Foundations of computing Volume 4

Nimrod Talmon

Algorithmic Aspects of Manipulation and Anonymization in Social Choice and Social Networks

Universitätsverlag der TU Berlin

Technische Universität Berlin

Nimrod Talmon

Algorithmic Aspects of Manipulation and Anonymization in Social Choice and Social Networks

Die Schriftenreihe *Foundations of Computing* der Technischen Universität Berlin wird herausgegeben von:
Prof. Dr. Stephan Kreutzer,
Prof. Dr. Uwe Nestmann,
Prof. Dr. Rolf Niedermeier

Foundations of Computing | 04

Nimrod Talmon

Algorithmic Aspects of Manipulation and Anonymization in Social Choice and Social Networks

Universitätsverlag der TU Berlin

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Universitätsverlag der TU Berlin, 2016
http://verlag.tu-berlin.de

Fasanenstr. 88, 10623 Berlin
Tel.: +49 (0)30 314 76131 / Fax: -76133
E-Mail: publikationen@ub.tu-berlin.de

Zugl.: Berlin, Technische Universität, Diss., 2015
1. Gutachter: Prof. Dr. Rolf Niedermeier
2. Gutachter: Prof. Jeffrey S. Rosenschein
3. Gutachter: Prof. Hadas Shachnai
Die Arbeit wurde am 14. Oktober 2015 an der Fakultät IV unter Vorsitz von Prof. Dr. Marc Alexa erfolgreich verteidigt.

Druck: docupoint GmbH
Satz/Layout: Nimrod Talmon

Umschlagfoto:
Hans Braxmeier | https://pixabay.com/en/euphonium-brass-instrument-93867 | CC0
https://creativecommons.org/publicdomain/zero/1.0

ISBN 978-3-7983-2804-4 (print)
ISBN 978-3-7983-2805-1 (online)

ISSN 2199-5249 (print)
ISSN 2199-5257 (online)

Zugleich online veröffentlicht auf dem institutionellen Repositorium der Technischen Universität Berlin:
DOI 10.14279/depositonce-4941
http://dx.doi.org/10.14279/depositonce-4941

Zusammenfassung

Diese Dissertation stellt eine Untersuchung von verschiedenen kombinatorischen Problemen im Umfeld von Wahlen und sozialen Netzwerken dar. Das Hauptziel ist die Analyse der Berechnungskomplexität mit dem Schwerpunkt auf der parametrisierten Komplexität. Dabei werden für jedes der untersuchten Probleme effiziente Algorithmen entworfen oder aber gezeigt, dass unter weit akzeptierten Annahmen solche Algorithmen nicht existieren können.

Die Probleme, welche im Kapitel 3 und im Kapitel 4 diskutiert werden, modellieren das Manipulieren einer gegebenen Wahl, bei welcher gewisse Beziehungen zwischen den Beteiligten angenommen werden. Dies kann so interpretiert werden, dass die Wahl innerhalb eines Sozialen Netzwerks stattfindet, in dem die Wähler oder die Kandidaten miteinander in Verbindung stehen.

Das Problem COMBINATORIAL CANDIDATE CONTROL, welches in Kapitel 3 untersucht wird, folgt der Arbeit von Chen et al. [42], und handelt von der Manipulation einer Wahl durch die änderung der Kandidatenmenge über welche die Wähler abstimmen. Genauer gesagt, gibt es einen externen Agenten, welcher neue Kandidaten hinzufügen oder existierende Kandidaten entfernen kann. Es wird eine kombinatorische Struktur über der Kandidatenmenge angenommen, so dass immer wenn der externe Agent einen Kandidaten hinzufügt oder entfernt, eine vordefinierte Kandidatenmenge (welche mit den ausgewählten Kandidaten in Beziehung steht) ebenfalls hinzugefügt bzw. entfernt wird.

Das Problem COMBINATORIAL SHIFT BRIBERY, welches in Kapitel 4 untersucht wird, folgt der Arbeit von Bredereck et al. [31], und thematisiert ebenfalls die Manipulation einer Wahl. Hier allerdings kann der externe Agent Änderungen des Abstimmungsverhaltens einiger Wähler herbeiführen. Dabei wird eine kombinatorische Struktur über den Wählern angenommen, so dass der externe Agent die Position des von ihm präferierten Kandidaten bei mehreren Wählern entsprechend vordefinierter Muster gleichzeitig ändern kann.

Das Problem ELECTION ANONYMIZATION, welches in Kapitel 5 untersucht wird, folgt der Arbeit von Talmon [139], und befasst sich ebenso mit Wahlen. Das Hauptanliegen hier ist es jedoch, die Privatsphäre der Wähler bei der Veröffentlichung der Stimmenabgaben zusammen mit einigen zusätzlichen (privaten) Informationen aufrecht zu erhalten. Die Aufgabe ist es eine gegebene Wahl so zu verändern, dass jede Stimmenabgabe mindestens k-fach vorkommt. Dadurch kann noch nicht einmal ein Gegenspieler einzelne Wähler identifizieren, wenn er die Stimmenabgaben einiger Wähler bereits kennt.

Die in Kapitel 6 und 7 untersuchten Probleme behandeln gleichermaßen Privatsphärenaspekte. Präziser gesagt, geht es darum, dass ein soziales Netzwerk (modelliert als Graph) veröffentlicht werden soll. Die Aufgabe ist es den Graphen zu anonymisieren; dies bedeutet man verändert den Graphen, so dass es für jeden Knoten mindestens $k-1$ weitere Knoten mit dem selben Grad gibt. Dadurch wird erreicht, dass selbst ein Gegenspieler, welcher die Knotengrade einiger Knoten kennt, nicht in der Lage ist einzelne Knoten zu identifizieren.

Bei dem Problem DEGREE ANONYMIZATION BY VERTEX ADDITION, welches in Kapitel 6 untersucht wird, und der Arbeit von Bredereck et al. [28] folgt, wird Anonymität durch Einführung neuer Knoten erreicht. Bei dem Problem DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS, welches in Kapitel 7 untersucht wird, und der Arbeit von Hartung and Talmon [90] folgt, wird Anonymität durch die Kontraktion von möglichst wenigen Kanten erreicht.

Das Hauptanliegen dieser Dissertation in Bezug auf die obig genannten Probleme ist es die Grenzen der effizienten Lösbarkeit auszuloten. Insbesondere da die meisten dieser Probleme berechnungsschwer (genauer NP-schwer bzw. sogar schwer zu approximieren) sind, werden einige eingeschränkte Fälle und Parametrisierungen der Probleme betrachtet. Das Ziel ist es effiziente Algorithmen für sie zu entwickeln, welche in Polynomzeit laufen, wenn einige Parameter konstante Werte aufweisen, oder besser noch zu zeigen, dass die Probleme "fixed-parameter tractable" für die betrachteten Parameter sind. Wenn solche Algorithmen nicht gefunden werden können, dann ist es das Ziel zu beweisen, dass diese Probleme tatsächlich nicht "fixed-parameter tractable" bezüglich der entsprechenden Parameter sind, oder noch besser zu zeigen, dass die Probleme NP-schwer sind, sogar wenn die entsprechenden Parameter konstante Werte aufweisen.

Abstract

This thesis presents a study of several combinatorial problems related to social choice and social networks. The main concern is their computational complexity, with an emphasis on their parameterized complexity. The goal is to devise efficient algorithms for each of the problems studied here, or to prove that, under widely-accepted assumptions, such algorithms cannot exist.

The problems discussed in Chapter 3 and in Chapter 4 are about manipulating a given election, where some relationships between the entities of the election are assumed. This can be seen as if the election occurs on top of an underlying social network, connecting the voters participating in the election or the candidates which the voters vote on.

The problem discussed in Chapter 3, COMBINATORIAL CANDIDATE CONTROL, following the paper by Chen et al. [42], is about manipulating an election by changing the set of candidates which the voters vote on. That is, there is an external agent who can add new candidates or delete existing candidates. A combinatorial structure over the candidates is assumed, such that whenever the external agent adds or removes a candidate, a predefined set of candidates (related to the chosen candidate) are added or removed from the election.

The problem discussed in Chapter 4, COMBINATORIAL SHIFT BRIBERY, following the paper by Bredereck et al. [31], is also about manipulating an election. Here, however, the external agent can change the way some voters vote. Specifically, a combinatorial structure over the voters is assumed, such that the external agent can change the position of its preferred candidate in sets of voters, following some predefined patterns.

The problem discussed in Chapter 5, ELECTION ANONYMIZATION, following the paper by Talmon [139], is also about elections. The main concern here, however, is preserving the privacy of the voters, when the votes are published, along with some additional (private) information. The task is to transform a given election such that each vote would appear at least k times. By doing so, even an adversary which knows how some voters vote, cannot identify individual voters.

The problems discussed in Chapter 6 and in Chapter 7 are also about privacy. Specifically, a social network (modeled as a graph) is to become publicly available. the task is to anonymize the graph; that is, to transform the graph such that, for every vertex, there will be at least $k-1$ other vertices with the same degree. By doing so, even an adversary which knows the degrees of some vertices cannot identify individual vertices. In the problem discussed in Chapter 6, DEGREE ANONYMIZATION BY VERTEX ADDITION, following the paper by Bredereck et al. [28], the way to achieve anonymity is by introducing new vertices. In the problem discussed in Chapter 7, DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS, following the paper by Hartung and Talmon [90], the way to achieve anonymity is by contracting as few edges as possible.

The main aim of this thesis, considering the problems mentioned above, is to explore some boundaries between tractability and intractability. Specifically, as most of these problems are computationally intractable (that is, NP-hard or even hard to approximate), some restricted cases and parameterizations for these problems are considered. The goal is to devise efficient algorithms for them, running in polynomial-time when some parameters are assumed to be constant, or, even better, to show that the problems are fixed-parameter tractable for the parameters considered. If such algorithms cannot be devised, then the goal is to prove that these problems are indeed not fixed-parameter tractable with respect to some parameters, or, even better, to show that the problems are NP-hard even when some parameters are assumed to be constant.

Preface

This thesis summarizes some of the research I did at TU Berlin from November 2013 to July 2015, while being part of the group led by Prof. Rolf Niedermeier.

I begin with mentioning the research which is not included in this thesis, and then provide a more elaborate description of the research which is included in this thesis.

Manipulating Elections. I worked on the COMBINATORIAL VOTER CONTROL problem, which is related to the problem studied in Chapter 3, where there is a combinatorial structure over the voters in a given election and an external agent can add or delete sets of voters. This work was presented at *the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS '14)* [41] and its extended version is published in *Theoretical Computer Science* [33]. I solved some open problems regarding control and bribery for Approval voting. The corresponding paper is to be published in *Information Processing Letter* [21]. I also studied election control problems for elections with few candidates and weighted voters, a work which is to be presented at *the 4th International Conference on Algorithmic Decision Theory (ADT '15)* [30].

Diffusion in Social Networks. I was interested (and still am, to some extent) in understanding the way information diffuses in social networks. The first work in this direction was about the EFFECTORS problem, where the goal is to explain a given state of a social network, assuming a specific diffusion process. The corresponding paper was presented at *the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC '15)* [34] and was invited to a special issue of *Information and Computation*.

The second work in this direction was to identify for which graph classes a Nash-equilibrium exists, with respect to a diffusion process which is modeled as a game, played over the vertices of a social network. This work was presented at *the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC '15)* [35].

Further Work. I studied approximation algorithms and the parameterized complexity of an interval multicover problem, a work which is published in *Information Processing Letters* [13]. I studied the parameterized complexity of a competitive scheduling problem, and the corresponding paper was presented at *10th International Symposium on Parameterized and Exact Computation* [98].

Next, I will briefly describe the history of the papers that did find their way into this thesis. As most of them are fruits of collaborative effort, I will specifically state my own contributions to them.

History of Chapter 6. When I arrived to Berlin, I read some of the papers published by the group members, and the series of papers considering anonymization caught my attention. Since some work have been done on anonymization by edge addition and vertex deletion, it was natural to study degree anonymization by vertex addition. I am responsible mainly for the hardness results for the property-preserving vertex addition variant, for some of the cases which are polynomial-time solvable, and, most importantly, for the fixed-parameter tractability result with respect to the solution size and the maximum degree and the fixed-parameter tractability result with respect to the anonymity level and the maximum degree. This research resulted in a paper which was presented at *the 10th International Conference on Algorithmic Aspects of Information and Management (AAIM '14)* [28] and is published in a special edition of *Theoretical Computer Science* [32].

History of Chapter 7. I was still interested in degree anonymization, and remembered that, while we were working on degree anonymization by vertex addition, the problem of degree anonymization by graph contractions was mentioned. I started working on this problem by myself, for a while. I found out that, in terms of its computational complexity, degree anonymization by graph contractions is harder than degree anonymization by vertex addition. Then, I visited Sepp Hartung, who was during a postdoctoral stay at the University of British Columbia, Vancouver, Canada. I presented my results to him, and he had a nice idea for showing fixed-parameter tractability when parameterizing by the maximum degree and the anonymity level. Back in Berlin, I continued studying this problem, a study that resulted in a paper which was presented at *the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC '15)* [90]. The journal version, which was invited to a special issue of *Information and Computation*, is currently under review.

