathrm{WDCE}_1(\mathsf{a})$.

$(1 \Rightarrow 2)$ Assume there is a k-clique in G. Let the set of vertices of the clique be $C = \{u_1, \ldots, u_k\}$. Then we can satisfy the degree requirement of H by deleting the edges $u_i u_j$, where $1 \leq i, j \leq k$, and the edges $v_i u_j$, where $1 \leq i, j \leq k$. The number of edges deleted is exactly k'.

$(2 \Rightarrow 3)$ This implication is trivial.

$(3 \Rightarrow 1)$ Assume that (H, k') is a YES-instance of $\mathrm{WDCE}_1(\mathsf{e}, \mathsf{a})$ and fix a solution of total cost at most k'. For $i \in \{r - k + 1, r + k\}$ let $V_i \subseteq V(G)$ be the set of vertices that have degree i after editing. Let D be the set of edges that are deleted in the solution and are incident with a vertex v_i for $1 \leq i \leq k$. Clearly $|D| \geq k^2$. Let $D_i \subseteq D$ denote the subset of deleted edges that have an end point in V_i, $i \in \{r - k + 1, r + k\}$. Each deleted edge $e \in D$ causes the degree of a vertex $v \in V(G)$ to be decreased by one, and therefore causes additional editing operations to repair the degree of v: either by edge additions (if $v \in V_{r+k}$) or by edge deletions (if $v \in V_{r-k+1}$). At least one edge addition is required to repair the degree change caused by 2 edges from D_{r+k}. At least $\binom{k}{2} = k(k-1)/2$ edge deletions are required to repair the degree change caused by k^2 edges from D_{r-k+1}; this is exactly the case if G contains a clique on k vertices $u_1, \ldots, u_k$, and $D_{r+1} = \{\, u_i v_j \mid 1 \leq i, j \leq k \,\}$. Thus the average cost caused by an edge in D_{r+k} is at least $1/2$, whereas the average cost caused by an edge in D_{r-k+1} is at least $k(k-1)/(2k^2) = (k-1)/(2k)$, which is smaller than $1/2$. Consequently, the editing cost is smallest if $D_{r+k} = \emptyset$ and $D_{r-k+1} = \{\, u_i v_j \mid 1 \leq i, j \leq k \,\}$ for vertices $u_1, \ldots, u_k$ that form a clique in G. However, in this case the total editing cost is exactly $|D_{r-k+1}| + |D_{r-k+1}|(k-1)(2k) = k^2 + \binom{k}{2} = k'$, hence this case is the only one possible.

$(2 \Leftrightarrow 4)$ This equivalence follows immediately from the definition of $\bar{H}$. □

Since the main reduction in the proof of Theorem 5.5.1 does not use edge addition it also applies to the weighted version of the problems:

Corollary 5.5.3. *The problem* $\mathrm{WDCE}(S)$ *is* $W[1]$*-hard for parameter* k *and* $\emptyset \neq S \subseteq \{\mathsf{e}, \mathsf{a}\}$.

In fact we may now combine the ultimate results of Theorems 4.6.3 and 5.5.1 and subsequent corollaries.

Theorem 5.5.4. *The problems defined by* WDCE(S) *with* $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}, \mathsf{a}\}$ *are* $W[1]$*-hard for parameter* k*. If* $\mathsf{v} \in S$ *the problems remain* $W[1]$*-hard even when restricted to the class* $\mathrm{WDCE}_1^r(S)$ *with parameter* k*.*

5.6 Bounded Degree Graphs

Fellows *et al.* [70] generalise Vertex Cover to obtain the following problem (as Bounded-Degree Deletion):

> Degree r Deletion
> *Instance:* A graph $G = (V, E)$, a positive integer k.
> *Question:* Can at most k vertices be deleted from G such that no vertex has degree greater than r?

They give a kernel with $(r^3 + 4r^2 + 6r + 4) \cdot k$ vertices via a generalisation of Nemhauser and Trotter's Vertex Cover techniques [124].

This problem may be considered as a special case of WDCE where $\delta(v) = \{0, \ldots, r\}$ for every vertex $v \in V(G)$. We denote this case by $\mathrm{WDCE}^{\leq r}$.

As we have only an upper bound on the degree, edge addition makes no sense as an operation.

Lemma 5.6.1. (G, k, r) *is a* Yes*-instance of* $\mathrm{WDCE}^{\leq r}(\mathsf{v}, \mathsf{e})$ *if and only if* (G, k, r) *is a* Yes*-instance of* $\mathrm{WDCE}^{\leq r}(\mathsf{v})$*.*

Proof. ($\Leftarrow$) Clearly if (G, k, r) is a Yes-instance of $\mathrm{WDCE}^{\leq r}(\mathsf{v})$ then it is a Yes-instance of $\mathrm{WDCE}^{\leq r}(\mathsf{v}, \mathsf{e})$ with the same set of deletions.

($\Rightarrow$) Assume (G, k, r) is a Yes-instance of $\mathrm{WDCE}^{\leq r}(\mathsf{v}, \mathsf{e})$ and let D be the set of vertices and edges deleted in the solution. We construct a new set D' consisting entirely of vertices by adding to D' all vertices in D and for each edge in D we add one endpoint to D'. Every vertex in $G - D'$ is in $G - D$, furthermore every vertex in the graph induced $G - D'$ has degree no greater than in the graph induced by $G - D$. Clearly $|D'| \leq |D|$, therefore (G, k, r) is a Yes-instance of $\mathrm{WDCE}^{\leq r}(\mathsf{v})$. □

5.6.1 A Kernelization for $\mathrm{WDCE}^{\leq r}(\mathsf{v}, \mathsf{e})$

Reduction Rules

Naturally Reduction Rule 1 (Section 4.4.1) applies immediately. Reduction Rule 2 (Section 4.4.1) also applies directly.

However as deleting an element of a clean region does not affect whether or not it is a clean region, we cannot apply Reduction Rule 3 (Section 4.4.1). Furthermore as elements of the clean region beyond the first layer need never be deleted, Reduction Rule 4 (Section 5.2.1), while still correct, can be improved.

Reduction Rule 5: Let (G, k, r) be an instance of $\text{WDCE}^{\leq r}(S)$ with clean region C. Replace (G, k, r) with a new instance (G', k, r) where G' is obtained from G by deleting all layers C_i of C where $i \geq 1$.

Claim 5.6.2. *Reduction Rule* 5 *is sound for* $\text{WDCE}^{\leq r}(\mathsf{v})$.

Proof. Let D be a set of deleted vertices. If D contains any vertices from any layer C_i where $i > 1$, we may replace D with a new solution D' where those vertices are not deleted. As C is clean, the solution will still be correct. □

Kernelization Lemma

Lemma 5.6.3. *Let* (G, k, r) *be a* YES*-instance of* $\text{WDCE}^{\leq r}(\mathsf{v})$ *reduced under Reduction Rules* 1, 2 *and* 5. *Then* $|V(G)| \leq k + k(k + r) + kr(k + r) = O(kr(k + r))$.

Proof. Let (G, k, r) be a YES-instance of $\text{WDCE}^{\leq r}(S)$ reduced under Reduction Rules 1, 2 and 5. As in the proofs of Lemmas 4.4.4, 5.2.2 and 5.2.5 we define three disjoint sets, D, H and X, where D is the set of vertices deleted in the solution, H is the set of vertices adjacent to vertices in D and X is the remaining vertices of the graph. As before H separates D and X, thus no element of X has a neighbour outside of $H \cup X$. $|D| \leq k$ by definition.

Claim 5.6.4. $|H| \leq |D| \cdot (k + r)$.

By Reduction Rule 1 no vertex has degree greater than $k + r$, therefore each vertex in D at most $k + r$ neighbours in H.

Claim 5.6.5. $|X| \leq |H| \cdot r$.

$G - D$ is clean, therefore vertices in H have at most r neighbours in $G - D$. As X consists entirely of clean regions, by Reduction Rule 5, every element of X must be adjacent to some element of H.

Therefore $|V(G)| = |D| + |H| + |X| \leq k + k(k + r) + kr(k + r)$. □

Theorem 5.6.6. $\text{WDCE}^{\leq r}(\mathsf{v})$ *is fixed-parameter tractable for parameter* $k + r$.

By Lemma 5.6.1 we have:

Corollary 5.6.7. $\text{WDCE}^{\leq r}(\mathsf{v}, \mathsf{e})$ *is fixed-parameter tractable for parameter* $k + r$.

5.7 WDCE and Treewidth

We now return to the WDCE problem with an alternate parameterization, the treewidth $\mathrm{tw}(G)$ of the input graph G (see Section 3.5.1 for the definition of treewidth). There are several options for parameterizing, dependent on what combination of the treewidth, the degree bound r and the editing cost k is chosen. Of course if both k and r are part of the parameterization, we already have a complete classification (summarised by Theorem 5.9.1). It can also be observed that if a graph G has treewidth $\mathrm{tw}(G) \leq t$, then there is some vertex $v \in V(G)$ with $d(v) \leq t$ [20]. As we only have deletion operations, the vertex v with $d(v) \leq t$ cannot have its degree constraint satisfied and must be deleted, however the resultant graph G' has $\mathrm{tw}(G') \leq t$, therefore this process cascades and the entire graph must be deleted, therefore for $\mathrm{WDCE}_1^r(\mathsf{v},\mathsf{e})$ if $r > t$ we may immediately answer No. Furthermore if $r \leq t$ and k is also a parameter, then Theorem 5.9.1 applies. As $\mathrm{WDCE}^*(S)$ for $\emptyset \neq S \subseteq \{\mathsf{e},\mathsf{a}\}$ is in P, parameterization by any combination of treewidth, k and r does not affect the complexity.

Samer and Szeider [145] show that the General Factor $= {}_\infty\mathrm{WDCE}_1(\mathsf{e})$ problem is $W[1]$-hard when parameterized by treewidth alone. Recall that the ${}_\infty\mathrm{WDCE}$ case can be obtained from the WDCE case by choosing k to be large.

Proposition 5.7.1 ([145])**.** ${}_\infty\mathrm{WDCE}_1(\mathsf{e})$ *is $W[1]$-hard when parameterized by the treewidth of the input graph. Furthermore it remains $W[1]$-hard when the input graphs are bipartite and the vertices of one partite set are assigned degree list* $\{1\}$.

If we set the vertex weights appropriately, we can also allow vertex deletion. However we can no longer claim unit weights.

Corollary 5.7.2. $\mathrm{WDCE}(\mathsf{v},\mathsf{e})$ *is $W[1]$-hard when parameterized by the treewidth of the input graph. Furthermore it remains $W[1]$-hard when the input graphs are bipartite and the vertices of one partite set are assigned degree list* $\{1\}$.

Proof. Let $(G, \mathrm{tw}(G))$ be an instance of ${}_\infty\mathrm{WDCE}_1(\mathsf{e})$. We reduce to an instance $(G, k, \mathrm{tw}(G))$ of $\mathrm{WDCE}(\mathsf{v},\mathsf{e})$ by setting k to be $\sum_{e\in E(G)} \rho(e)$ and for all vertices $v \in V(G)$ setting $\rho(v) = k + 1$. Therefore no vertex can be deleted within the cost, however we can delete all edges if needed. Clearly $(G, \mathrm{tw}(G))$ is a Yes-instance of ${}_\infty\mathrm{WDCE}_1(\mathsf{e})$ if and only if $(G, k, \mathrm{tw}(G))$ is a Yes-instance of $\mathrm{WDCE}(\mathsf{v},\mathsf{e})$. □

By subdividing the edges and setting the weights of the new vertices to 1 we can restrict the operations to vertex deletion alone, as subdivision does not increase the treewidth.