History of Chapter 3. After writing the paper concerning combinatorial voter control in elections [41], which introduced a specific concept of manipulating elections over combinatorial structures, I was visiting Prof. Piotr Faliszewski at the AGH University of Science and Technology, Krakow, Poland. It was natural to consider the COMBINATORIAL CANDIDATE CONTROL problem. Major parts of the work concerning this problem I did together with Piotr Faliszewski, and when I came back to Berlin, Jiehua Chen joined the project and proved some further results. This research was summarized in a paper which was presented at *the 28th AAAI Conference on Artificial Intelligence (AAAI '15)* [42]. The journal version is currently under review.

History of Chapter 4. The concept of combinatorial control was still of interest to me, so I was happy when Prof. Piotr Faliszewski came up with a nice model for the COMBINATORIAL SHIFT BRIBERY problem. I liked the model so I started studying it, and ended up with proving most of the results appearing in the corresponding paper, which was presented at *the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*. The co-authors edited the paper, polished the proofs, and added some more results to it. The journal version is to be published in Journal of Artificial Intelligence Research [22].

History of Chapter 5. I was still thinking about problems related to voting and social choice when I attended the Ph.D. defense of André Nichterlein, big part of which was about anonymizing data related to social networks. I thought that it would make sense to anonymize data related to elections. I formulated the ELECTION ANONYMIZATION problem and studied it. This study resulted in a paper which was presented at *the 20th International Symposium on Fundamentals of Computation Theory (FCT '15)* [139]. The journal version is currently under review.

Acknowledgments. I would like to thank Rolf Niedermeier for providing guidance and good atmosphere for research. I would like to thank Piotr Faliszewski for providing some more guidance and for making me enthusiastic about computational social choice.

I would like to thank my coauthors: René van Bevern, Robert Bredereck, Laurent Bulteau, Jiehua Chen, Stefan Fafianie, Vincent Froese, Sepp Hartung, Danny Hermelin, Falk Hüffner, Stefan Kratsch, Judith Kubitza, André Nichterlein, Dvir Shabtay, Piotr Skowron, and Gerhard J. Woeginger.

To Avital and Lotem and Ben and to all my family

Contents

1. Introduction

We begin, in Section 1.1, by providing an intuitive explanation for what this thesis is about. This section assumes no prior knowledge about algorithms and computational complexity, and can be safely skipped by readers familiar with the basics of computer science.

Then, in Section 1.2, we provide a more formal overview of the computational problems considered in this thesis, and explain our specific point of view when studying these problems.

1.1. Gentle Introduction

This thesis is concerned with the computational complexity study and the parameterized complexity study of some computational problems related to social networks. Section 1.1.1 explains what are computational problems and what is computational complexity. Then, Section 1.1.2 deals with computational intractability and explains how parameterized complexity helps to better classify hard computational problems. Finally, Section 1.1.3 describes the specific computational problems considered in this thesis.

1.1.1. Complexity of Computational Problems

We intuitively explain what is a computational problem, what is an algorithm, how we usually analyze the running time of an algorithm, and how we classify computational problems by their computational complexity. For a more elaborate introduction to these concepts, one might consult any textbook on computational complexity, for example, the book by Sipser [137].

Computational Problems. Perhaps the best way to explain what is a computational problem and what is the complexity of a computational problem is by discussing examples. Consider the following examples of computational problems.

(For simplicity, these computational problems regard numbers, but, indeed, other types of computational problems exist.)

ADDITION	Given two numbers, x and y, compute the value of $x + y$.
MULTIPLICATION	Given two numbers, x and y, compute the value of $x * y$.

It might be useful for some applications to solve these computational problems. That is, to find algorithms for these computational problems.

Algorithms. Perhaps the best way to explain what is an algorithm is by discussing examples. Consider the following algorithms.

For the first problem, the ADDITION problem, several algorithms are known. The algorithm that most people learn at school is sometimes called the *Short Addition* algorithm. As an example for this algorithm, consider the following input to the ADDITION problem:

"Add $x = 243$ and $y = 135$ together."

The way the Short Addition algorithm proceeds on this input is by performing the following *operations*. (For ease of presentation, and without loss of generality, we do not consider cases of "carry-over", which happen, for example, when adding 7 and 5 together.)

1) Add the ones ($3 + 5 = 8$), and write as the result's ones.

2) Add the tens ($4 + 3 = 7$), and write as the result's tens.

3) Add the hundreds ($2 + 1 = 3$), and write as the result's hundreds.

For the second problem, the MULTIPLICATION problem, several algorithms are known. The algorithm that most people learn at school is sometimes called the *Long Multiplication* algorithm. As an example for this algorithm, consider the following input to the MULTIPLICATION problem:

"Multiply $x = 243$ and $y = 135$ together."

The way the Long Multiplication algorithm proceeds on this input is by performing the following *operations*.

1) Multiply the ones of y with the ones of x, that is, $5 * 3 = 15$.

2) Multiply the ones of y with the tens of x, that is, $5 * 4 = 20$.

3) Multiply the ones of y with the hundreds of x, that is, $5 * 2 = 10$.

4) Multiply the tens of y with the ones of x, that is, $3 * 3 = 9$.

5) Multiply the tens of y with the tens of x, that is, $3 * 4 = 12$.

6) Multiply the tens of y with the hundreds of x, that is, $3 * 2 = 6$.

7) Multiply the hundreds of y with the ones of x, that is, $1 * 3 = 3$.

8) Multiply the hundreds of y with the tens of x, that is, $1 * 4 = 4$.

9) Multiply the hundreds of y with the hundreds of x, that is, $1 * 2 = 2$.

We are interested in counting the number of operations these algorithms perform on different inputs, as an estimate of their running times.

Analysis of Algorithms. We want to analyze the algorithms described above, in order to understand the amount of time it takes for these algorithms to compute their answer. We are not interested in the exact running times (as these also depend on the specific computing device on which these algorithms might run), but are satisfied with rough estimates of them. Thus, we are happy with counting the number of operations each of these algorithms performs.

Consider the example which is given for the Short Addition algorithm. There, the algorithm performs three operations, since it performs one operation per each digit. More generally, for two input numbers with n digits each, the algorithm performs n operations. Thus, we say that the algorithm runs in linear time, or, equivalently, that its running time is $O(n)$; the notation $O(n)$ is to be read as "order n": this means that the algorithm runs in time which is in order of magnitude as n.

Consider the example which is given for the Long Multiplication algorithm. There, the algorithm performs nine operations, since it performs one operation per each pair of digits (one digit of x and one digit of y). More generally, for two input numbers with n digits each, the algorithm performs $n * n = n^2$ operations. Thus, we say that the algorithm runs in quadratic time, or, equivalently, that

its running time is $O(n^2)$; the notation $O(n^2)$ is to be read as "order n squared": this means that the algorithm runs in time which is in order of magnitude as n squared.

Computational Complexity. We have described two computational problems, the ADDITION problem and the MULTIPLICATION problem, and one algorithm for each of them, the Short Addition algorithm for the ADDITION problem and the Long Multiplication algorithm for the MULTIPLICATION problem.

We have established the fact that the algorithm presented for the ADDITION problem is faster than the algorithm presented for the MULTIPLICATION problem. Specifically, the Short Addition algorithm runs in time which is proportional to the number of digits of the input numbers, while the Long Multiplication algorithm runs in time which is proportional to the number of digits of the input numbers squared.

One might ask whether it is true that multiplying two numbers is computationally harder than adding two numbers. This is the type of questions we ask in this thesis, and this is what computational complexity studies. That is, for two computational problems, A and B, we want to know:

"is problem A computationally harder than problem B?"

That is, the goal of computational complexity is to identify computational problems which are of similar complexity, and to distinguish between problems, based on their complexity.

Note that, only taking the above discussion into account, we cannot yet conclude whether MULTIPLICATION is harder than ADDITION, since there might be better algorithms for MULTIPLICATION which run in time which is linear in the input size (in effect, run as fast as the Short Addition algorithm). This question, quite strikingly, is still open. That is, while some better algorithms for MULTIPLICATION are known, none of them runs in $O(n)$ time, and, importantly, there is no proof that such an algorithm cannot exist.

1.1.2. Computational Intractability and Parameters

After establishing the basics of computational complexity, we are ready to consider problems which are believed to be computationally intractable.

Computational Tractability and Intractability. Consider the following fundamental computational problem, called the SET COVER problem. In this problem we are given a set of objects (for example, a set of numbers) and a collection of sets of these objects. The task is to find a subcollection (of minimum size) from the given collection of sets, where each object is a member of at least one set of our chosen subcollection. Consider the following example.

Example 1. We are given the numbers 1, 2, 3, 4, and 5, as well as the collection of sets $\{1,3,4\}$, $\{2,3,4\}$, and $\{2,4,5\}$.

The solution is the subcollection containing the sets $\{1,3,4\}$ and $\{2,4,5\}$, as it is a subcollection of minimum size (two sets) such that each number is a member of at least one set in this subcollection. △

One simple algorithm for the SET COVER problem proceeds by considering all subcollections from the collection of sets. For each subcollection, the algorithm checks whether each number is a member of at least one set in the subcollection. Finally, the algorithm chooses the smallest subcollection considered which satisfy the above property.

For the instance described in Example 1, the algorithm needs to consider the following subcollections (indeed, these are all the possible subcollections of the given collection of sets).

1) $\{1,3,4\}$.

2) $\{2,3,4\}$.

3) $\{2,4,5\}$.

4) $\{1,3,4\}$ and $\{2,3,4\}$.

5) $\{1,3,4\}$ and $\{2,4,5\}$.

6) $\{2,3,4\}$ and $\{2,4,5\}$.

7) $\{1,3,4\}$, $\{2,3,4\}$, and $\{2,4,5\}$.

That is, the algorithm needs to consider 7 subcollections. More generally, for an input with n sets, the algorithm needs to consider $2^n - 1$ subcollections. Thus, we say that the algorithm runs in exponential time, or, equivalently, that its running time is $O(2^n)$; the notation $O(2^n)$ is to be read as "order 2 to the n": this means that the algorithm runs in time which is in order of magnitude as 2^n.

(For simplicity, we assume that the algorithm performs one operation per each subcollection.)

To appreciate the growth of the exponential function, let us look at some values of 2^n, and compare them to some values of n^2. Consider the following table.

n	n^2	2^n
1	1	2
2	4	4
3	9	8
10	100	1024
20	400	1048576
30	900	1073741824
100	10000	1267650600228229401496703205376

The point is that the exponential function grows very fast. In fact, it grows so fast that, for example, if we feed an input of size 1000 digits to an algorithm which runs in exponential time, then the algorithm will halt only when the universe will collapse. Thus, the field of computational complexity distinguishes between the following two types of problems.

Problems which are polynomial-time solvable. (Type 1)

Problems which are believed not to be polynomial-time solvable. (Type 2)

Problems which are polynomial-time solvable are problems for which there are algorithms whose running time which can be written as n^c, for some constant c. Thus, Type 1 contains, for example, problems which can be solved in linear time (where $c = 1$, that is, $O(n)$; for example, the ADDITION problem) and problems which can be solved in quadratic time (where $c = 2$, that is, $O(n^2)$; for example, the MULTIPLICATION problem). Type 2 contains, for example, the SET COVER problem. This means, specifically, that it is believed that there is no algorithm for the SET COVER problem which runs in polynomial-time (this belief is in the heart of the famous P versus NP problem; specifically, it is believed that $\mathsf{P} \neq \mathsf{NP}$).

Computational complexity considers problems of Type 1 as being tractable (that is, efficiently solvable), while problems of Type 2 as being intractable (that is, as problems which cannot be solved efficiently; recall the collapse of the universe). In this thesis we consider several computational problems, and for each of them we try to establish whether it is of Type 1 (tractable) or of Type 2 (intractable).

It turns out that most problems considered in this thesis are intractable. To further study these problems, and to better understand their inherent complexity, we use the framework of parameterized complexity.

Parameterized Complexity. Lots of computational problems, including the problems considered in this thesis, can naturally be associated with some parameters. For example, three natural parameters for the SET COVER problem are the number of numbers given, the number of sets in the given collection, and the maximum size of the sets in the given collection. There are applications for which certain parameters for specific problems are known to be small. Thus, algorithms which are efficient whenever these parameters are small are of interest.

Parameterized complexity is concerned with finding such algorithms. That is, we ask whether it is possible to devise (parameterized) algorithms which are efficient whenever some parameters are small. In other words, while we require from a parameterized algorithm to be correct on all inputs, we require from it to be fast only on those inputs for which the respective parameters are small.

Specifically, similarly to the distinction done in computational complexity between tractable and intractable problems, parameterized complexity further distinguishes between tractable and intractable parameterized problems. While some efficient parameterized algorithms are known for some problems with respect to some parameters, it is believed that no efficient parameterized algorithms exist for some problems with respect to some parameters.

In this thesis, when faced with an intractable problem (that is, a computational problem of Type 2), we identify some natural parameters. Then, we try to understand for which of these parameters, when assumed to be small, we can devise efficient algorithms for the problem under consideration. If we cannot devise efficient parameterized algorithms, then we try to show that, under some assumptions, such algorithms cannot exist.

1.1.3. Computational Problems Considered in this Thesis

The computational problems considered in this thesis are related to social choice and social networks. Social choice is concerned with the study of collective

decision making, for example when people vote in an election. Social networks are important constructs in biology, economics, and sociology, modeling interactions between entities. We refer the reader to the book by Brams and Fishburn [18] for an overview on social choice and to the book by Newman [124] for an overview on social networks.

There are two types of computational problems considered in this thesis. The first two problems we consider, in Chapter 3 and in Chapter 4, regard elections performed on top some underlying social networks.