Corollary 5.7.3. WDCE(v) *is $W[1]$-hard when parameterized by the treewidth of the input graph. Furthermore it remains $W[1]$-hard when the input graphs are bipartite and the vertices of one partite set are assigned degree list $\{1\}$.*

5.7.1 Parameterizations Excluding k

If we consider versions of the problem where the number of edit operations is unbounded, we can obtain some further results for limited cases. In this setting, as the number of deletions is unbounded, we do not consider the trivial case where $V(G) = \emptyset$ as a valid solution.

Lemma 5.7.4. ${}_\infty\text{WDCE}_1^r(\mathsf{v})$ *is fixed-parameter tractable when parameterized by the treewidth of the input graph.*

Proof. As noted earlier, if $r > \text{tw}(G)$ for a graph G, then $(G, \text{tw}(G))$ is a No-instance of ${}_\infty\text{WDCE}_1^r(\mathsf{v})$, as the entire graph would have to be deleted. However if $r \leq \text{tw}(G)$, we may apply Courcelle's Theorem with the following second order sentence:

$$\exists S \forall v \forall u (Vv \to Sv \vee \exists v_1, \ldots, v_r (\textstyle\bigwedge_{i \neq j \in [r]} (v_i \neq v_j) \wedge \\ \textstyle\bigwedge_{i \in [r]} (\neg Sv_i \wedge Avv_i \wedge v \neq v_i) \wedge (Avu \to Su \vee \textstyle\bigvee_{i \in [r]} u = v_i)))$$

where Axy is shorthand for $\exists e(Ee \wedge Vx \wedge Vy \wedge Ixe \wedge Iye)$ (i.e, x and y are adjacent). The sentence ensures that there is a set S (the deleted vertices) such that for every vertex v and every vertex u, either v is deleted, or it is adjacent to r distinct vertices that haven't been deleted, and if u is adjacent to v, then it is one of these vertices, or it has been deleted. □

This can be extended to include edge deletion.

Lemma 5.7.5. ${}_\infty\text{WDCE}_1^r(\mathsf{v}, \mathsf{e})$ *is fixed-parameter tractable when parameterized by the treewidth of the input graph.*

Proof. As before if $r > \text{tw}(G)$, the instance is a No-instance. Then we need only construct a second order logic sentence that encodes the problem.

$$\exists S \forall v \forall e (Vv \to Sv \vee (\exists e_1, \ldots, e_r, v_1, \ldots, v_r (\phi_1 \wedge \phi_2)))$$

where ϕ_1 is the conjunction of subclauses (1)–(5):

(1) $\bigwedge_{i \in [r]} \neg Se_i \wedge \neg Sv_i$ "e_i and v_i have not been deleted;"

(2) $\bigwedge_{i\in[r]} Ee_i \wedge Vv_i$ "e_i is an edge and v_i is a vertex;"

(3) $\bigwedge_{i\in[r]} v_i \neq v$ "v is not equal to any v_i;"

(4) $\bigwedge_{i\in[r]} Iv_ie_i \wedge Ive_i$ "v and v_i are adjacent;"

(5) $\bigwedge_{i\neq j\in[r]} v_i \neq v_j$ "the v_is are distinct;"

and

$$\phi_2 = Ive \rightarrow (\bigvee_{i\in[r]} (e = e_i) \vee Se \vee \exists u(Iue \wedge u \neq v \wedge Su)).$$

ϕ_2 ensures that if there is an edge incident to v, then either it is one of the r edges making up the the regular degree of v, it was deleted, or its other endpoint was deleted. □

If vertex deletion and edge addition are allowed, then the problem becomes trivially polynomial.

Lemma 5.7.6. ${}_\infty\mathrm{WDCE}_1^r(\mathsf{v},\mathsf{e},\mathsf{a})$ *and* ${}_\infty\mathrm{WDCE}_1^r(\mathsf{v},\mathsf{a})$ *are polynomial-time solvable.*

Proof. As the number of editing steps is unlimited, we can simply delete all but $r+1$ vertices, and make the graph a K_{r+1}.

If there are fewer than $r+1$ vertices, it is not possible to have an r-regular graph, and we answer NO immediately. □

5.8 A Note on Extended Regularity Constraints

We now present additional results not otherwise included in this thesis. The techniques in Chapters 4 and 5 can be applied to other constraints similar to regularity with little change. We briefly summarise these extensions.

5.8.1 Edge-Degree Regularity

For an edge $uv \in E(G)$, the degree of uv, denoted $d(uv)$ and called the *edge-degree*, is the sum of the degrees of the endpoints, $d(u)+d(v)$. If for every edge $uv \in E(G)$ we have $d(uv) = r$, then G is *edge-degree r-regular*. Edge-degree constraints naturally extend vertex based degree constraints, notably any r-regular graph is edge-degree $2r$-regular. However an edge-degree regular graph may not be regular. Therefore the class of edge-degree regular graphs forms a proper superclass of the class of regular graphs.

We define the WEIGHTED EDGE DEGREE CONSTRAINT EDITING problem, or WEDCE similarly to the WDCE problem. For convenience we extend the notation for degree and weighted degree to edge-degree and weighted edge-degree.

> WEDCE(S)
>
> *Instance:* A graph $G = (V, E)$, two integers k and r, a weight function $\rho : V \cup E \to \{1, 2, \dots\}$, and a degree list function $\delta : E \to 2^{\{0,\dots,r\}}$.
>
> *Question:* Can we obtain from G a graph $G' = (V', E')$ using editing operations from S only, such that for all $uv \in E'$ we have $\sum_{uu' \in E'} \rho(uu') + \sum_{vv' \in E'} \rho(vv') \in \delta(uv)$, with total editing cost at most k?

Edge addition makes less sense in this context, much as vertex addition makes little sense in the WDCE context. Thus we restrict ourselves to vertex deletion and edge deletion.

$\text{WEDCE}_1^r(\mathsf{v})$ remains para-NP-complete for parameter r, and the $W[1]$-hardness results of Theorem 4.6.3 carry over to the WEDCE cases. $\text{WEDCE}(\mathsf{v}, \mathsf{e})$ is fixed-parameter tractable when parameterized by $k + r$, by instantiation of the generic search tree algorithm (Section 4.3). With correct modification of the definition of a clean region, the kernelization given in Section 5.2 gives a kernel of $O(k^2 r^{k+1} + k r^{k+2})$ vertices for $\text{WEDCE}^*(\mathsf{v}, \mathsf{e})$. If we restrict the editing operations to only edge deletion, this kernel can be reduced to $O(kr)$ vertices.

5.8.2 Edge Regularity and Strong Regularity

A graph G is (r, λ)-edge regular if every vertex has degree r and every edge $uv \in E(G)$ has $|N(u) \cap N(v)| = \lambda$. A graph G is (r, λ, μ)-strongly regular if it is (r, λ)-edge regular and for every pair u, v of non-adjacent vertices we have $|N(u) \cap N(v)| = \mu$. For this set of constraints, our problem becomes the WEIGHTED STRONGLY REGULAR EDITING (WSRE) problem.

> WSRE(S)
>
> *Instance:* A graph $G = (V, E)$, four integers k, r and $\lambda, \mu \leq r$, a weight function $\rho : V \cup E \to \{1, 2, \dots\}$, a degree function $\delta : V \to 2^{\{0,\dots,r\}}$, and two neighbourhood functions $\nu : V \times V \to 2^{\{0,\dots,\lambda\}}$ and $\xi : V \times V \to 2^{\{0,\dots,\mu\}}$.
>
> *Question:* Can we obtain from G a graph $G' = (V', E')$ using editing operations from S only, such that for all $v \in V'$ we have $\sum_{uv \in E'} \rho(uv) \in$

$\delta(v)$, for every $uv \in E'$ we have $|N(u) \cap N(v)| \in \nu(u,v)$, and for every $uv \notin E'$ we have $|N(u) \cap N(v)| \in \xi(u,v)$, with total editing cost at most k?

By choosing $\xi(u,v) = \{0,\ldots,r\}$ for all $u,v \in V(G)$ we can express the constraint of edge regularity.

As both constraints are extensions of regularity, the para-NP-completeness and $W[1]$-hardness results for WDCE carry over immediately. When parameterized by $k+r$ the kernelization for WDCE$^*(\mathsf{v},\mathsf{e})$ can be extended, again with appropriate redefinition of a clean region, to give a kernel for WSRE$^*(\mathsf{v},\mathsf{e})$ with $O(kr^2(k+r))$ vertices. The generic search tree algorithm of Section 4.3 can be instantiated to show fixed-parameter tractability for WSRE(v,e). If we allow edge addition, then the logic approach of Section 5.4 can be extended, with appropriate predicates, to show fixed-parameter tractability for WSRE(S) where $\emptyset \neq S \subseteq \{\mathsf{v},\mathsf{e},\mathsf{a}\}$ with parameter $k+r$.

5.9 Conclusion

The key results of Chapters 4 and 5 can be summarised by the following theorem:

Theorem 5.9.1. *For all non-empty subsets S of $\{\mathsf{v},\mathsf{e},\mathsf{a}\}$ the problem* WDCE(S) *is fixed-parameter tractable for parameter $k+r$, and $W[1]$-hard for parameter k. If $\mathsf{v} \in S$ then WDCE(S) remains $W[1]$-hard for parameter k even when all degree lists are restricted to $\{r\}$ and all vertices and edges have unit weight* 1.

However the additional results further illuminate the complexity of the problem class WDCE by showing that the problem is in general para-NP-complete for parameter r, thus indicating that the combined parameter $k+r$ is in a sense minimal in that both k and r are needed to render the problem tractable. Furthermore the NP-completeness of Edge Replacement Set, and the existence of a composition algorithm for it indicate that a reasonable algorithm for the cases where both vertex deletion and edge addition are available is unlikely. The best hope for progress here is that some new structural insight allows a new avenue of attack on the WDCE$_1^r(\mathsf{v},\mathsf{a})$ case.

Chapter 6

Degenerate Graphs

In this chapter we consider editing problems for degenerate graphs. A graph G is *r-degenerate* if every non-empty subgraph of G has a vertex of degree at most r [53]. The *degeneracy* of a graph G is the smallest number r such that G is r-degenerate. When we speak of "degenerate graphs" we implicitly assume a fixed upper bound r on their degeneracy. Degeneracy represents a more general form of degree constraint, where the constraint is expressed in terms of a property of the graph as a whole. Degenerate graph classes include planar graphs, regular graphs, graphs of bounded degree, graphs of bounded treewidth and H-minor-free graphs [95, 159]. Bounded degeneracy is also equivalent to bounded arboricity [53]. Several NP-hard problems become polynomial time solvable for degenerate graphs, for example any d-degenerate graph is $(d+1)$-colourable and such a colouring can be found in polynomial time. Moreover many problems that are $W[1]$-hard become fixed-parameter tractable when restricted to degenerate graphs [7, 28, 82]. Unfortunately there are also numerous problems that remain hard even when the degeneracy of the graph is a fixed constant. Despite this the wealth of problems which are fixed-parameter tractable makes the question of whether it is possible to obtain a degenerate graph with a limited amount of editing interesting. Unfortunately, as we will show, when parameterized by the number of editing operations, this problem is $W[P]$-hard, even if the degeneracy is fixed and the graph has fixed maximum degree.