More specifically, we study, in Chapter 3, how hard it is to manipulate an election by adding or removing candidates, when the candidates in the election are related to each other by an underlying social network. In Chapter 4, we again consider manipulating an election, but we allow an external agent to change some votes in the election, when the voters participating in the elections are related to each other by an underlying social network.

The last three problems we consider, in Chapter 5, in Chapter 6, and in Chapter 7, regard the privacy of entities comprising an election or a social network, where some data about the election or the social network is to be made publicly available. More specifically, in Chapter 5 we consider the problem of preserving the privacy of the voters, when data concerning the votes is to be published. Then, in Chapter 6 and in Chapter 7 we consider preserving the privacy of the entities comprising a social network, when the social network is to be published. In order to preserve the privacy of the entities comprising the social network, in Chapter 6 we allow to introduce new artificial entities to the social network, while in Chapter 7 we allow to merge sets of entities together.

Each chapter, after a brief discussion, provides a simple example to illustrate the specific computational problem considered in it.

1.2. Not-So-Gentle Introduction

In this thesis, for each computational problem under consideration, we are interested first in classifying whether the problem is polynomial-time solvable (that is, in P), by developing a combinatorial algorithm for it, or whether it is NP-hard, by providing a reduction from another NP-hard problem (we disregard computational problems that might be in between these classes). Then, for each problem which we prove (or conjecture) NP-hardness, we identify natural parameters for it, and further classify its parameterized complexity, either by devising fixed-parameter algorithms for it, thus proving containment in FPT, or

by proving W-hardness, by providing a parameterized reduction for it. Then, for some cases, we further investigate whether the parameterized problem is in XP (that is, can be solved in polynomial-time for constant values of the parameter), by devising an XP algorithm for it, or, showing that the problem is Para-NP-hard, by providing a reduction from another NP-hard problem while keeping the value of the parameter constant in the reduction. (See Chapter 2 for formal definitions of the complexity classes and notions mentioned above.)

We study two kinds of problems. The first kind is related to manipulating a given election, where some relationships between the entities of the election are assumed. This can be seen as if the election occurs on top of an underlying social network which connects either the voters participating in the election or the candidates which the voters vote on. The second kind of problems is concerned with issues of privacy when publishing social networks and election-related data. Next, we formally define most of the computational problems studied in this thesis (for clarity, omitting some variants of these problems), and state some main results for each of them.

Combinatorial Candidate Control. This problem is discussed in Chapter 3 and models the possibility of manipulating a given election by controlling the candidates which participate in the election, where there is some social network connecting the candidates.

In one of the variants of the problem we are given an election (C,V), where C is the set of candidates, V is the collection of voters, and where one of the candidates, p, is preferred by some external agent. The goal of the external agent is to delete as few candidates as possible, in order to make p the winner of the election, under some voting rule $\mathscr{R}$. There is some underlying structure connecting the candidates, such that whenever the external agent is deleting a candidate c from the election, a whole set of candidates, denoted by $\kappa(c)$, is also deleted from the election. Formally, the problem is defined as follows.

> $\mathscr{R}$-COMB-CCDC
> **Input:** An election (C,V), a preferred candidate $p \in C$, a bundling function κ, and a budget k.
> **Question:** Is there a set $C' \subseteq C$ with $|C'| \leq k$ such that p is a winner under the voting rule $\mathscr{R}$ in the resulting election $(C \setminus \kappa(C'), V)$?

As, generally, this problem is highly intractable, the computational complexity of this kind of election control is considered for elections with only a few voters.

It turns out that the parameterized complexity landscape of this problem is very diverse.

Combinatorial Shift Bribery. This problem is discussed in Chapter 4, and also considers issues of manipulating elections. Here, however, there is an underlying social network connecting the voters, rather than the candidates.

In one of the variants of the problem we are given an election (C,V), where C is the set of candidates, V is the collection of voters, and where one of the alternatives, p, is preferred by some external agent. There is a set of *shift actions*, such that each application of a shift action transforms the election by changing the position of p in some votes, as specified by the shift action. The goal of the external agent is to use as few shift actions as possible, in order to make p the winner of the election, under some voting rule $\mathscr{R}$. Formally, the problem is defined as follows.

> $\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY
> **Input:** An election (C,V), a preferred candidate $p \in C$, a set F of shift actions, and a budget k.
> **Question:** Is there a subset $F' \subseteq F$ of shift actions with $|F'| \leq k$ such that, after we apply the shift actions from F', p is a winner under the voting rule $\mathscr{R}$ in the resulting election?

It turns out that this problem is highly intractable, even for some very restricted cases. Some other cases, however, can be approximated efficiently.

Election Anonymization. This problem is discussed in Chapter 5 and is also about elections. Here, however, the main concern is preserving the privacy of the voters when publishing the votes of the voters participating in the election. The task is to modify the election, but not too much, while preserving the election winner (with respect to the given voting rule $\mathscr{R}$), and such that the resulting election is k-anonymized; an election is said to be k-anonymous if, for each voter in it, there are at least $k-1$ other voters which vote the same. The idea is that an adversary which knows the votes of some of the voters in the elections cannot identify specific voters in a k-anonymous election. Formally, the problem is defined as follows.

> $\mathscr{R}$-d-ELECTION ANONYMIZATION
> **Input:** An election $E = (C,V)$, anonymity level k, and a budget s.

Question: Is there a k-anonymous election E' such that $d(E,E') \leq s$ and the set of winners of E' equals that of E, under the voting rule $\mathscr{R}$ (where d is a metric, such that $d(E,E')$ models how close the elections E and E' are to each other)?

It turns out that the parameterized complexity landscape of this problem is very diverse, with cases ranging from being polynomial-time solvable to Para-NP-hard.

Degree Anonymization by Vertex Addition. This problem is discussed in Chapter 6 and is also about anonymization, but for social networks and not for elections.

Assuming an adversary which knows the degrees of some vertices in a given graph, the task is to transform a given graph into a k-anonymous graph, where a graph is said to be k-anonymous if, for every vertex in it, there are at least $k-1$ other vertices with the same degree. The way to achieve anonymity is by introducing new vertices. This problem is formally defined as follows.

DEGREE ANONYMIZATION (VA)
Input: A simple undirected graph $G = (V,E)$, anonymity level k, and a budget t.
Question: Is there a k-anonymous graph $G' = (V \cup V', E \cup E')$ such that $|V'| \leq t$ and $E' \subseteq \{\{u,v\} \subseteq V \cup V' : u \in V' \vee v \in V'\}$?

It turns out that this problem is computationally harder than the related problem of degree anonymization by edge addition. Specifically, while that problem is FPT with respect to the maximum degree, the problem discussed here is Para-NP-hard for this parameter.

Degree Anonymization by Graph Contractions. This problem is discussed in Chapter 7 and is also about anonymizing social networks. Here, however, the allowed operations are graph contractions (for example, edge contractions) and not vertex additions. This problem is formally defined as follows.

DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS
Input: An undirected graph $G = (V,E)$, anonymity level k, and a budget c.
Question: Can G be made k-anonymous by performing at most c contractions?

It turns out that this problem is computationally harder than the related problem of degree anonymization by vertex addition, but there are some cases for which it is fixed-parameter tractable.

Each chapter in this thesis corresponds to a different problem, and can be seen as a quest for tractability, where the goal is to understand the computational hardness inherent in some special cases and parameterizations and to devise parameterized algorithms for other parameterizations and special cases.

2. Preliminaries

Assuming only basic understanding of computational complexity, this chapter provides all the needed preliminaries for understanding this thesis. We begin with some basic mathematical definitions and notations (Section 2.1). Section 2.2 provides a brief introduction to parameterized complexity and to the main algorithmic techniques used for devising parameterized algorithms. Then, we formally define the two main mathematical structures which will be considered in this thesis: graphs (Section 2.3) and elections (Section 2.4). Finally, in Section 2.5, we provide a list of some computational problems used in this thesis.

2.1. General Notations

We use the following standard mathematical notations.

$\mathbb{N}$	denotes the set of natural numbers.
$\mathbb{Q}$	denotes the set of rational numbers.
$\mathbb{R}$	denotes the set of real numbers.
$[n]$	(for $n \in \mathbb{N}$) denotes the set $\{1, 2, \ldots, n\}$.
$[l, u]$	(for $l, u \in \mathbb{N}$) denotes the set $\{l, l+1, \ldots, u\}$.
$f(k)$	denotes a computable function dependent only on k.
$\text{poly}(n)$	denotes a polynomial function on n (that is, n^c for some constant c).

2.2. Parameterized Complexity

In Section 2.2.1, we provide an overview of the parameterized complexity classes which will be used in this thesis. Then, we provide an overview of the algorithmic techniques used in this thesis for devising parameterized algorithms. For a more detailed introduction to these concepts we refer the interested reader to the available textbooks on parameterized complexity [57, 82, 125, 53].

2.2.1. Parameterized Complexity Classes

We use the following parameterized complexity classes, where n denotes the size of the encoding of the input and k denotes the value of the parameter (a more thorough explanation follows).

P	denotes the class of problems solvable in n^c, for some fixed c.
FPT	denotes the class of problems solvable in $f(k) \cdot n^c$, for some fixed c.
XP	denotes the class of problems solvable in $O(n^k)$.

We assume the following widely-accepted assumptions.

P	$\neq$	NP
FPT	$\neq$	W[1]

Assuming the above, we have the following.

NP-hardness	rules out the possibility of an algorithm running in $O(n^c)$ time, for some fixed c.
W-hardness	rules out the possibility of an algorithm running in $f(k) \cdot n^c$ time, for some fixed c.
Para-NP-hardness	rules out the possibility of an algorithm running in $O(n^k)$ time.

A more detailed introduction to these complexity classes follows. In parameterized complexity, the goal is to find parameters which are "responsible" for the intractability of the problem under consideration. Then, for instances for which these parameters are small, we might be able to devise efficient algorithms.

For example, recall the VERTEX COVER problem, which, given an undirected graph G and an integer k, asks for a set of at most k vertices such that each edge is incident to at least one vertex from this set. While the VERTEX COVER is NP-hard [84], it is possible to solve it in $f(k)\cdot n^c$ time, where f is some computable function depending only on the value of k [57]. Therefore, if the size of the vertex cover is not too big, and, importantly, even if the graph is huge, we can still find a solution in "reasonable" time, that is, in polynomial time, such that the degree of the polynomial does not depend on the input size. Thus, we say that the VERTEX COVER problem is *fixed-parameter tractable* with respect to the solution size.

More formally, an instance (I,k) of a parameterized problem consists of the "classical" problem instance I and an integer k being the *parameter* [57, 82, 126]. A parameterized problem is called *fixed-parameter tractable* (FPT) if there is an algorithm solving it in $f(k)\cdot|I|^{O(1)}$ time, for an arbitrary computable function f only depending on the parameter k.

There are, however, some problems that, for some parameters, seem not to be in FPT. Two fundamental examples are the CLIQUE problem and the SET COVER problem, whose definitions are given in Section 2.5.

For these two problems, no algorithm running in time $f(h)\cdot n^c$ is known, where h is the solution size, and it is widely-accepted that no such algorithm exists (since these problems are W-hard). Building on these assumptions, one can show that a parameterized problem L is (presumably) not fixed-parameter tractable (that is, W-hard) by devising a *parameterized reduction* from the CLIQUE problem or the SET COVER problem (both, when parameterized by the solution size) to L. A parameterized reduction from a parameterized problem L to another parameterized problem L' is a function that, given an instance (I,k), computes in $f(k)\cdot|I|^{O(1)}$ time an instance (I',k') (with $k'\leq g(k)$) such that $(I,k)\in L \iff (I',k')\in L'$. Similarly to the usual way of proving NP-hardness, lots of problems are known to be W-hard and can be used as bases for parameterized reductions.

Note that W-hardness does not rule out the possibility of algorithms running in $|I|^{f(k)}$ time. This running time ensures that the algorithms run in polynomial time as long as the parameter is a constant. Importantly, however, this running time is inferior to the FPT running time, as the it grows fast with respect to the size of the input, which is not the case for FPT. Clearly, FPT $\subseteq$ XP. As an

example, we mention that a simple algorithm, checking all subsets of vertices of size k, proves XP-membership for the CLIQUE problem.

Finally, a parameterized problem which is NP-hard even for instances for which the parameter is a constant (and therefore, not in XP with respect to this parameter) is said to be Para-NP-hard.

2.2.2. Parameterized Algorithmics

We provide the main ideas behind some general algorithmic techniques used in this thesis to prove membership in FPT. Notably, most of the algorithmic results presented in this thesis are achieved by combining some of these techniques.

Branching. The idea here is to search the solution space while making sure that the size of the search tree and the time spent at each node is upper-bounded by some function dependent only on the parameter. The main observation used is that sometimes the number of possibilities that the algorithm needs to branch into are upper-bounded by some function dependent only on the parameter.

Theorem 5.4, Theorem 6.8, and Theorem 7.5 are due to branching.

Dynamic Programming. The idea here is to use the efficient recursion of dynamic programming while making sure that the size of the state, being stored at each recursive call, is upper-bounded by a function only dependent on the parameter, and, also, that the time spent at each recursive call can be upper-bounded by a function only dependent on the parameter.

Theorem 5.1 and Theorem 6.7 are due to dynamic programming.

Integer Linear Programming. The idea here is to formulate the problem at hand as an integer linear program (ILP) where the number of variables is upper-bounded by a function dependent only on the parameter. Then, fixed-parameter tractability follows by applying a famous result of Lenstra [104].

Theorem 5.5 and Theorem 6.8 are due to integer linear programming.