6.1 Some Problems that are Tractable on Degenerate Graphs

We present fixed-parameter tractability results for various problems with input restricted to degenerate graphs where the problems are in general $W[1]$-hard or $W[2]$-hard.

6.1.1 Preliminary Definitions

Let G be a graph and $A, B \subseteq V(G)$. We say A *dominates* B if for every vertex $b \in B$ there exists a vertex $a \in A$ such that $ab \in E(G)$. If A dominates $V(G)$, then A is a *dominating set* for G. An *independent set* in G is a set $I \subseteq V(G)$ such that for all vertices $x, y \in I$ we have $xy \notin E(G)$. An independent set I is maximal if and only if there is no independent set I' such that $I \subset I'$. Note that an independent set is maximal if and only if it is also a dominating set.

We now give an alternative definition of degeneracy. Given a graph G, let $v_1, \ldots, v_n$ be an ordering of $V(G)$, and let $V_i = \bigcup_{j \leq i} \{v_j\}$. G is r-degenerate if and only if an ordering exists such that for every vertex v_i we have $d_{V_i}(v_i) \leq r$ [102]. This ordering provides an intuitive way of dealing with degeneracy, as we can think of testing for degeneracy as a sequence of deletions where we iteratively pick a vertex of degree at most r to remove from the graph. The entire graph can be deleted this way if and only if it is r-degenerate [102].

6.1.2 Independence Problems

The first tractability result was mentioned as an observation by Golovach and Villanger [82] without proof. The kernelization was not noted.

> r-Degenerate Independent Set
>
> *Instance:* An r-degenerate graph $G = (V, E)$, a positive integer k.
>
> *Parameter:* k.
>
> *Question:* Does G have an independent set of size at least k?

Lemma 6.1.1. r-Degenerate Independent Set *is fixed-parameter tractable with a search tree with at most* $\mathrm{tr}(r+1, k) = ((r+1)^{k+1} - 1)/r$ *vertices, and a kernel with at most* $(r+1)k$ *vertices.*

Proof. Both of these results are simple extensions of the proofs for the equivalent results for planar graphs [78].

Given any vertex v in the graph, we know that any maximal independent set contains a vertex from $N[v]$, and that every independent set is a subset of some maximal independent set. Therefore there is a simple bounded search tree algorithm to solve the problem.

1. Choose a vertex u of degree at most r, whose existence is guaranteed by the degeneracy of the graph.

2. Branch on choosing a vertex v from $N[u]$ to add to the independent set.

3. Remove $N[v]$ from the graph and reduce k by 1.

4. If $k = 0$ return YES, if $k > 0$ and the graph is not empty, go to step 1, otherwise return NO.

The maximum branching factor of the search tree is $|N[u]| \leq r+1$, and the tree has depth at most k. Therefore the tree has at most $\mathrm{tr}(r+1, k)$ vertices.

For the kernelization we note that it is possible in polynomial time to colour an r-degenerate graph with $r+1$ colours (a greedy approach suffices). Therefore given such a colouring, at least one of the colour classes is of size at least $n/(r+1)$. Further, this forms a independent set. Then if $n/(r+1) \geq k$, this is a YES-instance. Otherwise $n < (r+1)k$, and the instance is kernelized. □

The search tree algorithm for r-DEGENERATE INDEPENDENT SET can also easily be adapted to demonstrate fixed-parameter tractability for the following problem:

> r-DEGENERATE MINIMUM MAXIMAL INDEPENDENT SET
> *Instance:* An r-degenerate graph $G = (V, E)$, a positive integer k.
> *Parameter:* k.
> *Question:* Does G have a maximal independent set of size at most k?

This problem can be equivalently formulated as INDEPENDENT DOMINATING SET [75].

Corollary 6.1.2. *r-DEGENERATE MINIMUM MAXIMAL INDEPENDENT SET is fixed-parameter tractable for parameter k.*

Proof. As noted in the previous proof, for every vertex v in the graph, at least one vertex from $N[v]$ is contained in any maximal independent set. Then we apply the search tree approach as for r-DEGENERATE INDEPENDENT SET, except we now answer YES if there exists a leaf of the search tree of distance at most k from the root, and the graph at this leaf is empty. Note that we need never extend the tree beyond distance k from the root, as either it is a YES answer at distance k from the root, or we can terminate that branch of the search. □

6.1.3 Clique Problems

Restricting the input to degenerate graphs also renders several clique based problems fixed-parameter tractable.

r-Degenerate #Clique

Instance: An r-degenerate graph $G = (V, E)$, a positive integer k.

Parameter: k.

Question: Compute the number of k-cliques in G.

Lemma 6.1.3. r-Degenerate #Clique *is fixed-parameter tractable.*

Proof. We can apply the following algorithm:

1. Find a vertex v of degree at most r.
2. Try all combinations (at most $\binom{r}{k-1}$) of creating a k-clique with v and $k-1$ neighbours. Count each successful attempt.
3. Remove v.
4. If there are at least k vertices in the graph, continue from step 1.
5. Return the total count.

Clearly we can always find a vertex of degree at most r, as the graph is r-degenerate. Step 2 counts all cliques containing this vertex, and no others, so we have not over counted, and we can safely remove the vertex from consideration. □

This algorithm can easily be adapted for the following general problem which is $W[1]$-hard on general graphs. Π is a polynomial time decidable property.

r-Degenerate Π Clique

Instance: An r-degenerate graph $G = (V, E)$, a positive integer k.

Parameter: k.

Question: Does G contain a k-clique with property Π?

Corollary 6.1.4. r-Degenerate Π Clique *is fixed-parameter tractable.*

Proof. Apply the enumeration algorithm for r-Degenerate #Clique, and check if any clique satisfies Π. □

This gives a classification for the following problems, all of which are $W[1]$-complete on general graphs. Partitioned Clique is the source of the reduction

for the main hardness result of Chapter 4, Theorem 4.6.3, and Clique is the source for several reductions in Chapters 4, 5 and this chapter.

r-Degenerate Clique

Instance: An r-degenerate graph $G = (V, E)$, a positive integer k.

Parameter: k.

Question: Does G contain a k-clique?

r-Degenerate Partitioned Clique

Instance: An r-degenerate graph $G = (V, E)$, with V partitioned into k equal size disjoint subsets.

Parameter: k.

Question: Does G contain a k-clique where each vertex of the clique is in a different set in the partition?

r-Degenerate Dominating Clique

Instance: An r-degenerate graph $G = (V, E)$, a positive integer k.

Parameter: k.

Question: Does G contain a k-clique that dominates the entire graph?

Corollary 6.1.5. *r-Degenerate Clique, r-Degenerate Partitioned Clique and r-Degenerate Dominating Clique are fixed-parameter tractable.*

6.1.4 Domination Problems

A *perfect code* of a graph is a set of vertices $V' \subseteq V$ such that V' is an independent set and every vertex in $V \setminus v'$ has exactly one neighbour in V'.

r-Degenerate Perfect Code

Instance: An r-degenerate graph $G = (V, E)$, an integer k.

Parameter: k.

Question: Does G have a perfect code of size at most k?

r-Degenerate Perfect Code was shown to be fixed-parameter tractable under the name Efficient Domination k-Set for degenerate graphs by Cai and Kloks [28].

Lemma 6.1.6 ([28])**.** *r-Degenerate Perfect Code is fixed-parameter tractable.*

Proof. Consider an arbitrary perfect code V'. Given any vertex v, exactly one vertex in $N[v]$ is in the perfect code. If v has a neighbour u in the perfect code, no

other vertex $u \neq w \in N(v)$ can be in the perfect code, however w may still require a vertex of $N(w)$ to be in the perfect code. Therefore it is useful to introduce an annotated version of the problem, to which a bounded search tree approach can be applied with more clarity.

> Annotated r-Degenerate Perfect Code
> *Instance:* An r-degenerate graph $G = (V, E)$, a set $B \subseteq V$ and an integer k.
> *Parameter:* k.
> *Question:* Is there a set $V' \subseteq V \setminus B$ such that V' is a perfect code for G, with size at most k?

Then the original problem will be the special case of the annotated problem where initially $B = \emptyset$. The search tree algorithm runs as follows:

1. Choose a vertex v of degree at most r.
2. Branch on all possibilities of choosing a vertex u to be in the perfect code from $N[v] \setminus B$. If $N[v] \subseteq B$, then terminate this branch.
3. Set $B = B \cup N[N[u]]$ and reduce k by one.
4. Delete $N[u]$.
5. If $k \geq 0$ and all vertices are deleted, answer Yes.
6. If $k = 0$ and vertices remain, terminate this branch.
7. If $k > 0$ and vertices remain, return to step 1.
8. If no branch returns Yes, return No.

Considering the original problem, the set B is then employed to ensure that vertices adjacent to the neighbourhood of a vertex in the perfect code cannot be themselves added to the perfect code. This ensures the correctness of the solution. The branching factor of the tree is at most $r + 1$, as we choose a vertex out of a set of at most $r + 1$ vertices at each iteration. The depth of the tree is at most k. Therefore the tree has at most $\mathrm{tr}(r + 1, k) = ((r + 1)^{k+1} - 1)/r$ vertices. □

Alon and Gutner [7] first demonstrated the fixed-parameter tractability of r-Degenerate Dominating Set, using an interesting annotated version of the

problem, illustrated below.

> r-Degenerate Dominating Set
>
> *Instance:* An r-degenerate graph $G = (V, E)$, an integer k.
>
> *Parameter:* k.
>
> *Question:* Does G have a dominating set of size at most k?

Lemma 6.1.7 ([7]). r-Degenerate Dominating Set *is fixed-parameter tractable.*

First we define a *black/white* graph, where the vertex set $V = B \uplus W$ is a disjoint union of the set B of *black* vertices and the set W of *white* vertices. Now we may define the r-Degenerate Black/White Dominating Set problem:

> r-Degenerate Black/White Dominating Set
>
> *Instance:* A black/white graph $G = (B \uplus W, E)$, an integer k.
>
> *Parameter:* k.
>
> *Question:* Is there a set $V' \subseteq B \uplus W$ of size at most k that dominates B?

It is for this problem that Alon and Gutner show fixed-parameter tractability. Then r-Degenerate Dominating Set may be solved by first setting $B = V$ and $W = \emptyset$. The proof involves the following key lemma:

Lemma 6.1.8 ([7]). *Let $G = (B \uplus W, E)$ be an r-degenerate black/white graph. If $|B| > (4r + 2)k$, then there are at most $(4r + 2)k$ vertices in G that dominate at least $|B|/k$ vertices of B.*

Alon and Gutner's algorithm uses a bounded search tree approach. Given an instance (G, k) of r-Degenerate Black/White Dominating Set, either $|B| \leq (4r + 2)k$, in which case there are at most $O(k^{(4r+2)k})$ ways of splitting B into k parts, each of which has a vertex dominating it, or $|B| > (4r + 2)k$, in which case by Lemma 6.1.8 there are at most $(4r + 2)k$ vertices which dominate enough of the graph to be chosen to be in the dominating set. The algorithm proceeds by branching on the two cases as appropriate. When a vertex is added to the dominating set, it is removed from the graph, and all white neighbours are coloured black. This results in a $O(k^{rk} \cdot |V(G)|)$-time algorithm.