Kernelization. The idea here is to perform a specific kind of preprocessing on the input. The goal is to transform, in polynomial-time, a given problem instance (I,k) into an equivalent instance (I',k') whose size is upper-bounded by a function dependent only on k. That is, (I,k) is a yes-instance if and only if (I',k') is a yes-instance, where $k' \leq g(k)$ and $|I'| \leq g(k)$, for some computable

function g. Indeed, such a transformation is a polynomial-time self-reduction with the constraint that the reduced instance is "small" (measured by $g(k)$). In case that such transformation exists, (I', k') is called a *kernel* of size $g(k)$. Usually, a kernel is achieved by exhaustively applying a set of *reduction rules*, each of them transforming an instance to another instance, while shrinking the size of the instance. A reduction rule is said to be *correct* if the transformed instance is always equivalent to the original instance, that is, both are yes-instances or both are no-instances. It is well-known that a parameterized problem is fixed-parameter tractable if and only if it admits a kernelization [88, 100].

Theorem 6.10 and Corollary 7.1 are due to kernelization.

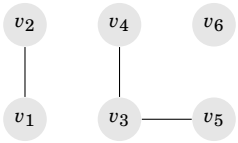
2.3. Graphs and Social Networks

Some of the problems we consider in this thesis directly concern social networks. Mathematically, we view a social network as an undirected graph. One might imagine that the vertices correspond to the persons in the social network and that there is an edge between two vertices if, for example, the corresponding persons are friends.

Next, we provide some definitions and notations regarding graphs. For these definitions, we assume a given undirected graph $G = (V, E)$ where V is the set of vertices and $E \subseteq \binom{V}{2}$ is the set of edges (we use the notation $\binom{V}{2}$ as a shorthand for the set of all unordered pairs of vertices). Indeed, unless specifically stated otherwise, all graphs considered in this thesis are undirected. For more information on graph theory we refer the reader to any textbook on graph theory, for example the book by Diestel [55].

$\deg(v)$	denotes the degree of vertex v, for $v \in V$.
Δ	denotes the maximum degree in the graph, that is, $\Delta := \max_{v \in V} \deg(v)$.
B_d	denotes the set of vertices of degree d, for $0 \leq d \leq \Delta$, that is, $B_d := \{v \in V : \deg(v) = d\}$. B_d is called the *block* of degree d.

As an example, consider the following graph.



In the above graph, we have $\Delta = 2$, since

$$0 = \deg(v_6),$$

$$1 = \deg(v_1) = \deg(v_2) = \deg(v_4) = \deg(v_5),$$

$$2 = \deg(v_3).$$

Hence, we have that

$$B_0 = \{v_6\},$$

$$B_1 = \{v_1, v_2, v_4, v_5\},$$

$$B_2 = \{v_3\}.$$

Most of the time we will consider simple graphs, but for non-simple graphs which might have self-loops and parallel edges, we define the degree of a vertex v with x neighbors and y self-loops to be, as usual, $x + 2y$ (note that, in particular, we count a self-loop twice).

2.3.1. Special Graph Classes

We consider the following special graph classes.

A *tree*	is a connected graph with no cycles.
A *path graph*	is a tree with no vertices of degree larger than 2.
A *caterpillar tree*	is a tree for which removing the leaves and their incident edges leaves a path graph.
A *d-regular graph*	is a graph where all vertices have degree d.
A *cubic graph*	is a 3-regular graph.

2.4. Elections and Control

An election E is a pair (C,V) where

$C = \{c_1,\ldots,c_m\}$ is the set of candidates.

$V = (v_1,\ldots,v_n)$ is the collection of voters.

For clarity, we usually refer to the voters as females while the candidates are referred to as males. Each voter v_i is equipped with her preference order, which is a total order $\succ_{v_i}$ over C. A voter which prefers c to all other candidates is called a *c-voter*.

For a voter v_ℓ and two candidates, c_i and c_j, we write $v_\ell\colon c_i \succ c_j$ to indicate that v_ℓ prefers c_i to c_j. For a subset A of candidates, by writing A (or, sometimes, $\overrightarrow{A}$) within a preference order description (for example, $A \succ a \succ b$, where a and b are some candidates not in A) we mean listing the members of A in some arbitrary, but fixed, order. By writing $\overleftarrow{A}$ we mean listing the alternatives in the reverse of this arbitrary, and fixed, order. Given an election $E = (C,V)$ and two candidates, $c_i, c_j \in C$, we define $N_E(c_i,c_j) := |\{v_\ell \in V : (v_\ell\colon c_i \succ c_j)\}|$ (that is, $N_E(c_i,c_j)$ is the number of voters preferring c_i to c_j). For more information about social choice, election, and voting, we refer the interested reader to any book on these topics, for example the book by Arrow et al. [2] (for some research challanges in parameterized algorithmics for computational social choice, we refer the reader to the survey by [26]).

2.4.1. Voting Rules

A voting rule $\mathscr{R}$ is a function that, given an election $E = (C,V)$, returns a set $\mathscr{R}(E) \subseteq C$ of election winners. Note that we use the non-unique winner model, meaning that a voting rule might return several tied winners. We consider the following voting rules.

The Plurality rule. Each candidate receives one point for each voter which ranks him first, and the winners are the highest-scoring candidates.

The Veto rule. Each candidate receives one point for each voter which ranks him last, and the winners are the lowest-scoring candidates.

The t-Approval and the t-Veto rules. Under t-Approval, $t \geq 1$, each candidate gets a point for each voter that ranks him among her top t positions. The candidates receiving the highest number of points are the winners.

For m candidates, t-Veto is a synonym for $(m-t)$-Approval. Indeed, 1-Approval is the Plurality rule and 1-Veto is the Veto rule. We refer to all t-Approval and t-Veto voting rules as Approval-based rules.

The Borda rule. Under the Borda rule, each candidate $c \in C$ receives

$$\sum_{d \in C \setminus \{c\}} N_E(c,d)$$

points, and the candidates with the highest points win. Equivalently, under the Borda rule, each voter gives each candidate c as many points as there are candidates that this voter ranks below c.

The Maximin rule. Under the Maximin rule, each candidate $c \in C$ receives

$$\min_{d \in C \setminus \{c\}} N_E(c,d)$$

points, and the candidates with the highest points wins.

The Copeland$^\alpha$ rule. Under the Copeland$^\alpha$ rule, each candidate c receives

$$|\{d \in C \setminus \{c\} : N_E(c,d) > N_E(d,c)\}| + \alpha |\{d \in C \setminus \{c\} : N_E(c,d) = N_E(d,c)\}|$$

points. Intuitively, under the Copeland$^\alpha$ rule we conduct a head-to-head contest among each pair of candidates. For a given pair of candidates, c and d, the candidate who is preferred to the other by a majority of voters receives one point. If there is a tie, then both candidates receive α points, where α is a rational number and $0 \leq \alpha \leq 1$.

The Condorcet rule. A candidate c is a (strong-)Condorcet winner if, for each other candidate $c' \in C \setminus \{c\}$, it holds that

$$|\{v \in V : c \succ_v c'\}| > |\{v \in V : c' \succ_v c\}|,$$

that is, if he beats all other candidates in head-to-head contests. The Condorcet rule elects the (unique) Condorcet winner if it exists, returning an empty set otherwise.

A candidate c is a *weak-Condorcet* winner if, for each other candidate $c' \in C \setminus \{c\}$, it holds that

$$|\{v \in V : c \succ_v c'\}| \geq |\{v \in V : c' \succ_v c\}|,$$

that is, if he beats or ties all other candidates in head-to-head contests. The weak-Condorcet rule elects the set of weak-Condorcet winners.

Example 2. Consider the following example election. The election consists of the candidates $\{a, b, c\}$ and of the voters $\{v_1, v_2, v_3, v_4\}$, with the following preference orders.

$$v_1 : a \succ c \succ b \succ d$$

$$v_2 : a \succ c \succ b \succ d$$

$$v_3 : b \succ c \succ d \succ a$$

$$v_4 : c \succ b \succ d \succ a$$

In this example, voter v_3 prefers b the most, then c, then d, and a last. Considering some of the voting rules considered in this thesis, we have the following winners.

- a is the Plurality winner of this election.
- c is the Borda winner and the 2-Approval winner.
- Both a and c are the weak-Condorcet winners.
- There is no strong-Condorcet winner. △

2.5. Some Useful Computational Problems

In this section, we describe some computational problems that will be useful later in this thesis, mainly to show NP-hardness, W-hardness, and Para-NP-hardness.

SET COVER (used, for example, in Theorem 5.2 and Theorem 6.2)

Definition. Given a universe of elements X, a collection $\mathscr{S}$ of sets of elements of X, and a budget h, the task is to decide whether there is a subcollection $\mathscr{S}' \subseteq \mathscr{S}$ of sets such that $|\mathscr{S}'| \leq h$ and $\bigcup_{S \in \mathscr{S}'} S = X$. Such a subcollection $\mathscr{S}'$ is called a *set cover*.

Complexity. The SET COVER problem is NP-hard and W[2]-hard with respect to the solution size [57, 84]. The problem remains NP-hard even when all sets contain three elements, each element appears in exactly three sets, and each element is to be covered exactly once [86] (this restricted variant is called RESTRICTED EXACT COVER BY 3-SETS).

CLIQUE (used, for example, in Theorem 6.4)

Definition. Given an undirected graph G and an integer h, the task is to decide whether there is a set of h pairwise adjacent vertices.

Complexity. The CLIQUE problem is NP-hard and W[1]-hard with respect to the solution size [57, 84].

MULTICOLORED CLIQUE (used, for example, in Theorem 7.3)

Definition. Given an undirected graph whose vertices are colored in h colors and an integer h, the task is to decide whether there is a set of h pairwise adjacent vertices such that each vertex is of different color.

Complexity. The MULTICOLORED CLIQUE problem is NP-hard and W[1]-hard with respect to the solution size even on regular graphs [81, 84].

INDEPENDENT SET (used, for example, in Theorem 6.1)

Definition. Given an undirected graph G and an integer h, the task is to decide whether there is a set of h pairwise non-adjacent vertices.

Complexity. The INDEPENDENT SET problem is NP-hard even on cubic graphs [84] and W[1]-hard with respect to the solution size [57].

VERTEX COVER (used, for example, in Theorem 7.4)

Definition. Given an undirected graph G and an integer h, the task is to decide whether there is a set of h vertices such that each edge is incident to at least one vertex from the set.

Complexity. The VERTEX COVER problem is NP-hard even on cubic graphs [84] and FPT with respect to the solution size [57].

PARTITION INTO TRIANGLES (used, for example, in Theorem 7.4)

Definition. Given an undirected graph G with n vertices, the task is to decide whether the vertices can be partitioned into $n/3$ sets of vertices such that each set form a triangle in G.

Complexity. The PARTITION INTO TRIANGLES problem is NP-hard even on 4-regular graphs [140].

SUBSET SUM (ZERO VARIANT) (used, for example, in Theorem 4.1)

Definition. Given a set of integers $A = \{a_1, \ldots, a_n\}$, the task is to decide whether there is a set $A' \subseteq A$ such that $\sum_{a_i \in A'} a_i = 0$.

Complexity. The SUBSET SUM (ZERO VARIANT) problem is weakly NP-hard [84].

CHANGE MAKING (used, for example, in Theorem 6.6)

Definition. Given integers $a_1, \ldots, a_n$ and integers m and b, the task is to decide whether there are non-negative integers $x_1, \ldots, x_n$ such that $\Sigma_{i \in [n]} x_i \leq m$ and $\Sigma_{i \in [n]} x_i a_i = b$.

Complexity. The CHANGE MAKING problem is weakly NP-hard [110].

STRICTLY THREE PARTITION (used, for example, in Theorem 7.1)

Definition. Given a set of integers $S = \{a_1, \ldots, a_{3m}\}$ such that $\sum_{a_i \in S} a_i = mB$ and $\forall i \in [3m] : B/4 < a_i < B/2$, the task is to decide whether there are m disjoint sets $S_1, \ldots, S_m$, $S_i \subseteq S$ for $i \in [m]$, each of containing three integers, such that $\forall j \in [m] : \sum_{a_i \in S_j} a_i = B$.

Complexity. The STRICTLY THREE PARTITION problem is strongly NP-hard [84].

NUMERICAL MATCHING WITH TARGET SUMS (used, for example, in Theorem 7.2)

Definition. Given three sets of integers $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $C = \{c_1, \ldots, c_n\}$, the task is to decide whether the elements of A and B can be paired such that, for each $i \in [n]$, c_i is the sum of the ith pair.

Complexity. The NUMERICAL MATCHING WITH TARGET SUMS problem is strongly NP-hard [84].

3. Candidate Control

In the following two chapters we consider the possibility of manipulating elections, where there is some underlying structure connecting the voters or the candidates participating in the election.

From this point of view, this chapter and Chapter 4 represent a step towards understanding the possibility of manipulating elections occurring over social networks.

In the combinatorial problem considered in Chapter 4, an external agent is allowed to manipulate a given election by changing how some voters vote, depending on some structure connecting the voters participating in the election. In this chapter, however, an external agent is allowed to manipulate a given election by changing the set of candidates participating in the election. There is an underlying social network connecting the candidates participating in the election such that whenever the external agent adds or deletes a candidate from the election, a whole group of (related) candidates are added or deleted from the election.