Subsequently Raman *et al.* [139] extended this result to r-Degenerate Vector Dominating Set.

> r-Degenerate Vector Dominating Set
> *Instance:* An r-degenerate graph $G = (V, E)$, a function $L : V \to \mathbb{N}^+$ and a positive integer k.
> *Parameter:* k.
> *Question:* Is there a set $V' \subseteq V$ such that $|N(v) \cap V'| \geq L(v)$ for every $v \in V \setminus V'$ and $|V'| \leq k$?

Lemma 6.1.9 ([139])**.** r-Degenerate Vector Dominating Set *is fixed-parameter tractable.*

6.2 Some Hard Problems for Degenerate Graphs

Although restricting the input to degenerate graphs is a powerful method of obtaining fixed-parameter tractability results, it is also easy to generate problems that are hard even for 2-degenerate graphs. A *subdivided k-clique* in a graph G is a set V' of k independent vertices where for every pair of vertices $u, v \in V'$ there is a vertex w of degree 2 such that $uw, vw \in E(G)$.

> Subdivided Clique
> *Instance:* A graph $G = (V, E)$, a positive integer k.
> *Parameter:* k.
> *Question:* Does G have a subdivided k-clique?

Lemma 6.2.1. Subdivided Clique *is $W[1]$-hard, even when restricted to 2-degenerate graphs.*

Proof. Assume $k \geq 4$.

The reduction is from Clique. Let (G, k) be an instance of Clique. We construct an instance (G', k) of Subdivided Clique where for each vertex $v \in V(G)$ we add a vertex v' to $V(G')$. For each edge $uv \in E(G)$ we add a vertex e_{uv} to $V(G')$ and two edges $u'e_{uv}$ and $v'e_{uv}$ to $E(G')$ where $u', v' \in V(G')$ are the vertices corresponding to u and v respectively. As all e_{uv} vertices are of degree 2, and their deletion leaves all other vertices with degree 0, the graph is 2-degenerate.

We claim that (G, k) is a Yes-instance of Clique if and only if (G', k) is a Yes-instance of Subdivided Clique. The forward direction of the proof is trivial, so we concentrate on the reverse.

Assume (G', k) is a YES-instance of SUBDIVIDED CLIQUE. Then there is a set of k vertices that are pairwise at distance two that form the subdivided clique. By definition each of these vertices is adjacent to at least $k-1$ other vertices. As the e_{uv} vertices added in the subdivision have degree 2, the vertices of the subdivided clique must correspond to vertices of G. Therefore the corresponding vertices in G must have been mutually adjacent, i.e. these k vertices formed a k-clique in G. □

Golovach and Villanger [82] give a hardness proof for the following domination problem:

(k, d)-CENTER
Instance: A graph $G = (V, E)$, integers k and d.
Parameter: k.
Question: Is there a set $V' \subseteq V$ of size at most k such that all vertices are of distance at most d from some vertex in V'?

Here we present a similar hardness proof for a variant of the problem:

ANNOTATED DISTANCE DOMINATION
Instance: A black/white graph $G = (B \uplus W, E)$, a function $D : B \uplus W \to \mathbb{N}^+$ and an integer k.
Parameter: k.
Question: Is there a set $V' \subseteq B$ such that for every $v \in (B \uplus W) \setminus V'$ there is a vertex $u \in V'$ where $d(u, v) \leq D(v)$?

Lemma 6.2.2. ANNOTATED DISTANCE DOMINATION *is* $W[2]$*-hard, even when restricted to 2-degenerate graphs and* $D(v) = 2$ *for every vertex* v.

Proof. The reduction is from DOMINATING SET, which is known to be $W[2]$-complete [59]:

DOMINATING SET
Instance: A graph $G = (V, E)$ and a positive integer k.
Parameter: k.
Question: Does G have a dominating set of size at most k?

Let (G, k) be an instance of DOMINATING SET. We construct an instance $((G', D), k+1)$ of ANNOTATED DISTANCE DOMINATION where for each vertex $v \in V(G)$ we add a vertex v' to B. Then for each edge $uv \in E(G)$ with endpoints u and v we add a vertex e_{uv} to W and two edges $u'e_{uv}$ and $v'e_{uv}$ to $E(G')$

where u' and v' correspond to u and v respectively. Add a vertex x to B. For each vertex $e_{uv} \in W$ add a vertex e_{uvx} to W and the edges $e_{uv}e_{uvx}$ and xe_{uvx} to $E(G')$. Finally add two vertices y and z to W and the edges xy and yz to $E(G')$. Set $D(v) = 2$ for all vertices. The vertices e_{uvx} have degree 2, and if they are deleted, the vertices e_{uv} have degree 2. The vertices y and z have degree at 2 and 1 respectively. When the vertices e_{uvx} are removed, the vertex x has degree 2. All other vertices are adjacent only to the vertices e_{uv}. Therefore G' is 2-degenerate.

(G, k) is a YES-instance of DOMINATING SET if and only if $((G', D), k + 1)$ is a YES-instance of ANNOTATED DISTANCE DOMINATION. Again the forward direction is trivial, as the at most k vertices corresponding to the dominating set of G along with the vertex x form a suitable distance dominating set for G' with size at most $k + 1$.

Assume $((G', D), k + 1)$ is a YES-instance of ANNOTATED DISTANCE DOMINATION. x must be in the distance dominating set, z must be dominated, and x is the only black vertex within a distance of 2. Thus all white vertices are dominated by x. However x dominates no other vertices. Thus the remaining un-dominated vertices are all black, that is, those vertices corresponding to the original vertices of the graph, precisely $B \setminus \{x\}$. Therefore there is a set $S \subseteq B \setminus \{x\}$ of size at most k such that every vertex of $B \setminus \{x\}$ is at distance at most two from some member of S. Then in G this would form a dominating set of size at most k. □

This type of proof can be constructed for most graph problems, giving a 'distance' or 'subdivided' version that is hard even on 2-degenerate graphs.

6.3 Editing to Obtain Degenerate Graphs

Despite the ease of developing hard problems for degenerate graphs, there are obviously many natural problems, including main problems such as CLIQUE, INDEPENDENT SET and DOMINATING SET, that are fixed-parameter tractable for degenerate graphs as discussed in Section 6.1. Thus we move to the main results of the chapter, and examine editing problems for degenerate graphs. Unfortunately these seem to be hard, $W[P]$-hard in all examined cases, even when the input is restricted to bounded degree graphs.

The general class of problems we will examine is as follows:

r-DEGENERATE(S)
Instance: A graph $G = (V, E)$, an integer $k \geq 0$.
Parameter: k.
Question: Can an r-degenerate graph H be obtained from G by at most k editing steps using the operations of S?

In the case of degenerate graphs, edge addition is never a useful editing operation, so we do not consider any variant involving it. Therefore S is restricted to vertex deletion (v) and edge deletion (e). We also note that a 0-degenerate graph is an independent set, so the vertex deletion variant becomes VERTEX COVER, and is fixed-parameter tractable [26], the edge deletion variant is trivially polynomial time solvable. A 1-degenerate graph is a forest, so 1-DEGENERATE(v) is the FEEDBACK VERTEX SET problem and 1-DEGENERATE(e) is the FEEDBACK EDGE SET problem, both of which are also fixed-parameter tractable [58, 36].

Lemma 6.3.1. *(G, k) is a* YES*-instance of* r-DEGENERATE(v, e) *if and only if (G, k) is a* YES*-instance of* r-DEGENERATE(v).

The proof of Lemma 6.3.1 runs essentially identically to that of Lemma 5.6.1.

As we can always preferentially delete a vertex over an edge by Lemma 6.3.1, 1-DEGENERATE(v, e) is also fixed-parameter tractable.

Proposition 6.3.2. *r-DEGENERATE(S) is fixed-parameter tractable for $r \leq 1$ where $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}\}$.*

6.3.1 A Note on Degeneracy

When considering the degeneracy of a graph, it is convenient to think in terms of the greedy algorithm for determining whether a graph is degenerate, which successively removes vertices of low degree as outlined in Section 6.1.1. Thus when demonstrating the degeneracy of a graph it will frequently be expressed by the ability to remove vertices as they have sufficiently low degree. Clearly this may create some confusion between the act of deleting vertices to create a degenerate graph, as defined in the problem, and the descriptive deletion of vertices to demonstrate degeneracy. To the end of clarifying this possible confusion, we define *deletion* to refer to the editing operation specified in the problem, and *clearing* to refer to the descriptive act of removing vertices to demonstrate degeneracy.

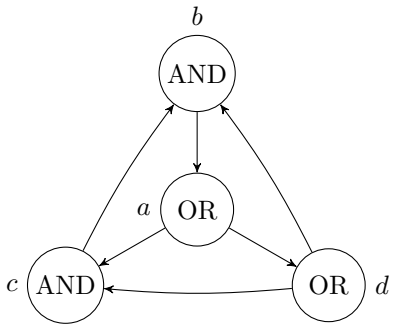


Figure 6.1: An example monotone cyclic circuit. The sets $\{a\}$ and $\{b\}$ activate the entire circuit, whereas the sets $\{c\}$ and $\{d\}$ do not.

6.3.2 Cyclic Monotone Circuit Activation and Almost Degenerate Gate Gadgets

We now use a relaxation of the circuits described in Section 2.3.4, where we no longer require the underlying graph D to be acyclic. Recall that a circuit is *monotone* if it includes no NOT gates. Let A be a set of gates of a cyclic monotone circuit D. We define a sequence of sets $A_0, A_1, A_2, \ldots$ where $A_0 = A$ and A_{i+1} is obtained from A_i by adding all all gates that output true if all their predecessors in A_i are set to true. The sequence is extended until $A_i = A_{i+1}$, we then call A_i the *closure* of A. As D is finite the sequence has at most $|V(D)|$ elements. If the closure of A is D we say that A *activates* the circuit D. An example of a monotone cyclic circuit is given in Figure 6.1.

For the subsequent hardness proofs we use the following problem:

> Cyclic Monotone Circuit Activation
>
> *Instance:* A cyclic monotone circuit D, a positive integer k.
>
> *Parameter:* k.
>
> *Question:* Is there a set A of size at most k that activates D?

This was shown to be $W[P]$-complete by Szeider [157]. Moreover the result holds with the following restriction:

Lemma 6.3.3 ([157])**.** Cyclic Monotone Circuit Activation *is $W[P]$-complete, and remains $W[P]$-complete if there are no input or output gates and each gate has at most two inputs and two outputs.*

Before introducing the reduction proper, we will present the construction of the gate gadgets used in the reduction. Two gadgets will be used for representing gates, an AND gadget and an OR gadget, which will be used to represent AND gates and

OR gates respectively. Both gadgets consist of three parts, an *input* vertex v^i, an independent set L, and an r-clique C which includes a designated *output* vertex v^o. There is an edge between every vertex in L and v^i, and between every vertex in L and every vertex in C. The difference between AND gadgets and OR gadgets occurs in the size of L. An AND gadget has r vertices in L, an OR gadget has $r-1$ vertices in L. Figures 6.2 and 6.3 give examples of the gadgets. Note that the construction of the gadgets require that $r \geq 2$, otherwise $L = \emptyset$.

When connected to form the representation of the circuit, each input vertex will have one or two additional edges, coming from the output vertices of other gate gadgets (the *predecessor* gadgets), and each output vertex will have one or two additional edges connecting to the input vertices of other gate gadgets (the *successor* gadgets). Initially, no vertex in the graph has degree less than $r+1$, thus the graph is not r-degenerate.

Then for an AND gadget A, if all predecessor gadgets are cleared or deleted, the input vertex of A will have degree r, and may be cleared, which leaves all vertices in L with degree r, thus they may be cleared, and then the vertices of $C - v^o$ have degree $r-1$ and may also be cleared. This leaves the output vertex v^o with degree at most 2. As we assume that $r \geq 2$, v^o may also be cleared. Note that if a predecessor gate of A is not cleared or deleted, the input vertex will have degree at least $r+1$ and will not be able to be cleared.

Similarly for an OR gadget O, if at least one of the predecessor gates of O is cleared or deleted, the input vertex will have degree at most r, and the gate gadget can be cleared as for the AND gadget.

Note that the vertices in L always have degree $r+1$. As we do not modify this in any construction, the graphs constructed are always $r+1$ degenerate. Furthermore the highest degree vertex in any gadget is the ouput vertex, which at most can have degree $2r+1$, therefore all the graphs constructed have maximum degree at most $2r+1$.

6.3.3 Vertex Deletion

Theorem 6.3.4. *r-DEGENERATE(v) with $r \geq 2$ is $W[P]$-hard, and remains $W[P]$-hard even when the input graph is $(r+1)$-degenerate and has maximum degree $2r+1$.*

Proof. We reduce from CYCLIC MONOTONE CIRCUIT ACTIVATION. For this reduction we assume the additional restrictions of no input or ouput gates, and fan-in

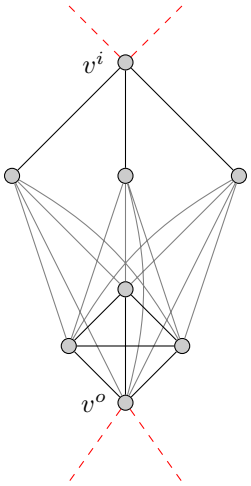


Figure 6.2: OR gadget for $r = 4$

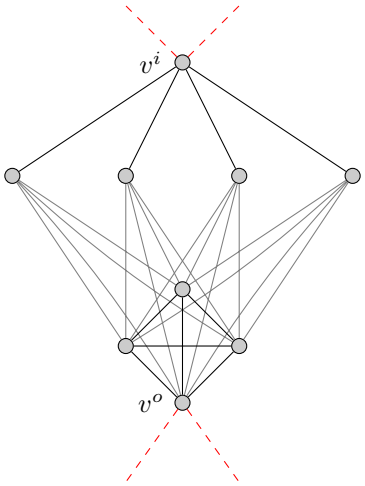


Figure 6.3: AND gadget for $r = 4$

and fan-out of at most 2 hold.

Consider an instance (D, k) of Cyclic Monotone Circuit Activation. Then we construct an instance (G, k) of r-Degenerate(v) as follows using the AND and OR gadgets described in Section 6.3.2:

If a gate has only 1 input and 1 output, it may be removed from the circuit, and its input connected to its output, as it will output true if and only if the input is true, therefore it performs no function. For each gate g in D we have a gadget g' in G according to the following scheme:

- If g is an AND gate, g' is an AND gadget.
- If g is an OR gate with one input, g' is an AND gadget.
- If g is an OR gate with two inputs, g' is an OR gadget.

For each gate g in D with corresponding gadget g' in G let h be a successor of

g with corresponding gadget h' in G. There is an edge between the output vertex $v^o_{g'}$ of g' and the input vertex $v^i_{h'}$ of h'.

Assume (D, k) is a YES-instance of CYCLIC MONOTONE CIRCUIT ACTIVATION, then there is a set of at most k gates $D' = \{g_1, \ldots, g_{k'}\}$ that activates D. Let $G' = \{g'_1, \ldots, g'_{k'}\}$ be the corresponding set of gadgets in D'. Then we argue that deleting the input vertex of each gadget in G' makes G r-degenerate. For all gadgets in G', deleting the input vertex renders them r-degenerate as described in Section 6.3.2, and they may be cleared. Let g be a gate in $D - D'$ with corresponding gadget g' in G. As D is activated, sufficient predecessors of g must be activated. Assume g is an AND gate, then all predecessors of g must be activated. Therefore all predecessors of g' can be deleted or cleared, so $d(v^i_{g'}) = r$, and g' can be cleared. Assume g is an OR gate, then at least one of the predecessors of g is activated. Therefore at least one of the predecessor of g' can be deleted or cleared. If g' had one predecessor, g' is an AND gadget, where $|L| = r$, so initially $d(v^i_{g'}) = r + 1$. After the predecessor is deleted or cleared, $d(v^i_{g'}) = r$, so g' can be cleared. If g' had two predecessors, then g' is an OR gadget, with $|L| = r - 1$, so deleting or clearing one predecessor leaves $d(v^i_{g'}) = r$, and g' can also be cleared. Applying this argument inductively from the set D' shows that (G, k) is a YES-instance of r-DEGENERATE(v).

Assume that (G, k) is a YES-instance of r-DEGENERATE(v). Then there is a set of at most k vertices $V' = \{v_1, \ldots, v_{k'}\}$ whose deletion leaves G r-degenerate. We may assume that each vertex v_j comes from a different gadget and furthermore that $v_j = v^i_{g'_j}$ for some gadget g'_j. Section 6.3.2 shows that one vertex deletion is sufficient to leave the gadget r-degenerate, therefore if there were two vertices deleted from one gadget, we may obtain a smaller solution by deleting only the input vertex of the gadget. Thus the set of deleted vertices V' corresponds to a set of at most k gadgets $G' = \{g'_1, \ldots, g'_{k'}\}$. We argue that the corresponding set of gates $D' = \{g_1, \ldots, g_{k'}\}$ in D activates D. Let g' be a gadget in $G - G'$ with corresponding gate g in $D - V(D')$. As $G - G'$ is r-degenerate, some of the predecessors of g' must be cleared before g' can be cleared. If g' is an AND gadget, then all predecessors of g' are deleted or cleared. Therefore in D all predecessors of g are activated, and g will be activated. If g' is an OR gadget, then at least one predecessor of g' is deleted or cleared, therefore in D one predecessor of the OR gate g is activated, and g will be activated. Therefore applying this argument inductively starting from G' shows that (D, k) is a YES-instance of CYCLIC MONOTONE CIRCUIT ACTIVATION. □

It is also possible to show that r-DEGENERATE(v) is in $W[P]$, and therefore $W[P]$-complete. We sketch the basic idea: degeneracy is a polynomial-time decidable property, so we may take the guess and check approach derived from Definition 3.3.4. Non-deterministically choose k vertices to delete, these vertices can be represented in at most $\log m$ bits (where m is the size of the input). Then apply the polynomial time greedy algorithm to verify that the remaining graph is r-degenerate.

6.3.4 Edge Deletion

When considering the edge deletion operation instead of vertex deletion, we first have to modify the AND gadget as we need to be able to remove a single edge to represent activating the gate, but still retain the property that both edges from predecessor gadgets have to be removed before the gadget can otherwise be cleared. Note that the OR gadget requires no such modification. For clarity we will refer to this new gadget as the AND *edge* gadget. It is constructed of six components, an input vertex v^i, an independent set S of $r-1$ stabilising vertices, a stabilising r-clique B, an *internal activation* vertex v^a, an independent set L of size r and an r-clique C with a designated output vertex v^o. Edges exist between the input vertex and every vertex in S, and between every vertex in S and every vertex in B. The edge between v^i and v^a will be referred to as the *activation* edge e^a. Then as before there is an edge between v^a and every vertex in L, and between every vertex in L and every vertex in C. For convenience we label v^i, S and B collectively as the *head* and the remainder of the gadget as the *tail*. Figure 6.4 gives an example for $r = 4$.

As before when properly connected v^i will have at most two additional edges from the output vertices of its predecessor gadgets, and v^o will have at most two additional edges to its successor gadgets. Then each vertex has degree at least $r+1$, and in particular v^i has degree $r+2$, thus requiring the removal of two edges to clear, v^a has degree $r+1$, requiring the removal of at least one edge to clear (with e^a being the obvious candidate) and v^o has degree at least $2r-1$, so clearing of the successor gadgets will not cause incorrect clearing of this gadget.

For convenience with OR gadgets, we will also arbitrarily designate one of the edges between v^i and L as e^a.

We are now ready to prove the following:

Theorem 6.3.5. *r-DEGENERATE(e) with $r \geq 2$ is $W[P]$-hard, and remains $W[P]$-*

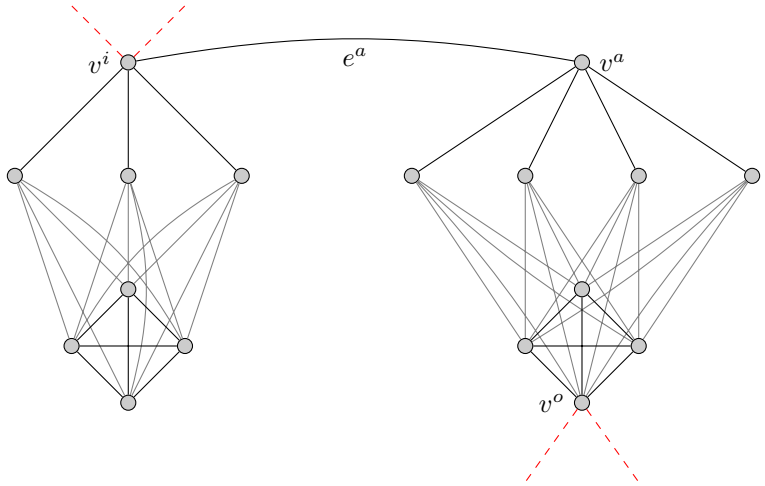


Figure 6.4: AND edge gadget for $r = 4$

hard even when the input graph is $(r + 1)$*-degenerate and has maximum degree* $2r + 1$.

Proof. The instance (G, k) is constructed as in the proof of Theorem 6.3.4 , except we now replace AND gadgets with AND edge gadgets.

Clearly OR gadgets work as before, except now we choose to delete e^a in representation of activating the corresponding gate.

The correspondence between activation and clearing works as before, we need only argue that AND edge gadgets clear correctly. Let g be an AND edge gadget with input vertex v^i, activation edge e^a and internal activation vertex v^a. If all predecessors of g are cleared then $d(v^i) = r$ so v^i may be cleared, which removes e^a, and then $d(v^a_g) = r$, so the rest of the gadget may be cleared. If g is chosen to have its activation edge e^a deleted from it, then $d(v^a_g) = r$ and the tail of the gadget will clear, however the head may not, as $d(v^i)$ may still be $r + 1$. In this case the head will clear when at least one if its predecessors clears. If G can be rendered r-degenerate by deleting at most k edges, then we are guaranteed that all predecessors will eventually be cleared, therefore the head will also clear.