Specifically, in this chapter we consider both the standard scenario of adding and deleting candidates, where one asks whether a given candidate can become a winner (or, in the destructive case, can be precluded from winning) by adding or deleting few candidates, as well as the combinatorial scenario where adding or deleting a candidate automatically means adding or deleting a whole group of candidates.

Since it turns out that this problem is highly intractable in general, we concentrate on the well-motivated scenario of elections with only a few voters (but, possibly, a huge number of candidates). Considering several fundamental voting rules, the results obtained in this chapter show that the parameterized complexity of candidate control, with the number of voters as the parameter, is much more varied than in the setting with many voters, and that the combinatorial scenario behaves somehow different from the non-combinatorial scenario.

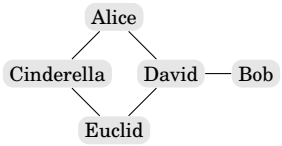


Figure 3.1. The social network used in the illustrating example.

3.1. Illustrating Example

Consider the following example, which will be used throughout the thesis.

Example 3. There is a group of five people: Alice, Bob, Cinderella, David, and Euclid. The social network depicted in Figure 3.1 shows the friendship relationships between the people in this group (for example, David is the only friend of Bob).

This group of people decided to choose (that is, to elect) a group representative, and it is currently agreed that everybody will participate as voters and as candidates in the corresponding election. Clearly, each person in the group has preferences over which person should be the group representative. These preferences are described in Figure 3.2 (for example, Cinderella prefers herself the most, then Alice, then David, then Bob, and finally Euclid).

$$
\begin{array}{rcccccccccc}
\text{Alice} & : & \text{Alice} & \succ & \text{Cinderella} & \succ & \text{David} & \succ & \text{Bob} & \succ & \text{Euclid} \\
\text{Bob} & : & \text{Euclid} & \succ & \text{Bob} & \succ & \text{David} & \succ & \text{Alice} & \succ & \text{Cinderella} \\
\text{Cinderella} & : & \text{Cinderella} & \succ & \text{Alice} & \succ & \text{David} & \succ & \text{Bob} & \succ & \text{Euclid} \\
\text{David} & : & \text{David} & \succ & \text{Cinderella} & \succ & \text{Euclid} & \succ & \text{Alice} & \succ & \text{Bob} \\
\text{Euclid} & : & \text{Cinderella} & \succ & \text{David} & \succ & \text{Euclid} & \succ & \text{Alice} & \succ & \text{Bob}
\end{array}
$$

Figure 3.2. The election used in the illustrating example.

Under the Plurality voting rule, we have that Cinderella is the current winner, since she is positioned first in the votes of Cinderella and Euclid, and all other candidates are positioned first in the vote of at most one person.

This outcome of the election, however, is clearly not the desired outcome of the election for Bob, since Bob would like Euclid to win the election.

Corresponding to the computational problem considered in this chapter, let us assume that Bob can choose one person from the group and persuade him (or her) not to participate as a candidate in the election. Moreover, taking their friendship social network into account, it so happens that whenever Bob persuades a person to not participate as a candidate in the election, as a result, also his (or her) friends will not participate in the election. In this chapter, we ask the following question: "which person shall Bob persuade?"

In the current example, the answer is Alice. That is, if Bob decides to persuade Alice not to participate as a candidate in the election, then Cinderella and David would not participate as candidates in the election as well, and the resulting election would be as shown in Figure 3.3.

$$
\begin{aligned}
\text{Alice} &: \text{Bob} \succ \text{Euclid} \\
\text{Bob} &: \text{Euclid} \succ \text{Bob} \\
\text{Cinderella} &: \text{Bob} \succ \text{Euclid} \\
\text{David} &: \text{Euclid} \succ \text{Bob} \\
\text{Euclid} &: \text{Euclid} \succ \text{Bob}
\end{aligned}
$$

Figure 3.3. The resulting election.

Now, under the Plurality voting rule, we have that Euclid is the winner, exactly what Bob hoped for. △

3.2. Introduction

Election control problems model the issue of affecting the result of an election by either introducing some new candidates or voters or by removing some existing candidates or voters from the election. In this chapter, we focus on election

control by adding or deleting candidates (so-called "candidate control"), for the case where the election involves only a few voters.

We focus on very simple, practical voting rules such as the Plurality rule and the Veto rule, but discuss several other rules as well. Specifically, we consider the t-Approval and the t-Veto rules, as well as the Borda, the Maximin, and the Copeland rules.

Besides the standard candidate control problem mentioned above, we are particularly interested in their *combinatorial* variants, where instead of adding or deleting individual candidates, the external agent adds or deletes whole groups of candidates. In this we follow the path initiated by [33], which introduced this type of combinatorial variant for control, specifically on combinatorial voter control. The point is that the candidates might be connected to each other by some underlying social network, therefore influencing each other. For example, whenever a candidate decides to participate in the election, he influences his friends to also participate in it. This combinatorial type of manipulating elections is the subject of Chapter 4 as well, albeit for different kind of manipulating operation.

Indeed, most of the candidate control problems for most of the typically studied voting rules are NP-hard (indeed, candidate control problems are NP-hard even for the Plurality rule; nonetheless, there are some natural examples of candidate control problems with polynomial-time algorithms). It turns out that for the case of elections with few voters, that is, for control problems parameterized by the number of voters, the computational complexity landscape of candidate control is much more varied and sometimes quite surprising. A high-level discussion of our results is presented in Section 3.3.1 (to get a quick feel of the nature of the results we obtain, the reader might also wish to consult Table 3.1 and Table 3.2).

The study of the computational complexity of control problems in elections was initiated by Bartholdi et al. [5] and was continued by many researchers (see the surveys of Faliszewski et al. [73, 76] as well as Section 3.2.3 for a literature overview). While many researchers have considered the case of few candidates—see, for example, the classic work of Conitzer et al. [50] on election manipulation and subsequent papers regarding control [75, 77, 97]—very little effort was invested into studying the case of few voters (perhaps the most notable example of a paper focusing on this case is that of Brandt et al. [20]).

One possible reason for this situation is that the case of few voters may seem somewhat less natural. After all, presidential elections, an archetype of elections, rarely involve more than a few candidates but do involve millions of voters. We

argue that the case of few voters is as natural and as important to study, especially in the context of multi-agent settings and various non-political elections.

3.2.1. Elections with Few Voters

Let us now argue why we believe that elections with few voters are natural, and why (combinatorial) candidate control is an important issue regarding such elections. First, let us look at the following examples, which include few voters but possibly very many candidates.

Hiring committee. Consider a university department which is going to hire a new faculty member. Typically, the committee consists of relatively few faculty members, but it may consider hundreds of applications for a given position. The members of the committee have to aggregate their opinions regarding the candidates and it is quite natural to assume that at some point this would be done through voting.

Holiday planning. Consider a group of people who are planning to spend holidays together. The group typically would consist of no more than a dozen persons, but—technically—they have to choose from all possible options provided by the travel agents, hotels, airlines, etc. This example is particularly relevant to the case of multi-agent systems: one may foresee that in the future we will delegate the task of finding the most satisfying holiday location to our personal software agents that will negotiate with travel agents and other travelers on our behalf.

Meta-search engine. Dwork et al. [60] argued that one can build a web meta-search engine that queries several other search engines (the few voters) regarding a given query, aggregates their rankings of the web pages (the many candidates), and outputs the consensus ranking.

In all of the examples above, it is clear that prior to holding an election, the voters, or some particular individual, first shrink the set of candidates. In the case of the hiring committee, most of the applications are removed from the considerations early in the evaluation process (based on the number of journal publications, for example). The group of people planning holidays first (implicitly) removes most of the possible holiday options and, then, removes those candidates that do not fully fit their preferences: for example, they remove destinations

which are too expensive, or they remove holiday places by the sea when they are interested in hiking in the mountains, etc. The search engines usually disregard those web pages that appear completely irrelevant to a given query.

This natural process of modifying the candidate set, however, creates a natural opportunity for manipulating the result. A particularly crafty agent may remove those candidates that prevent his or her favorite candidate from winning. Similarly, after the initial process of thinning down the candidate set, a voter may request that some candidates are added back into consideration, possibly to help her favorite candidate. More importantly, it is quite realistic to assume that the voters in a small group know each other so well as to reliably predict each others' votes (this is particularly applicable to the example of the hiring committee). Thus, it is natural and relevant to study the computational complexity of candidate control parameterized by the number of voters. While control problems do not model the full game-theoretic process of adding or deleting candidates, they allow agents to compute what effects they might be able to achieve, and, if the corresponding computational problem is tractable, also how to achieve these effects.[1]

Finally, it is quite natural to consider the case where adding or deleting a particular candidate means also adding or deleting a number of other candidates. For example, if a hiring committee removes some candidate from consideration, then it might have to also remove all those with weaker publication records; if people planning holidays disregard some expensive hotel, then they might also want to remove those that cost more. Our model of combinatorial control captures these settings as well.

3.2.2. Main Contributions of this Chapter

The research presented in this chapter sheds light on some surprising patterns that were not (nearly as) visible in the context of classical complexity analysis of election control. The two most interesting patterns can be summarized as follows (by constructive control we mean variants of our problems where we want to ensure some candidate's victory, whereas by destructive control we mean cases where the goal is to prevent some candidate from winning).

[1] To the best of our knowledge, game-theoretic aspects of this process of adding or deleting candidates has not been studied in detail. However, there is a related line of research regarding *strategic candidacy*, where the candidates themselves may decide to run or not [58] (see also, for example, the works of Lang et al. [102] and Polukarov et al. [130] for some recent results).

1) In the non-combinatorial setting, destructive candidate control is easy for all our voting rules, either in the fixed-parameter tractability sense or via outright polynomial-time algorithms.

2) In the combinatorial setting, control by deleting candidates appears to be computationally harder than control by adding candidates.

We also found an interesting difference in the complexity of non-combinatorial constructive control by deleting candidates between the Plurality rule and the Veto rule (recall that under the Plurality rule we elect the candidate that is ranked first most often; under the Veto rule we elect the candidate that is ranked last least often). This is especially interesting since the rules are so similar and there is no such difference for the case of adding candidates.

Finally, we emphasize that almost all of the results presented in this chapter follow by applying proof techniques that might be useful in further research on the complexity of election problems with few voters. In particular, our W[1]-hardness results follow via reductions from the MULTICOLORED CLIQUE problem and have quite a universal structure. Similarly, our Para-NP-hardness proofs follow either via reductions from the CUBIC VERTEX COVER problem and use a universal trick to encode graphs within eight votes, or are based on embedding SET COVER instances in our problems. Our FPT algorithms also have a fairly universal structure and are based on what we call *signatures*. Indeed, we believe that, besides the specific results obtained here, introducing these proof techniques is an important contribution of this chapter.

3.2.3. Related Work

The computational complexity study of election control problems was initiated by Bartholdi et al. [5], and was later followed by numerous researchers, including, for example, Hemaspaandra et al. [95], Hemaspaandra et al. [96], Meir et al. [118], as well as others (we refer the interested reader to the survey by Faliszewski et al. [76], the book chapter of Faliszewski and Rothe [73], and to several recent papers on the topic, including those of Parkes and Xia [129], Menton and Singh [120], Erdélyi et al. [68, 69], and Rothe and Schend [134]). Briefly put, it turns out that, for standard voting rules, control problems are typically NP-hard (however, it is worth noting that some of these hardness results disappear in restricted domains, as shown, for example, by Brandt et al. [19], Faliszewski et al. [78], and Magiera and Faliszewski [112]).

Many authors also considered the computational complexity of manipulating elections by partitioning candidates or voters (see, for example, the works by Bartholdi et al. [5], Erdélyi et al. [67], Faliszewski et al. [75], Hemaspaandra et al. [95], and Menton [119], Menton and Singh [120]). Under control by partition of voters we first partition the voters into two groups, then conduct separate elections within these groups, and finally hold an election among the winners of these subelections. Further, manipulating elections by combining several control operations was studied by Faliszewski et al. [77].

There is a growing body of research regarding the parameterized complexity of voting problems (see, for example, the survey by Betzler et al. [11]), where typical parameters include the solution size (for example, the number of candidates which can be added, that is, the budget) and parameters related to the election size (that is, the number of candidates or the number of voters). When considering the solution size as the parameter, control problems typically turn out to be hard, that is, typically, W[1]-hard or W[2]-hard (see, for example, the works of Betzler and Uhlmann [9], Liu et al. [106], and Liu and Zhu [105]).

In contrast, taking the number of candidates as the parameter almost always leads to FPT results (see, for example, the papers by Faliszewski et al. [77] and by Hemaspaandra et al. [97]). These results are usually obtained by formulating the problem as an integer linear problem, as done, for example, in Theorem 5.5.

So far, however, only Betzler and Uhlmann [9] considered an election control problem parameterized by the number of voters (for the Copeland rule), and Brandt et al. [20] showed NP-hardness results for several winner determination problems even for constant numbers of voters (that is, Para-NP-hardness). Further, the parameter "number of voters" received also some limited attention in other voting settings (see, for example, the works of Betzler et al. [10], Dorn and Schlotter [56], and Bredereck et al. [27]).

The study of combinatorial election control was initiated by Chen et al. [41], who focused on voter control. Another different notion of combinatorial control was studied by Erdélyi et al. [70]. We further stress that our combinatorial view of control is different from the studies of combinatorial voting domains considered, for example, by Boutilier et al. [17], Xia and Conitzer [144], and by Mattei et al. [116].