Then the argument proceeds as before, except we replace the set of deleted vertices V' with a set of deleted edges E'. □

Again membership in $W[P]$ can be demonstrated by a guess and check algorithm (Section 3.3). First non-deterministically choose k edges to delete, then deterministically check that the resulting graph is r-degenerate.

6.3.5 Vertex and Edge Deletion

Corollary 6.3.6. *r-DEGENERATE(v, e) with $r \geq 2$ is $W[P]$-hard, and remains $W[P]$-hard even when the input graph is $(r+1)$-degenerate and has maximum degree $2r + 1$.*

Proof. The result follows immediately from Lemma 6.3.1. □

6.4 Conclusion

Although restricting input to degenerate graphs renders many $W[1]$-hard problems fixed-parameter tractable, editing problems are clearly not so well behaved. Furthermore as the editing problems considered here are $W[P]$-hard, a Frick and Grohe [79] style logic based approach cannot be applied unless there is some collapse in the W-hierarchy. In fact as the problems remain $W[P]$-hard on graphs of bounded degree, they can be immediately seen to be $W[P]$-hard on graphs of effectively bounded local treewidth.

It may be more fruitful to examine classes intermediate between degenerate graphs and regular graphs (or bounded degree graphs), such as *nowhere-dense* graphs, introduced by Nešetřil and Ossona de Mendez [125], for which (k, d)-CENTER is fixed-parameter tractable [48].

Chapter 7

Conclusion

The main work of this thesis is a parameterized complexity classification of a number of degree constrained editing problems. In particular we show that the WDCE(S) class of problems is in general fixed-parameter tractable when parameterized by the number k of editing steps and the bound r on the degree lists (Section 5.4), and $W[1]$-hard when parameterized by k alone (Sections 4.6 and 5.5). The problems remain $W[1]$-hard even when the degree constraints are uniformly restricted to $\{r\}$ and the graph is unweighted, if vertex deletion is one of the available editing operations. When parameterized by r alone, the problems are para-NP-complete (Section 4.1), suggesting that (k, r) is in some sense a minimal parameterization for tractability, as neither k nor r alone is sufficient. When $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}\}$ we obtain kernelizations: WDCE(S) admits a kernel of size $O(k^2 r^{k+1} + k r^{k+2})$ (Section 5.2); WDCE*(S) admits a kernel of size $O(kr(k+r))$ (Section 4.4). In the WDCE* case, the kernelization remains correct when degree constraints are uniformly restricted to $\{r\}$ and the graph is unweighted, thus giving a kernel for the $\mathrm{WDCE}_1^r(\mathsf{v}) = r$-Regular Induced Subgraph problem that improves Moser and Thilikos' [122] kernel. When vertex deletion and edge addition are allowed, we show that it is unlikely that WDCE* has either a polynomially sized kernel or a polynomial time kernelization (of a certain form), as Edge Replacement Set is a subproblem (Section 5.3). In the case where the degree lists are singletons and the only operations are edge deletion and edge addition, we show that the problem is polynomial time solvable (Section 4.5).

We also show that WDCE problems remain hard in general when parameterized by the treewidth of the input graph (Section 5.7), however when the limit on the number of editing steps is removed, the WDCE_1^r problems become fixed-parameter

tractable when parameterized by the treewidth of the input graph (Section 5.7.1).

When the degree constraint considered is degeneracy, editing problems appear to become much harder, even when r is a constant: r-DEGENERATE(S) is $W[P]$-hard when parameterized by the number k of editing steps, and remains $W[P]$-hard when the input graph is $(r+1)$-degenerate, and has bounded degree (Section 6.3), where $\emptyset \neq S \subseteq \{\mathsf{v}, \mathsf{e}\}$. This implies that the problem is $W[P]$-hard on graphs of effectively bounded local treewidth, so a logical approach as in Section 5.4 is inapplicable. Interestingly, many otherwise hard problems such as INDEPENDENT SET, DOMINATING SET and CLIQUE are fixed-parameter tractable when the input graph is degenerate. In fact INDEPENDENT SET admits a linear kernel when restricted to degenerate graphs (Section 6.1).

7.1 Future Research

Although the WDCE class of problems and the variants presented here cover a wide range of degree constrained editing problems, there are many more forms of degree constraints that are still to be examined. In particular, more general forms of degree constraints are interesting. Although the editing problem for degenerate graphs is $W[P]$-hard, there should be some further classes between regular/bounded degree graphs and degenerate graphs for which the editing problem is fixed-parameter tractable. For example nowhere-dense graphs are interesting in this sense, particularly as (k,d)-CENTERS is fixed-parameter tractable for nowhere-dense graphs, whereas it is $W[2]$-hard for degenerate graphs. Alternatively we may consider constraints that are not specified for each vertex, but for the graph as a whole, for example a list of desired degrees.

Parameter:	k				$k+r$			
Editing Operations:	{v}	{v, e}	{e}	{e, a}	{v, e, a}	{v}	{v, e}	{e}	{e, a}	{v, e, a}
WDCE$_1^r$	$W[1]$-hard[1]	$W[1]$-hard[1]	P[3]	P[3]	$W[1]$-hard[1]	FPT[4]	FPT[4]	P[3]	P[3]	FPT[4]
WDCE*	$W[1]$-hard[1]	$W[1]$-hard[1]	P[3]	P[3]	$W[1]$-hard[1]	FPT[4]	FPT[4]	P[3]	P[3]	FPT[4]
WDCE	$W[1]$-hard[1]	$W[1]$-hard[1]	$W[1]$-hard[2]	$W[1]$-hard[2]	$W[1]$-hard[1]	FPT[4]	FPT[4]	FPT[4]	FPT[4]	FPT[4]
r-DEGENERATE	$W[P]$-hard[5]	$W[P]$-hard[7]	$W[P]$-hard[6]	N/A	N/A	$W[P]$-hard[5]	$W[P]$-hard[7]	$W[P]$-hard[6]	N/A	N/A

Table 7.1: A summary of the key results of Chapters 4, 5 and 6. The results are given by the following Theorems: [1] Theorem 4.6.3, [2] Theorem 5.5.1, [3] Theorem 4.5.2, [4] Theorem 5.4.1, [5] Theorem 6.3.4, [6] Theorem 6.3.5, [7] Corollary 6.3.6.

Bibliography

[1] K. R. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed-parameter intractability II (extended abstract). In P. Enjalbert, A. Finkel, and K. W. Wagner, editors, *10th Annual Symposium on Theoretical Aspects of Computer Science (STACS'93)*, volume 665 of *Lecture Notes in Computer Science*, pages 374–385, Würzburg, Germany, 1993. Springer.

[2] K. R. Abrahamson, J. A. Ellis, M. R. Fellows, and M. E. Mata. On the complexity of fixed parameter problems (extended abstract). In *30th Annual Symposium on Foundations of Computer Science (FOCS'89)*, pages 210–215, Research Triangle Park, North Carolina, USA, 1989. IEEE Computer Soc.

[3] F. N. Abu-Khzam. Kernelization algorithms for d-hitting set problems. In F. K. H. A. Dehne, J.-R. Sack, and N. Zeh, editors, *10th International Workshop on Algorithms and Data Structures (WADS'07)*, volume 4619 of *Lecture Notes in Computer Science*, pages 434–445. Springer, 2007.

[4] F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In L. Arge, G. F. Italiano, and R. Sedgewick, editors, *Workshop on Algorithm Engineering and Experiments (ALENEX/ANALC'04)*, pages 62–69. Society for Industrial and Applied Mathematics (SIAM), 2004.

[5] F. N. Abu-Khzam and H. Fernau. Kernels: Annotated, proper and induced. In *The Second International Workshop on Parameterized and Exact Computation 2006 (IWPEC'06)*, Lecture Notes in Computer Science, pages 264–275. Springer, 2006.

[6] I. Adler, M. Grohe, and S. Kreutzer. Computing excluded minors. In *The 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'08)*, pages 641–650, 2008.

[7] N. Alon and S. Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. In *The 13th Annual International Computing and Combinatorics Conference (COCOON'07)*, volume 4598 of *Lecture Notes in Computer Science*, pages 394–405. Springer, 2007.

[8] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[9] O. Amini, I. Sau, and S. Saurabh. Parameterized complexity of the smallest degree-constrained subgraph problem. In M. Grohe and R. Niedermeier, editors, *Third International Workshop on Parameterized and Exact Computation (IWPEC'08)*, volume 5018 of *Lecture Notes in Computer Science*, pages 13–29. Springer, 2008.

[10] R. P. Anstee. An algorithmic proof of Tutte's f-factor theorem. *J. Algorithms*, 6(1):112–131, 1985.

[11] T. Asano and T. Hirata. Edge-deletion and edge-contraction problems. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (STOC'82)*, pages 245–254. ACM, 1982.

[12] T. Asano and T. Hirata. Edge-contraction problems. *J. Comput. System Sci.*, 26(2):197–208, 1983.

[13] R. Balasubramanian, M. R. Fellows, and V. Raman. An improved fixed-parameter algorithm for vertex cover. *Information Processing Letters*, 65(3):163–168, 1998.

[14] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[15] H. Bensmail, J. Golek, M. M. Moody, J. O. Semmes, and A. Haoudi. A novel approach for clustering proteomics data using bayesian fast fourier transform. *Bioinformatics*, 21(10):2210–2224, 2005.

[16] M. Bläser. Computing small partial coverings. *Information Processing Letters*, 85(6):327–331, 2003.

[17] S. Böcker, S. Briesemeister, Q. B. A. Bui, and A. Truß. A fixed-parameter approach for weighted cluster editing. In A. Brazma, S. Miyano, and

T. Akutsu, editors, *Proceedings of the 6th Asia-Pacific Bioinformatics Conference (APBC'08)*, volume 6 of *Advances in Bioinformatics and Computational Biology*, pages 211–220. Imperial College Press, 2008.

[18] S. Böcker, S. Briesemeister, Q. B. A. Bui, and A. Truß. Going weighted: Parameterized algorithms for cluster editing. In B. Yang, D.-Z. Du, and C. A. Wang, editors, *Second International Conference on Combinatorial Optimization and Applications (COCOA'08)*, volume 5165 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2008.

[19] S. Böcker, S. Briesemeister, and G. W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. In C. C. McGeoch, editor, *7th International Workshop on Experimental Algorithms (WEA'08)*, volume 5038 of *Lecture Notes in Computer Science*, pages 289–302. Springer, 2008.

[20] H. Bodlaender. Discovering treewidth. In P. Vojtáš, M. Bieliková, B. Charron-Bost, and O. Sýkora, editors, *Software Seminar (SOFSEM'05)*, volume 3381 of *Lecture Notes in Computer Science*, pages 1–16, 2005.

[21] H. Bodlaender, R. Tan, and J. van Leeuwen. Finding a $\triangle$-regular supergraph of minimum order. *Discrete Appl. Math.*, 131(1):3–9, 2003.

[22] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25:1305–1317, 1996.

[23] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels (extended abstract). In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *35th International Colloquium on Automata, Languages and Programming (ICALP'08) Part I: Tack A: Algorithms, Automata, Complexity, and Games (Extended Abstract)*, volume 5125 of *Lecture Notes in Computer Science*, pages 563–574. Springer, 2008.

[24] H. L. Bodlaender, S. Thomassé, and A. Yeo. Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels. Technical report, Utrecht University, 2008.

[25] J. A. Bondy and U. S. R. Murty. *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, 2008.

[26] J. Buss and J. Goldsmith. Nondeterminism within *P*. *SIAM J. Comput.*, 22(3):560–572, 1993.

[27] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[28] L. Cai and T. Kloks. Parameterized tractability of some (efficient) y - domination variants for planar graphs and t-degenerate graphs. In *International Computer Symposium (ICS)*, 2000.

[29] K. Cameron. Induced matchings. *Discrete Appl. Math.*, 24(1–3):97–102, 1989.

[30] D. M. Cardoso, M. Kamiński, and V. Lozin. Maximum k-regular induced subgraphs. *J. Comb. Optim.*, 14(4):455–463, 2007.

[31] L. S. Chandran and F. Grandoni. Refined memorisation for vertex cover. *Information Processing Letters*, 93(3):125–131, 2005.

[32] G. Chartrand and L. Lesniak. *Graphs and Digraphs*. Chapman and Hall/CRC, fourth edition, 2004.

[33] F. Cheah and D. G. Corneil. The complexity of regular subgraph recognition. *Discrete Appl. Math.*, 27(1–2):59–68, 1990.

[34] J. Chen, I. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41:280–301, 2001.

[35] J. Chen, I. Kanj, and G. Xia. Simplicity is beauty: Improved upper bounds for vertex cover. Technical report, School of CTI, DePaul University, 2005.

[36] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In R. E. Ladner and C. Dwork, editors, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC'08)*, pages 177–186. Assoc. Comput. Mach., New York, 2008.

[37] Y. Chen and J. Flum. Machine characterization of the classes of the W-hierarchy. In M. Baaz and J. A. Makowsky, editors, *17th International Workshop on Computer Science Logic (CSL'03)*, volume 2803 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2003.

[38] Y. Chen, J. Flum, and M. Grohe. Machine-based methods in parameterized complexity theory. *Theoret. Comput. Sci.*, 339(2–3):167–199, 2005.

[39] B. Chor, M. Fellows, and D. W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In J. Hromkovic, M. Nagl, and B. Westfechtel,

editors, *30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'04)*, volume 3353 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2004.

[40] C. J. Colbourn and E. S. El-Mallah. The complexity of some edge deletion problems. *IEEE Transactions on Circuits and Systems*, 35(3):354–362, 1988.

[41] S. A. Cook and P. McKenzie. Problems complete for deterministic logarithmic space. *J. Algorithms*, 8(3):385–394, 1987.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill, 2^{nd} edition, 2001.

[43] G. Cornuéjols. General factors of graphs. *J. Combin. Theory Ser. B*, 45(2):185–198, 1988.

[44] B. Courcelle. The monadic second order logic of graphs, I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[45] B. Courcelle. The monadic second-order logic of graphs XVI: Canonical graph decompositions. *Logical Methods in Computer Science*, 2:1–46, 2006.

[46] P. Damaschke. On the fixed-parameter enumerability of cluster editing. In D. Kratsch, editor, *31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG'05)*, volume 3787 of *Lecture Notes in Computer Science*, pages 283–294. Springer, 2005.

[47] P. Damaschke. Fixed-parameter tractable generalizations of cluster editing. In T. Calamoneri, I. Finocchi, and G. F. Italiano, editors, *6th Italian Conference on Algorithms and Complexity (CIAC'06)*, volume 3998 of *Lecture Notes in Computer Science*, pages 344–355. Springer, 2006.

[48] A. Dawar and S. Kreutzer. Dominating sets and network centres in nowhere-dense classes of graphs. Unpublished Manuscript, 2008.

[49] F. Dehne, M. R. Fellows, M. Langston, F. A. Rosamond, and K. Stevens. An $\mathcal{O}^*(2^{O(k)})$ FPT algorithm for the undirected feedback vertex set problem. In *Proceedings of the 11th COCOON*, volume 3595 of *Lecture Notes in Computer Science*, pages 859–869. Springer, 2005.

[50] F. Dehne, M. R. Fellows, F. A. Rosamond, and P. Shaw. Greedy localization, iterative compression, and modeled crown reductions: New FPT techniques,

and improved algorithm for set splitting, and a novel $2k$ kernelization for vertex cover. In *The First International Workshop on Parameterized and Exact Computation (IWPEC'04)*, volume 3162 of *Lecture Notes in Computer Science*, pages 271–280. Springer, 2004.

[51] F. Dehne, M. Langston, X. Luo, S. Pitre, P. Shaw, and Y. Zhang. The cluster editing problem: Implementations and experiments. In *The Second International Workshop on Parameterized and Exact Computation (IWPEC'06)*, volume 4169 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2006.

[52] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

[53] R. Diestel. *Graph Theory.* Springer, 1997.

[54] R. Downey, M. R. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, volume 49 of *AMS-DIMACS*, pages 49–99. American Mathematical Society, 1999.

[55] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. *Congr. Numer.*, 87:161–187, 1992.

[56] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.*, 24(4):873–921, 1995.

[57] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoret. Comput. Sci.*, 141(1&2):109–131, 1995.

[58] R. G. Downey and M. R. Fellows. Parameterized computational feasibility. In P. Clote and J. Remmel, editors, *Feasible Mathematics*, volume II, pages 219–244, 1995.

[59] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[60] R. G. Downey, M. R. Fellows, and K. W. Regan. Descriptive complexity and the W-hierarchy. In P. Beame and S. Buss, editors, *Proof Complexity*

and Feasible Arithmetic, volume 39 of *AMS-DIMACS Volume Series*, pages 119–134. American Mathematical Society, 1998.

[61] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[62] H. D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, second edition, 1999.

[63] H. D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer, second edition, 1996.

[64] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, 69B:125–130, 1965.

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

[66] J. Edmonds and E. Johnson. Matchings: A well solved class of integer linear programs. In *Combinatorial Structures and their Applications*, pages 89–92, 1970.

[67] J. Ellis, H. Fan, and M. R. Fellows. The dominating set problem is fixed parameter tractable for graphs of bounded genus. *J. Algorithms*, 52(2):152–168, 2004.

[68] M. R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In H. L. Bodlaender, editor, *29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'03)*, volume 2880 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003.

[69] M. R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In H. L. Bodlaender and M. A. Langston, editors, *Second International Workshop on Parameterized and Exact Computation (IWPEC'06)*, volume 4169 of *Lecture Notes in Computer Science*, pages 276–277. Springer, 2006.

[70] M. R. Fellows, J. Guo, H. Moser, and R. Niedermeier. A generalization of Nemhauser and Trotter's local optimization theorem. In S. Albers and J.-Y. Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science (STACS'09)*, pages 409–420, 2009.

[71] M. R. Fellows, P. Heggernes, F. A. Rosamond, C. Sloper, and J. A. Telle. Finding k disjoint triangles in an arbitrary graph. In *30th International Workshop*

on Graph-Theoretic Concepts in Computer Science (WG'04), volume 3353 of *Lecture Notes in Computer Science*, pages 235–244. Springer, 2004.

[72] M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoret. Comput. Sci.*, 410(1):53–61, 2009.

[73] M. R. Fellows and M. A. Langston. Nonconstructive advances in polynomial-time complexity. *Information Processing Letters*, 26(3):155–162, 1987.

[74] M. R. Fellows, M. A. Langston, F. A. Rosamond, and P. Shaw. Efficient parameterized preprocessing for cluster editing. In E. Csuhaj-Varjú and Z. Ésik, editors, *16th International Symposium on the Fundamentals of Computation Theory (FCT'07)*, volume 4639 of *Lecture Notes in Computer Science*, pages 312–321. Springer, 2007.

[75] H. Fernau. *Parameterized Algorithmics: A Graph-Theoretic Approach*. Habilitationsschrift, Universität Tübingen, Germany, 2005.

[76] J. Flum and M. Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

[77] J. Flum and M. Grohe. Model-checking problems as a basis for parameterized intractability. *Logical Methods in Computer Science*, 1(1), 2005.

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

[79] M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. *J. ACM*, 48(6):1184–1206, 2001.

[80] 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.

[81] B. Gerards. Matching. In *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 135–224. North-Holland Publishing Co., 1995.

[82] P. Golovach and Y. Villanger. Parameterized complexity for domination problems on degenerate graphs. In H. Broersma, T. Erlebach, T. Friedetzky, and D. Paulusma, editors, *34th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'08)*, volume 5344 of *Lecture Notes in Computer Science*, pages 195–205. Springer, 2008.

[83] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory Comput. Syst.*, 38(4):373–392, 2005.

[84] J. Guo. Fixed-parameter algorithms for graph-modeled date clustering. In J. Chen and S. B. Cooper, editors, *6th Annual Conference on the Theory and Applications of Models of Computation*, volume 5532 of *Lecture Notes in Computer Science*, pages 39–48. Springer, 2009.

[85] J. Guo. A more effective linear kernelization for cluster editing. *Theoret. Comput. Sci.*, 410(8–10):718–726, 2009.

[86] J. Guo, F. Hüffner, R. Niedermeier, and S. Wernicke. Improved fixed-parameter algorithms for two feedback set problems. In *Ninth Workshop on Algorithms and Data Structures (WADS'05)*, volume 3608 of *Lecture Notes in Computer Science*, pages 158–168. Springer, 2005.

[87] J. Guo and R. Niedermeier. A fixed-parameter tractable algorithm for multi-commodity demand flow in trees. *Information Processing Letters*, 97:109–114, 2006.

[88] P. Heggernes, C. Paul, J. A. Telle, and Y. Villanger. Interval completion with few edges. In D. S. Johnson and U. Feige, editors, *39th Annual ACM Symposium on Theory of Computing (STOC'07)*, pages 374–381. Assoc. Comput. Mach., New York, 2007.

[89] P. Hell and D. G. Kirkpatrick. On the completeness of a generalized matching problem. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 240–245, 1978.

[90] F. Hüffner. Algorithm engineering for optimal graph bipartizartion. In *Fourth International Workshop on Experimental and Efficient Algorithms (WEA'05)*, volume 3503 of *Lecture Notes in Computer Science*, pages 240–252. Springer, 2005.

[91] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. In E. S. Laber, C. F. Bornstein, L. T. Nogueira, and L. Faria, editors, *8th Latin American Symposium on Theoretical Informatics (LATIN'08)*, volume 4957 of *Lecture Notes in Computer Science*, pages 711–722. Springer, 2008. Journal Version to Appear in Theory of Computing Systems.

[92] T. M. Islam. *Characterizing Hardness in Parameterized Complexity*. PhD thesis, University of Waterloo, 2007.

[93] S. Khot and V. Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoret. Comput. Sci.*, 289(2):997–1008, 2002.

[94] B. Korte and J. Vygen. *Combinatorial Optimization*, volume 21 of *Algorithms and Combinatorics*. Springer, Berlin, fourth edition, 2008.

[95] A. V. Kostochka. Lower bound of the Hadwiger number of graphs by their average degree. *Combinatorica*, 4(4):307–316, 1984.

[96] S. Kratsch and M. Wahlström. Two edge modification problems without polynomial kernels. In *The Fourth International Workshop on Parameterized and Exact Computation (IWPEC'09)*, 2009. To appear.

[97] M. S. Krishnamoorthy and N. Deo. Node-deletion NP-complete problems. *SIAM J. Comput.*, 8(4):619–625, 1979.

[98] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(2):83–97, 1955.