Finally, viewing some of the work presented in this thesis as considering elections held over some underlying social networks, we mention some of the work done on voting in social networks. Conitzer [48] and Conitzer [49], as well as Procaccia et al. [131], studied two adaptations of the maximum likelihood approach to voting, where the votes, taking into account also relationships between the

voters, are seen as noisy estimations of some "ground truth". Mossel et al. [123] studied how preferences diffuse in social networks.

3.2.4. Organization of this Chapter

This chapter is organized as follows. After some preliminaries, in Section 3.3, we discuss our results, in Section 3.3.1. Specifically, in Section 3.3.4 we discuss our proof techniques in a high-level fashion, giving the main intuitive ideas, and provide the most illustrative full proofs in the following sections: in Section 3.4 we discuss W[1]-hardness proofs based on the Multicolored Clique Proof Technique, in Section 3.5 we discuss Para-NP-hardness proofs based on the Cubic Vertex Cover Proof Technique, in Section 3.6 we discuss Para-NP-hardness proofs based on the Set-Embedding Proof Technique, in Section 3.7 we discuss FPT algorithms based on the Signature Proof Technique. Finally, in Section 3.8, we present challenges for future research. Some remaining results, as well as the proofs not presented in the main text of this chapter, are deferred to the appendix to this chapter.

3.3. Specific Preliminaries

We study *candidate control* in elections, considering both constructive control (CC) and destructive control (DC), by either adding candidates (AC) or deleting candidates (DC). Thus, for example, CCAC refers to constructive control by adding candidates.

For the case of control by adding candidates problems, we make the standard assumption that the voters have preference orders over all the candidates—both those already registered and those that can be added (that is, the unregistered candidates). Naturally, when the election is conducted we consider the preference orders to be restricted only to those candidates that either were originally registered or were added. Similarly, for the case of control by deleting candidates, to compute the result of the election we restrict the preference orders only to those candidates that were not deleted.

Importantly, we consider combinatorial variants of our problems, where adding (deleting) a single candidate also automatically adds (deletes) a whole group of other candidates (in this we follow our earlier work on combinatorial voter control [33]; see also the work of Erdélyi et al. [70]). In these *combinatorial variants* (denoted with a prefix Comb), we use bundling functions κ such that for

each candidate c, $\kappa(c)$ is the set of candidates which are also added if c is added (or, respectively, which are also deleted if c is deleted). For each candidate c, we require that $c \in \kappa(c)$ and call $\kappa(c)$ the *bundle* of c. For a given subset of candidates B, we write $\kappa(B)$ to denote $\bigcup_{c \in B} \kappa(c)$. Bundling functions are encoded by explicitly listing their values for all arguments.

Formally, given a voting rule $\mathscr{R}$, our problems are defined as follows (we only list the combinatorial generalizations; the non-combinatorial variants can be "derived" by using the identity function as κ).

$\mathscr{R}$-COMB-CCAC
Input: An election (C, V), a set A of unregistered candidates such that the voters from V have preference orders over $C \cup A$, a preferred candidate $p \in C$, a bundling function $\kappa \colon A \to \mathscr{P}(A)$, and an integer k.
Question: Is there a subset $A' \subseteq A$ with $|A'| \leq k$ such that $p \in \mathscr{R}(C \cup \kappa(A'), V)$?

$\mathscr{R}$-COMB-CCDC
Input: An election (C, V), a preferred candidate $p \in C$, a bundling function $\kappa \colon C \to \mathscr{P}(C)$, and an integer k.
Question: Is there a subset $C' \subseteq C$ with $|C'| \leq k$ such that $p \in \mathscr{R}(C \setminus \kappa(C'), V)$?

The destructive variants of our problems, $\mathscr{R}$-COMB-DCAC and $\mathscr{R}$-COMB-DCDC, are defined analogously, except that we replace the preferred candidate p with the despised candidate d, and we ask whether it is possible to ensure that d is *not* a winner of the election. In the DCDC case, we explicitly disallow deleting any bundle containing the despised candidate. In the standard, non-combinatorial, variants of control we omit the prefix "Comb" and assume that for each candidate c we have $\kappa(c) = \{c\}$, omitting the bundling function in the respective discussions.

We argue, informally, that our model of combinatorial candidate control is about the simplest that one can think of. Indeed, in a scenario with m candidates, there are at most m corresponding bundles of candidates that can be added/deleted. In real life, however, one might expect many more. Notably, even such a simple model turns out to be computationally difficult and, thus, we believe that it is instructive to consider such a simplified model first. Moreover, in many cases (for example, combinatorial constructive control by deleting candidates) we already obtain very strong hardness results.

3.3.1. Overview of Our Results and Proof Techniques

In this section, we review our results, discuss some interesting patterns we identified in them, and provide a high-level overview of our proof techniques. This section can be viewed as a guide for helping the reader better understand the implications of our results, presented in Table 3.1 and in Table 3.2, and to get an intuitive understanding of our means of obtaining them. We begin by discussing the results regarding the Approval-based voting rules, then discuss the results for the other voting rules, and finally describe our proof techniques.

Before we discuss the results presented in the tables, we note the following:

1) for t-Approval and t-Veto we mean $t \geq 2$;
2) for Copeland$^\alpha$, we mean $0 \leq \alpha \leq 1$;
3) results marked with ♣ and ♢ are due to Faliszewski et al. [77, 75], those marked with ♡ are due to Loreggia et al. [108], and those marked with ♠ follow from the work of Betzler and Uhlmann [9] for $\alpha \in \{0, 1\}$ and are due to this chapter for the remaining values;
4) question mark (?) means that the computational complexity is still open.

3.3.2. Results for Approval-Based Voting Rules

Approval-based rules are perhaps the simplest and the most frequently used ones, so results regarding them are of particular interest. Further, they exhibit quite interesting behavior with respect to the complexity of candidate control parameterized by the number of voters. For the Approval-based voting rules discussed here, the results regarding Plurality (and, to some extent, Veto), are by far the most important ones.

In terms of standard complexity theory, all constructive and destructive candidate control problems for Plurality and Veto are NP-complete (see the works of Bartholdi et al. [5] and Hemaspaandra et al. [95] for the results regarding the Plurality rule; the results for Veto are easy to derive based on those for Plurality and, for example, Elkind et al. [65] showed that Veto-CCAC is NP-complete[2]).

Yet, if we consider parameterization by the number of voters, the results change quite drastically. We make the following observations.

[2]For the case of t-Approval, $t \geq 2$, Faliszewski et al. [79] showed that t-Approval-CCAC and t-Approval-CCDC are NP-complete. For the case of t-Veto, $t \geq 2$, our W[1]-hardness proofs double as NP-completeness proofs (technically, proving W[1]-hardness does not automatically imply NP-hardness, but it does hold for our reductions).

Table 3.1. The complexity of candidate control (constructive (CC) and destructive (DC), adding candidates (AC) and deleting candidates (DC)) problems for Approval-based voting rules parameterized by the number of voters.

Problem	Plurality	Veto	t-Approval	t-Veto
$\mathscr{R}$-CCAC	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
$\mathscr{R}$-CCDC	FPT	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
$\mathscr{R}$-DCAC	FPT	FPT	FPT	FPT
$\mathscr{R}$-DCDC	FPT	FPT	FPT	FPT
$\mathscr{R}$-COMB-CCAC	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
$\mathscr{R}$-COMB-CCDC	Para-NP-h (1)	Para-NP-h (1)	Para-NP-h (1)	Para-NP-h (1)
$\mathscr{R}$-COMB-DCAC	FPT	FPT	W[1]-h / XP	? / XP
$\mathscr{R}$-COMB-DCDC	Para-NP-h (3)	Para-NP-h (1)	Para-NP-h (2)	Para-NP-h (1)

1) The results for Plurality and Veto are no longer the same (specifically, Plurality-CCDC is FPT whereas Veto-CCDC is W[1]-hard). This is quite surprising given both the similarities between these rules and the fact that their standard complexity-theoretic results are almost identical.

2) For all t-Approval and t-Veto rules (including Plurality and Veto), the destructive non-combinatorial candidate control problems are fixed parameter tractable. The constructive variants of these problems—with the exception of Plurality-CCDC—are W[1]-hard.

3) For the combinatorial setting there is a sharp difference between control by adding candidates and control by deleting candidates. Specifically, for both the Plurality rule and the Veto rule, COMB-DCAC is fixed-parameter tractable and COMB-CCAC is W[1]-hard, whereas Comb-CCDC and COMB-DCDC are Para-NP-hard. For t-Approval and t-Veto with $t \geq 2$, the patterns are even simpler. All the adding-candidates cases are W[1]-hard (with one open case), and all the deleting-candidates cases are Para-NP-hard.

Table 3.2. The complexity of candidate control (constructive (CC) and destructive (DC), adding candidates (AC) and deleting candidates (DC)) problems for the Maximin, Borda, and the Copeland voting rules parameterized by the number of voters.

Problem	Maximin	Borda	Copeland$^\alpha$
$\mathscr{R}$-CCAC	Para-NP-h (10)	Para-NP-h (10)	Para-NP-h (20) ♠
$\mathscr{R}$-CCDC	P ♣	Para-NP-h (10)	Para-NP-h (26) ♠
$\mathscr{R}$-DCAC	P ♣	P ♡	P ♢
$\mathscr{R}$-DCDC	P ♣	P ♡	P ♢
$\mathscr{R}$-COMB-CCAC	Para-NP-h (6)	Para-NP-h (2)	Para-NP-h (3) ♠
$\mathscr{R}$-COMB-CCDC	Para-NP-h (1)	Para-NP-h (1)	Para-NP-h (1) ♠
$\mathscr{R}$-COMB-DCAC	P	Para-NP-h (2)	Para-NP-h (3)
$\mathscr{R}$-COMB-DCDC	Para-NP-h (5)	Para-NP-h (2)	Para-NP-h (3)

We conclude by noting that in each of the W[1]-hard cases discussed here we also provide an XP algorithm. This means that, under the assumption P $\neq$ NP, these cases cannot be strengthened to Para-NP-hardness results; thus, in some sense, our results are tight. It is quite interesting that in the non-combinatorial setting the demarcation line between fixed-parameter tractable and W[1]-hard problems goes along the constructive-vs-destructive axis, whereas for the combinatorial setting the line between W[1]-hard (or, in-FPT for Plurality and Veto) and Para-NP-hard problems goes along the adding-vs-deleting-candidates axis.

3.3.3. Results for Maximin, Borda, and Copeland

The results for the Maximin, the Borda, and the Copeland$^\alpha$ rules are quite different from those for the case of the t-Approval and the t-Veto rules. Here, instead of FPT and W[1]-hardness results we find polynomial-time algorithms and Para-NP-hardness results. Specifically, it has already been reported in the

literature that there are polynomial-time algorithms for destructive candidate control under the Borda rule [108], the Copeland$^\alpha$ rule [75], and the Maximin rule [77]. For constructive candidate control, Para-NP-hardness was already known for Copeland^0 and Copeland^1 [9], while in this chapter we establish the same Para-NP-hardness for the remaining values of α (that is, for $0 < \alpha < 1$), as well as for the Borda rule and for the Maximin rule (in the latter case, only for CCAC; CCDC is known to be in P [77]).

The most interesting results regard the combinatorial setting, where almost all of our problems are Para-NP-hard. The only exception is Maximin-COMB-DCAC, which can be solved in polynomial time using an algorithm that, in essence, is identical to the one for the non-combinatorial setting. Our hardness proofs mostly rely on a set-embedding technique (see the next section), which more-or-less directly embeds instances of SET COVER in our control problems. Due to the generality of this approach, we also prove that for every voting rule $\mathscr{R}$ that satisfies the unanimity principle (that is, for every voting rule that chooses as the unique winner the candidate that is ranked first by all the voters), $\mathscr{R}$-COMB-CCDC is Para-NP-hard.

Summarizing the discussion above, the high-level intuition for our more involved voting rules is that constructive candidate control is hard even in the non-combinatorial setting, whereas destructive candidate control is tractable in the non-combinatorial settings, but becomes Para-NP-hard in the combinatorial settings (the only exception is the Maximin rule for the destructive control by adding candidates cases (Maximin-(Comb)-DCAC)).

3.3.4. Proof Techniques

We believe that one of the most important contributions of this chapter comes from identifying four very general proof techniques for establishing the results presented here. Indeed, we believe that these techniques might be useful in studying the complexity of other election problems (especially, parameterized by the number voters). Next we provide their high-level, intuitive descriptions.

Overview of the Multicolored Clique Proof Technique. This is a technique used for establishing W[1]-hardness results. The main idea is to provide a reduction from the MULTICOLORED CLIQUE (MCC) problem parameterized by the size of the desired clique. Recall that each vertex is associated with one color out of h colors overall and we seek a clique of size h where each vertex is of a different color. The high-level description of our technique is as follows.

We provide a reduction that, given an MCC-instance, introduces a candidate for each vertex and two candidates for each edge. We have to ensure that only the successful control actions add exactly the candidates (for some cases, delete all but exactly the candidates) which correspond to a multicolored clique (note that we mean both the candidates corresponding to the vertices of the clique as well as the candidates corresponding to the edges connecting them).

We enforce this constraint using pairs of carefully crafted votes such that if we have two vertices but not an edge between them, then some candidate receives one more point than it should have for our preferred candidate to win. Note that the colors help us to upper-bound the number of voters needed for the construction (specifically, the number of voters created in the construction depends only on the parameter, therefore giving rise to the parameterized hardness). Formal proofs using this technique are given in Section 3.4.

Overview of the Cubic Vertex Cover Proof Technique. This is a technique used for establishing Para-NP-hardness results for non-combinatorial constructive candidate control problems. The main idea is to provide a reduction from the CUBIC VERTEX COVER (CVC) problem. Recall that this problem is a variant of the standard VERTEX COVER problem where the input graph is guaranteed to be cubic (that is, such that each vertex in it is incident to exactly three edges).

The crucial observation used in this technique is that the edges of a cubic graph can be partitioned into four disjoint matchings. This fact allows us to encode all information regarding the graph in a constant number of votes, in a way that ensures that the actions of adding or deleting candidates correspond to covering edges. Formal proofs using this technique are given in Section 3.5.

Overview of the Set-Embedding Proof Technique. This is a fairly simple technique for showing Para-NP-hardness results for combinatorial control by adding or deleting candidates. The idea is to reduce from the standard SET COVER problem using the bundling function to encode the given sets.

Due to the power of bundling, a constant number of voters suffices for the reduction. Formal proofs using this technique are given in Section 3.6.

Overview of the Signature Proof Technique. This is a group of two similar techniques for showing FPT results for the case of destructive control under the t-Approval and the t-Veto rules.

The first technique in this group is applicable for problems of destructive control by adding candidates. The main idea is to group candidates by finding some equivalences between them such that it will not make a difference which candidate from a specific equivalence class we add. In other words, often it is possible to limit the number of candidates that one has to consider by identifying their most crucial properties (such as the subsets of voters where the candidates are ranked ahead of some given candidate); we refer to these properties as *signatures*. Our fixed-parameter tractability results follow by upper-bounding the number of such equivalence classes as a function solely depending on the number of voters.

The second technique is of similar nature and applies to problems of destructive control by deleting candidates. Formal proofs using this technique are given in Section 3.7.

Almost all proofs in this chapter follow by applying one of the four techniques above. The few remaining ones follow by direct arguments and are given in Section 3.8. The following sections are ordered by the proof technique employed and present the most notable results. Proofs omitted from the main text of this chapter are presented in the appendix to this chapter.

3.4. Multicolored Clique Proof Technique

We start our technical discussion by describing the Multicolored Clique proof technique. We apply it to show W[1]-hardness of candidate control problems for some cases for the t-Approval and the t-Veto rules. Indeed, all the W[1]-hardness proofs in this chapter are based on this technique. Specifically, we prove the following statements (throughout this chapter, all results are for the parameterization by the number of voters):

1) For each fixed integer $t \geq 1$ and for each voting rule $\mathscr{R} \in \{t\text{-Approval}, t\text{-Veto}\}$, $\mathscr{R}$-CCAC (and therefore also $\mathscr{R}$-COMB-CCAC) is W[1]-hard.

2) For each fixed integer $t \geq 2$ and for each voting rule $\mathscr{R} \in \{\text{Veto}, t\text{-Approval}, t\text{-Veto}\}$, $\mathscr{R}$-CCDC is W[1]-hard.

3) For each fixed integer $t \geq 2$, t-Veto-COMB-DCAC is W[1]-hard.

Recall the MULTICOLORED CLIQUE problem, which is W[1]-hard with respect to the solution size even on regular graphs [57].

MULTICOLORED CLIQUE
Input: An undirected graph whose vertices are colored in h colors and an integer h.
Question: Is there a set of h pairwise adjacent vertices such that each vertex is of different color?

All the results in this section follow by reductions from the MULTICOLORED CLIQUE problem and are quite similar in spirit. Thus, we start by providing some common notation and observations for all of them.

Let $I = (G,h)$ be a MULTICOLORED CLIQUE instance with graph G and an integer h. The vertices of G are partitioned into h sets, $V_1(G),\dots,V_h(G)$, each containing all vertices colored with a given color. Without loss of generality, we assume that each $V_i(G)$ contains the same number of vertices, denoted by n', and we rename the vertices so that for each color i, $1 \le i \le h$, we have $V_i(G) = \{v_1^{(i)},\dots,v_{n'}^{(i)}\}$. The task is to decide whether there is a clique of order h where each vertex comes from a different set $V_i(G)$. Moreover, and without loss of generality, we assume that each edge in G connects vertices with different colors, that the input graph contains at least two vertices, and that this graph is connected.

In our reductions, given an instance $I = (G,h)$, we build elections with the following candidates related to the graph G (in addition to the candidates specific to a particular reduction). For each vertex $v \in V(G)$, we introduce a candidate denoted by the same symbol. For each edge $e = \{u,v\}$, we introduce two candidates, (u,v) and (v,u); indeed, while our original graph is undirected, for our construction we treat each undirected edge as two directed ones, one in each direction.

In the description of our preference orders, we will use the following orders over subsets of candidates. For each color i, when we write $V_i(G)$ in a preference order, we mean the order

$$v_1^{(i)} \succ v_2^{(i)} \succ \cdots \succ v_{n'}^{(i)}.$$

For each color i, each vertex $v_\ell^{(i)} \in V_i(G)$, and each color j, $j \neq i$, we write $L(v_\ell^{(i)},j)$ to denote the order obtained from

$$(v_\ell^{(i)},v_1^{(j)}) \succ \cdots \succ (v_\ell^{(i)},v_{n'}^{(j)})$$

by removing those candidates $(v_\ell^{(i)},v_h^{(j)})$ for which there is no edge $\{v_\ell^{(i)},v_h^{(j)}\}$ in G. Intuitively, $L(v_\ell^{(i)},j)$ lists all edge candidates for edges which have endpoint $v_\ell^{(i)}$ and go to vertices of color j (the particular order of these edges in $L(v_\ell^{(i)},j)$ is irrelevant for our constructions).

Similarly, for each pair of colors i and j, $1 \le i,j \le h$, $i \ne j$, we write $E(i,j)$ to mean the order

$$L(v_1^{(i)},j) \succ L(v_2^{(i)},j) \succ \cdots \succ L(v_{n'}^{(i)},j).$$

Intuitively, $E(i,j)$ lists all the edge candidates between the vertices from $V_i(G)$ and the vertices from $V_j(G)$ (note, however, that $E(i,j)$ and $E(j,i)$ are different).

The following two preference orders are crucial for the MULTICOLORED CLIQUE technique. For each pair of colors, i,j, $1 \le i,j \le h$, $i \ne j$, we define $R(i,j)$ and $R'(i,j)$ as follows:

$$R(i,j)\colon v_1^{(i)} \succ L(v_1^{(i)},j) \succ v_2^{(i)} \succ L(v_2^{(i)},j) \succ \cdots \succ v_{n'}^{(i)} \succ L(v_{n'}^{(i)},j),$$

$$R'(i,j)\colon L(v_1^{(i)},j) \succ v_1^{(i)} \succ L(v_2^{(i)},j) \succ v_2^{(i)} \succ \cdots \succ L(v_{n'}^{(i)},j) \succ v_{n'}^{(i)}.$$

The idea behind $R(i,j)$ and $R'(i,j)$ is as follows. Consider a setting where u is a vertex of color i and v is a vertex of color j (that is, $u \in V_i(G)$ and $v \in V_j(G)$). Note that $R(i,j)$ and $R'(i,j)$ contain all candidates from $V_i(G)$ and $E(i,j)$. If we restrict these two preference orders to candidates u and (u,v), then they will become $u \succ (u,v)$ and $(u,v) \succ u$. That is, in this case they are reverses of each other. However, if we restrict them to u and some candidate (u',v') different from (u,v), then either they will be both $u \succ (u',v')$ or they will be both $(u',v') \succ u$. Using this effect is at the heart of our constructions.

With the above setup, we are ready to prove the results of this section. Here we give the most interesting examples of proofs; the remaining ones are given in the appendix to this chapter.

Theorem 3.1. PLURALITY-CCAC*, parameterized by the number n of voters, is* W[1]*-hard.*

Proof. Let $I = (G,h)$ be our instance of MULTICOLORED CLIQUE with graph G and an integer h. Let the notation be the same as described above the theorem statement. We form an instance I' of Plurality-CCAC as follows. Let the registered candidate set C consist of two candidates, p and d, and let the set A of unregistered candidates contain all the vertex candidates and all the edge candidates for G. Let p be the preferred candidate. We construct the election such that the current winner is d. We introduce the following groups of voters (when we write "$\cdots$" in a preference order, we mean listing all the remaining candidates in some arbitrary order; below the proof we also provide an example of applying this construction).

1) For each color i, $1 \leq i \leq h$, we have one voter with preference order of the form

$$V_i(G) \succ d \succ \cdots \succ p.$$

2) For each pair of colors i, j ($1 \leq i, j \leq h$, $i \neq j$), we have $h-1$ voters with preference order of the form

$$E(i,j) \succ d \succ \cdots \succ p.$$

3) For each pair of colors i, j ($1 \leq i, j \leq h$, $i \neq j$), we have two voters, one with preference order of the form

$$R(i,j) \succ d \succ \cdots \succ p,$$

and one with preference order of the form

$$R'(i,j) \succ d \succ \cdots \succ p.$$

4) We have h voters with preference order of the form

$$d \succ \cdots \succ p,$$

and h voters with preference order of the form

$$p \succ \cdots \succ d.$$

Note that the total number of voters is $3h + 2(h+1) \cdot \binom{h}{2}$ and that the current winner is d with the score of $(2h + 2(h+1) \cdot \binom{h}{2})$ points. We set the budget k to be $h + 2\binom{h}{2} = h^2$. This completes the construction. It is easy to see that this is a parameterized reduction.

We now claim that it is possible to ensure that p becomes a winner by adding at most k candidates if and only if I is a yes-instance.

First, assume that I is a yes-instance of MULTICOLORED CLIQUE and let Q be a size-h subset of vertices that forms a multicolored clique in I. It is easy to see that if we add to our election the h vertex candidates from Q and all the edge candidates that correspond to edges between the candidates from Q, then, in the resulting election, each candidate (including p and d) will have h points (for example, each of the added vertex candidates will receive one point from the first

group of voters and $h-1$ points from the third group of voters). Thus, everyone will win.

Now, assume that it is possible to ensure p's victory by adding at most k candidates. Let A' be a subset of candidates such that $|A'| \leq k = h + 2\binom{h}{2}$ and adding the candidates from A' to the election ensures that p is a winner. Irrespective of the contents of the set A', in the resulting election p will have h points. Thus, it follows that d must lose all the points from the first three groups of voters. This implies that for each color i, $1 \leq i \leq h$, A' contains exactly one candidate from $V_i(G)$ and for each pair of colors i, j $(1 \leq i, j \leq h,\ i \neq j)$, A' contains exactly one edge candidate (u,v) such that $u \in V_i(G)$ and $v \in V_j(G)$ (The fact that A' contains at least one candidate of each type follows since otherwise d would have more than h points; the fact that it contains exactly one of each type follows by a simple counting argument).

Now it suffices to prove that for each pair of vertex candidates $u, v \in A'$, we also have $(u,v) \in A'$. To show this, first observe that there is a total of $h + 2(h+1)\cdot\binom{h}{2} = h^3 = h \cdot k$ voters from the first three groups that will give points to the newly added candidates. Since each added candidate can have at most h points, it follows that $|A'| = k$ and each added candidate receives exactly h points. By the observations regarding preference orders $R(i,j)$ and $R'(i,j)$, if $u, v \in A'$ but $(u,v) \notin A'$, then either some vertex candidate or some edge candidate would be ranked first by at least two voters from the third group. If this were the case for an edge candidate, then—including the votes from the second group—this candidate would have more than h points and p would not be a winner. If this were the case for a vertex candidate (and none of the edge candidates were ranked first by more than one of the voters in the third group), then this vertex candidate would receive at least h points from the voters in the third group and one point from the voters in the first group. Again, p would not be a winner. Thus, it must be that for each pair of candidates $u, v \in A'$ we have $(u,v) \in A'$. However, this proves that G has a multicolored clique of order h. □

We provide an example for the reduction described in the proof of Theorem 3.1. The input graph is depicted in Figure 3.4 and we take $h := 3$. The constructed election is presented in Table 3.3 and the election after we add the candidates:

$$v_1^{(1)}, v_2^{(2)}, v_2^{(3)},$$
$$(v_1^{(1)}, v_2^{(2)}), (v_1^{(1)}, v_2^{(3)}), (v_2^{(2)}, v_2^{(3)}),$$
$$(v_2^{(2)}, v_1^{(1)}), (v_2^{(3)}, v_1^{(1)}), (v_2^{(3)}, v_2^{(2)}),$$

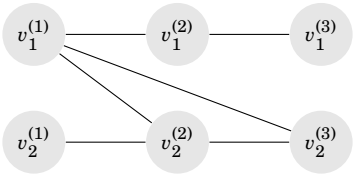


Figure 3.4. The input graph considered in our example for Theorem 3.1.

corresponding to the vertices and edges of the multicolored clique $\{v_1^{(1)}, v_2^{(2)}, v_2^{(3)}\}$, is presented in Table 3.4.

We now consider the Veto-CCAC case, which, despite of having a relatively simple modification of the last proof, is quite intriguing.

Theorem 3.2. VETO-CCAC*, parameterized by the number n of voters, is* W[1]*-hard.*

Proof. One can use the same construction (and proof) as for the Plurality-CCAC case (Theorem 3.1), but with the following modifications (note that the order is important, that is, we perform the second modification only after we have performed the first modification):

1) swap the occurrences of p and d in every vote, and

2) reverse each vote.

In effect, prior to adding candidates, p is vetoed by all but h voters and d is vetoed by exactly h voters. It is easy to verify that if we add vertex candidates and edge candidates that correspond to a multicolored clique, then every candidate in the election is vetoed by exactly h voters and all the candidates are winners.