[99] H. W. Kuhn. Variants of the Hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3(4):253–258, 1956.

[100] J. M. Lewis. On the complexity of the maximum subgraph problem. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 265–274. Assoc. Comput. Mach., New York, 1978.

[101] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. System Sci.*, 20(2):219–230, 1980.

[102] D. R. Lick and A. T. White. k-degenerate graphs. *Canad. J. Math.*, 22:1082–1096, 1970.

[103] L. Lin and S. Sahni. Fair edge deletion problems. *IEEE Transactions on Computers*, 38(5):756–761, 1989.

[104] G. Liu and B. Zhu. Some problems on factorizations with constraints in bipartite graphs. *Discrete Appl. Math.*, 128(2–3):421–434, 2003.

[105] Z. Liu, D. Chen, H. Bensmail, and Y. Xu. Clustering gene expression data with kernel principal components. *J. Bioinformatics and Computational Biology*, 3(2):303–316, 2005.

[106] L. Lovász. The factorization of graphs. In *Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969)*, pages 243–246. Gordon and Breach, New York, 1970.

[107] L. Lovász. The factorization of graphs. II. *Acta Math. Acad. Sci. Hungar.*, 23:223–246, 1972.

[108] L. Lovász. Antifactors of graphs. *Periodica Mathematica Hungarica*, 4(2):121–123, 1973.

[109] L. Lovász and M. D. Plummer. *Matching Theory (North-Holland mathematics studies)*. Elsevier Science Publishers, North-Holland, 1986.

[110] J. MacCuish, C. Nicolaou, and N. E. MacCuish. Ties in proximity and clustering compounds. *Journal of Chemical Information and Computer Sciences*, 41(1):134–146, 2001.

[111] D. Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2):153–183, 2005.

[112] D. Marx. Chordal deletion is fixed-parameter tractable. In F. V. Fomin, editor, *32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'06)*, volume 4271 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2006.

[113] D. Marx and I. Schlotter. Obtaining a planar graph by vertex deletion. In A. Brandstädt, D. Kratsch, and H. Müller, editors, *33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'07)*, volume 4769 of *Lecture Notes in Computer Science*, pages 292–303. Springer, 2007.

[114] D. Marx and I. Schlotter. Parameterized graph cleaning problems. In H. Broersma, T. Erlebach, T. Friedetzky, and D. Paulusma, editors, *34th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'08)*, volume 5344 of *Lecture Notes in Computer Science*, pages 287–299. Springer, 2008.

[115] L. Mathieson. The parameterized complexity of editing graphs for bounded degeneracy. 2009. Submitted.

[116] L. Mathieson, E. Prieto, and P. Shaw. Packing edge disjoint triangles: A parameterized view. In F. Dehne, R. Downey, and M. Fellows, editors, *Proceedings of the 1st International Workshop on Parameterized and Exact Com-*

putation (IWPEC'04), volume 3162 of *Lecture Notes in Computer Science*, pages 127–137. Springer, 2004.

[117] L. Mathieson and S. Szeider. The parameterized complexity of regular subgraph problems and generalizations. In J. Harland and P. Manyem, editors, *Fourteenth Computing: The Australasian Theory Symposium (CATS'08)*, volume 77 of *CRPIT*, pages 79–86, Wollongong, NSW, Australia, 2008. ACS.

[118] L. Mathieson and S. Szeider. Parameterized graph editing with chosen vertex degrees. In B. Yang, D. Du, and C. Wang, editors, *Second Annual International Conference on Combinatorial Optimization and Applications (COCOA'08)*, volume 5165 of *Lecture Notes in Computer Science*, pages 13–22, St John's, Canada, 2008. Springer.

[119] L. Mathieson and S. Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. 2009. Submitted.

[120] D. Mölle, S. Richter, and P. Rossmanith. A faster algorithm for the Steiner tree problem. Technical report, Department of Computer Science, RWTH Aachen, 2005.

[121] H. Moser and D. Thilikos. Parameterized complexity of finding regular induced subgraphs. In *Algorithms and Complexity in Durham 2006 (ACiD06)*, Texts in Algorithmics, pages 107–118. College Publications, 2006. A full and updated version of the paper is to appear in the J. of Discrete Algorithms.

[122] H. Moser and D. Thilikos. Parameterized complexity of finding regular induced subgraphs. *J. Discrete Algorithms*, 2008. Article in Press.

[123] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Appl. Math.*, 113(1):109–128, 2001.

[124] G. L. Nemhauser and L. E. T. Jr. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

[125] J. Nešetřil and P. O. de Mendez. First order properties on nowhere dense structures. Unpublished Manuscript, 2008.

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

[127] R. Niedermeier and P. Rossmanith. Upper bounds for vertex cover further improved. In *Proceedings of the 16th STACS*, volume 1563 of *Lecture Notes in Computer Science*, pages 561–570. Springer, 1999.

[128] R. Niedermeier and P. Rossmanith. An efficient fixed-parameter algorithm for 3-hitting set. *J. Discrete Algorithms*, 1:89–102, 2003.

[129] R. Niedermeier and P. Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *J. Algorithms*, 47(2):63–77, 2003.

[130] N. Nishimura, P. Ragde, and D. M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. *Discrete Appl. Math.*, 152(1–3):229–245, 2005.

[131] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[132] C. H. Papadimitriou and M. Yannakakis. On the complexity of database queries. *J. Comput. System Sci.*, 58(3):407–427, 1999.

[133] K. Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. System Sci.*, 67(4):757–771, 2003.

[134] J. Plesník. A note on the complexity of finding regular subgraphs. *Discrete Math.*, 49(2):161–167, 1984.

[135] E. Prieto. *Systematic Kernelization in FPT Algorithm Design*. PhD thesis, University of Newcastle, Australia, 2005.

[136] E. Prieto and C. Sloper. Looking at the stars. *Theoret. Comput. Sci.*, 351(3):437–445, 2006.

[137] F. Protti, M. D. da Silva, and J. L. Szwarcfiter. Applying modular decomposition to parameterized bicluster editing. In H. L. Bodlaender and M. A. Langston, editors, *Second International Workshop on Parameterized and Exact Computation (IWPEC'06)*, volume 4169 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[138] S. Rahmann, T. Wittkop, J. Baumbach, M. Martin, A. Truss, and S. Böcker. Exact and heuristic algorithms for weighted cluster editing. In *Computational Systems Bioinformatics (CSB'07)*, volume 6, pages 391–401, 2007.

[139] V. Raman, S. Saurabh, and S. Srihari. Parameterized algorithms for generalized domination. In *Proceedings of Combinatorial Optimization and Applications, 2nd International Conference (COCOA '08)*, volume 5165 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008.

[140] I. Razgon. Parameterized dfvs and multicut problems on dags. In *Proceedings of the 3rd Algorithms and Complexity in Durham Workshop (ACiD'07)*, volume 9 of *Texts in Algorithmics*, pages 119–128. College Publications, 2007.

[141] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

[142] B. A. Reed. Finding approximate separators and computing tree width quickly. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 221–228. Assoc. Comput. Mach., New York, 1992.

[143] N. Robertson and P. D. Seymour. Graph minors. I. Excluding a forest. *J. Combin. Theory Ser. B*, 35(1):39–61, 1983.

[144] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Combin. Theory Ser. B*, 92(2):325–357, 2004.

[145] M. Samer and S. Szeider. Tractable cases of the extended global cardinality constraint. In *Computing: The Australasian Theory Symposium (CATS'08)*, volume 77 of *CRPIT*, pages 67–74. ACS, 2008.

[146] A. A. Schoone, H. L. Bodlaender, and J. van Leeuwen. Diameter increase caused by edge deletion. *J. Graph Theory*, 11(3):409–427, 1987.

[147] J. Scott, T. Ideker, R. M. Karp, and R. Sharan. Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology*, 13(2):133–144, 2006.

[148] A. Sebő. General antifactors of graphs. *J. Combin. Theory Ser. B*, 58(2):174–184, 1993.

[149] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Appl. Math.*, 144(1–2):173–182, 2004.

[150] Y. Shiloach. Another look at the degree constrained subgraph problem. *Information Processing Letters*, 12(2):89–92, 1981.

[151] I. A. Stewart. Deciding whether a planar graph has a cubic subgraph is NP-complete. *Discrete Math.*, 126(1–3):349–357, 1994.

[152] I. A. Stewart. Finding regular subgraphs in both arbitrary and planar graphs. *Discrete Appl. Math.*, 68(3):223–235, 1996.

[153] I. A. Stewart. On locating cubic subgraphs in bounded-degree connected bipartite graphs. *Discrete Math.*, 163(1–3):319–324, 1997.

[154] I. A. Stewart. On the fixed-parameter tractability of parameterized model-checking problems. *Information Processing Letters*, 106:33–36, 2008.

[155] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoret. Comput. Sci.*, 3(1):1–22, 1976.

[156] L. J. Stockmeyer and V. V. Vazirani. NP-completeness of some generalizations of the maximum matching problem. *Information Processing Letters*, 15(1):14–19, 1982.

[157] S. Szeider. Backdoor sets for DLL subsolvers. *Journal of Automated Reasoning*, 35(1–3):73–88, 2005. Reprinted as Chapter 4 of the book SAT 2005 - Satisfiability Research in the Year 2005, edited by E. Giunchiglia and T. Walsh, Springer-Verlag, 2006.

[158] D. K. Tasoulis, G. J. Ross, and N. M. Adams. Visualising the cluster structure of data streams. In M. R. Berthold, J. Shawe-Taylor, and N. Lavrac, editors, *Advances in Intelligent Data Analysis VII, 7th International Symposium on Intelligent Data Analysis, (IDA'07)*, volume 4723 of *Lecture Notes in Computer Science*, pages 81–92. Springer, 2007.

[159] A. Thomason. The extremal function for complete minors. *J. Combin. Theory Ser. B*, 81(2):318–338, 2001.

[160] W. T. Tutte. A short proof of the factor theorem for finite graphs. *Canad. J. Math.*, 6:347–352, 1954.

[161] W. T. Tutte. Spanning subgraphs with specified valencies. *Discrete Math.*, 9(1):97–108, 1974.

[162] W. T. Tutte. Graph factors. *Combinatorica*, 1(1):79–97, 1981.

[163] R. J. Urquhart. *Degree Constrained Subgraphs of Linear Graphs*. PhD thesis, University of Michigan, Ann Arbor, USA, 1967.

[164] H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer, 1999.

[165] T. Watanabe, T. Ae, and A. Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discrete Appl. Math.*, 3(2):151–153, 1981.

[166] T. Watanabe, T. Ae, and A. Nakamura. On the NP-hardness of edge-deletion and -contraction problems. *Discrete Appl. Math.*, 6(1):63–78, 1983.

[167] D. B. West. *Introduction to Graph Theory*. Prentice Hall, second edition, 2001.

[168] M. Yannakakis. Node- and edge-deletion NP-complete problems. In *The Tenth Annual ACM Symposium on the Theory of Computing (STOC'78)*, pages 253–264. Assoc. Comput. Mach., New York, 1978.

[169] M. Yannakakis. Edge-deletion problems. *SIAM J. Comput.*, 10(2):297–309, 1981.

[170] M. Yannakakis. Node-deletion problems on bipartite graphs. *SIAM J. Comput.*, 10(2):310–327, 1981.

Index