For the reverse direction, analogously as in the Plurality case (Theorem 3.1), we note that we have to add exactly one vertex candidate of each color and exactly one edge candidate for each (ordered) pair of colors (otherwise p would receive more than h vetoes). To argue that for each pair of vertex candidates u and v that we add, we also have to add edge candidate (u,v), we use the same reasoning as in the Plurality case, but pointing out that if some candidate receives two vetoes from the third group of voters, then some other one receives, altogether, fewer than h vetoes and p is not a winner. □

Table 3.3. The election constructed in the proof of Theorem 3.1 for our example, that is, for the input graph depicted in Figure 3.4. The registered candidates (typeset in bold) are d and p. The figure shows the specific voters, grouped by the groups as defined in Theorem 3.1, and by the color(s) for which the voters correspond to.

Group	Color(s)	Preference order
1	(1)	$v_1^{(1)} \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$
1	(2)	$v_1^{(2)} \succ v_2^{(2)} \succ \mathbf{d} \succ \cdots$
1	(3)	$v_1^{(3)} \succ v_2^{(3)} \succ \mathbf{d} \succ \cdots$
2	(1,2)	$(v_1^{(1)}, v_1^{(2)}) \succ (v_1^{(1)}, v_2^{(2)}) \succ (v_2^{(1)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(2,1)	$(v_1^{(2)}, v_1^{(1)}) \succ (v_2^{(2)}, v_1^{(1)}) \succ (v_2^{(2)}, v_2^{(1)}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(1,3)	$(v_1^{(1)}, v_2^{(3)}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(3,1)	$(v_2^{(3)}, v_1^{(1)}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(2,3)	$(v_1^{(2)}, v_1^{(3)}) \succ (v_2^{(2)}, v_2^{(3)}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(3,2)	$(v_1^{(3)}, v_1^{(2)}) \succ (v_2^{(3)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$ (2 copies)
3	(1,2)	$v_1^{(1)} \succ (v_1^{(1)}, v_1^{(2)}) \succ (v_1^{(1)}, v_2^{(2)}) \succ v_2^{(1)} \succ (v_2^{(1)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$
3	(1,2)	$(v_1^{(1)}, v_1^{(2)}) \succ (v_1^{(1)}, v_2^{(2)}) \succ v_1^{(1)} \succ (v_2^{(1)}, v_2^{(2)}) \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$
3	(2,1)	$v_1^{(2)} \succ (v_1^{(2)}, v_1^{(1)}) \succ v_2^{(2)} \succ (v_2^{(2)}, v_1^{(1)}) \succ (v_2^{(2)}, v_2^{(1)}) \succ \mathbf{d} \succ \cdots$
3	(2,1)	$(v_1^{(2)}, v_1^{(1)}) \succ v_1^{(2)} \succ (v_2^{(2)}, v_1^{(1)}) \succ (v_2^{(2)}, v_2^{(1)}) \succ v_2^{(2)} \succ \mathbf{d} \succ \cdots$
3	(1,3)	$v_1^{(1)} \succ (v_1^{(1)}, v_2^{(3)}) \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$
3	(1,3)	$(v_1^{(1)}, v_2^{(3)}) \succ v_1^{(1)} \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$
3	(3,1)	$v_1^{(3)} \succ v_2^{(3)} \succ (v_2^{(3)}, v_1^{(1)}) \succ \mathbf{d} \succ \cdots$
3	(3,1)	$v_1^{(3)} \succ (v_2^{(3)}, v_1^{(1)}) \succ v_2^{(3)} \succ \mathbf{d} \succ \cdots$
3	(2,3)	$v_1^{(2)} \succ (v_1^{(2)}, v_1^{(3)}) \succ v_2^{(2)} \succ (v_2^{(2)}, v_2^{(3)}) \succ \mathbf{d} \succ \cdots$
3	(2,3)	$(v_1^{(2)}, v_1^{(3)}) \succ v_1^{(2)} \succ (v_2^{(2)}, v_2^{(3)}) \succ v_2^{(2)} \succ \mathbf{d} \succ \cdots$
3	(3,2)	$v_1^{(3)} \succ (v_1^{(3)}, v_1^{(2)}) \succ v_2^{(3)} \succ (v_2^{(3)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$
3	(3,2)	$(v_1^{(3)}, v_1^{(2)}) \succ v_1^{(3)} \succ (v_2^{(3)}, v_2^{(2)}) \succ v_2^{(3)} \succ \mathbf{d} \succ \cdots$
4		$\mathbf{d} \succ \cdots$ (3 copies)
4		$\mathbf{p} \succ \cdots$ (3 copies)

Table 3.4. The election from Table 3.3 with both the registered candidates and the added candidates corresponding to picking the multicolored clique $\{v_1^{(1)}, v_2^{(2)}, v_2^{(3)}\}$ typeset in bold. The figure shows the specific voters, grouped by the groups as defined in Theorem 3.1, as well as by the color(s) for which the voters correspond to.

Group	Color(s)	Preference order
1	(1)	$\mathbf{v_1^{(1)}} \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$
1	(2)	$v_1^{(2)} \succ \mathbf{v_2^{(2)}} \succ \mathbf{d} \succ \cdots$
1	(3)	$v_1^{(3)} \succ \mathbf{v_2^{(3)}} \succ \mathbf{d} \succ \cdots$
2	(1,2)	$(v_1^{(1)}, v_1^{(2)}) \succ (\mathbf{v_1^{(1)}}, \mathbf{v_2^{(2)}}) \succ (v_2^{(1)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(2,1)	$(v_1^{(2)}, v_1^{(1)}) \succ (\mathbf{v_2^{(2)}}, \mathbf{v_1^{(1)}}) \succ (v_2^{(2)}, v_2^{(1)}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(1,3)	$(\mathbf{v_1^{(1)}}, \mathbf{v_2^{(3)}}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(3,1)	$(\mathbf{v_2^{(3)}}, \mathbf{v_1^{(1)}}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(2,3)	$(v_1^{(2)}, v_1^{(3)}) \succ (\mathbf{v_2^{(2)}}, \mathbf{v_2^{(3)}}) \succ \mathbf{d} \succ \cdots$ (2 copies)
2	(3,2)	$(v_1^{(3)}, v_1^{(2)}) \succ (\mathbf{v_2^{(3)}}, \mathbf{v_2^{(2)}}) \succ \mathbf{d} \succ \cdots$ (2 copies)
3	(1,2)	$\mathbf{v_1^{(1)}} \succ (v_1^{(1)}, v_1^{(2)}) \succ (\mathbf{v_1^{(1)}}, \mathbf{v_2^{(2)}}) \succ v_2^{(1)} \succ (v_2^{(1)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$
3	(1,2)	$(v_1^{(1)}, v_1^{(2)}) \succ (\mathbf{v_1^{(1)}}, \mathbf{v_2^{(2)}}) \succ \mathbf{v_1^{(1)}} \succ (v_2^{(1)}, v_2^{(2)}) \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$
3	(2,1)	$v_1^{(2)} \succ (v_1^{(2)}, v_1^{(1)}) \succ \mathbf{v_2^{(2)}} \succ (\mathbf{v_2^{(2)}}, \mathbf{v_1^{(1)}}) \succ (v_2^{(2)}, v_2^{(1)}) \succ \mathbf{d} \succ \cdots$
3	(2,1)	$(v_1^{(2)}, v_1^{(1)}) \succ v_1^{(2)} \succ (\mathbf{v_2^{(2)}}, \mathbf{v_1^{(1)}}) \succ (v_2^{(2)}, v_2^{(1)}) \succ \mathbf{v_2^{(2)}} \succ \mathbf{d} \succ \cdots$
3	(1,3)	$\mathbf{v_1^{(1)}} \succ (\mathbf{v_1^{(1)}}, \mathbf{v_2^{(3)}}) \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$
3	(1,3)	$(\mathbf{v_1^{(1)}}, \mathbf{v_2^{(3)}}) \succ \mathbf{v_1^{(1)}} \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$
3	(3,1)	$v_1^{(3)} \succ \mathbf{v_2^{(3)}} \succ (\mathbf{v_2^{(3)}}, \mathbf{v_1^{(1)}}) \succ \mathbf{d} \succ \cdots$
3	(3,1)	$v_1^{(3)} \succ (\mathbf{v_2^{(3)}}, \mathbf{v_1^{(1)}}) \succ \mathbf{v_2^{(3)}} \succ \mathbf{d} \succ \cdots$
3	(2,3)	$v_1^{(2)} \succ (v_1^{(2)}, v_1^{(3)}) \succ \mathbf{v_2^{(2)}} \succ (\mathbf{v_2^{(2)}}, \mathbf{v_2^{(3)}}) \succ \mathbf{d} \succ \cdots$
3	(2,3)	$(v_1^{(2)}, v_1^{(3)}) \succ v_1^{(2)} \succ (\mathbf{v_2^{(2)}}, \mathbf{v_2^{(3)}}) \succ \mathbf{v_2^{(2)}} \succ \mathbf{d} \succ \cdots$
3	(3,2)	$v_1^{(3)} \succ (v_1^{(3)}, v_1^{(2)}) \succ \mathbf{v_2^{(3)}} \succ (\mathbf{v_2^{(3)}}, \mathbf{v_2^{(2)}}) \succ \mathbf{d} \succ \cdots$
3	(3,2)	$(v_1^{(3)}, v_1^{(2)}) \succ v_1^{(3)} \succ (\mathbf{v_2^{(3)}}, \mathbf{v_2^{(2)}}) \succ \mathbf{v_2^{(3)}} \succ \mathbf{d} \succ \cdots$
4		$\mathbf{d} \succ \cdots$ (3 copies)
4		$\mathbf{p} \succ \cdots$ (3 copies)

To see why this result is intriguing, let us consider the following voting rule, that we call *TrueVeto*. Under TrueVeto, a candidate c is a winner if none of the voters ranks c last. It is quite easy to see that TrueVeto-CCAC is NP-complete (by a reduction from the SET COVER problem, for example), but it is also in FPT (when parameterized by the number of voters; an algorithm similar to that for Plurality-DCAC, based on our Signatures technique, works; see Section 3.3.4). If a Veto election contained more candidates than voters, then at least one candidate would never be vetoed and, in effect, the election would be held according to the TrueVeto rule. This means that in the proof which shows that Veto-CCAC is W[1]-hard, the election has fewer candidates than voters, even after adding the candidates (and keep in mind that the number of voters is the parameter!). Thus, the hardness of the problem lays in picking few spoiler candidates to add from a large group of them. If we were adding more candidates than we had voters, the problem would be FPT.

Now, we move on to the deleting candidates case. We will give a detailed proof for Veto-CCDC (on the one hand, Plurality-CCDC is in FPT, and on the other hand, it is instructive to see a detailed proof for the case of Veto). The proof still follows the general ideas of the multicolored clique technique, but since we delete candidates, we have to adapt the approach.

Theorem 3.3. VETO-CCDC*, parameterized by the number n of voters, is* W[1]*-hard.*

Proof. We provide a parameterized reduction from the MULTICOLORED CLIQUE problem. Let $I = (G,h)$ be our input instance with graph G and an integer h, and let the notation be as described in the introduction to the section. We form an instance I' of Veto-CCDC as follows. Let the registered candidate set C consist of all the vertex candidates plus all the edge candidates for G, plus the preferred candidate p. We construct the following groups of voters (set $H = 2\binom{h}{2} = h \cdot (h-1)$):

1) For each color i, $1 \leq i \leq h$, we introduce $2H - (h-1)$ voters with preference order of the form
$$\cdots \succ p \succ V_i(G).$$

2) For each pair of colors i, j ($1 \leq i, j \leq h$, $i \neq j$) we introduce $2H - 1$ voters with preference order of the form
$$\cdots \succ p \succ E(i,j).$$

3) For each pair of colors, i, j ($1 \leq i, j \leq h$, $i \neq j$) we introduce two voters, one with preference order of the form

$$\cdots \succ p \succ R(i,j),$$

and one with preference order of the form

$$\cdots \succ p \succ R'(i,j).$$

4) We introduce $2H$ voters with preference order of the form $\cdots \succ p$.

We set the number k of candidates that can be deleted to $|V(G)| - h + 2|E(G)| - H$ (with the intention that one should delete all the candidates except for p and those corresponding to the vertices and edges of the multicolored clique of order h). This completes the construction. Note that the total number of voters is

$$(2H - (h-1)) \cdot h + (2H-1) \cdot H + H \cdot 2 + 2H \cdot 1 = 2H \cdot (H + h + 1).$$

Since the input graph is connected and contains at least two vertices (which means that the election has more than $H + h + 1$ candidates), there is at least one candidate, either a vertex candidate or an edge candidate, which has fewer than $2H$ vetoes. Thus, p is currently not a winner.

We claim that p can become a winner by deleting at most k candidates if and only if I is a yes-instance. First, it is easy to see that if G contains an order-h multicolored clique and Q is the set of h vertices that form such a clique, then we can ensure that p is a winner. It suffices to delete all candidates from $V(G) \setminus Q$ and all edge candidates except the ones of the form (u,v), where both u and v belong to Q. In effect, each remaining candidate will have $2H$ vetoes and all the candidates will tie for victory. To see this, note that after deleting the candidates, p still receives $2H$ vetoes from the last group of voters. Now, for each color i, $1 \leq i \leq h$, consider the remaining vertex candidate of color i (call this vertex $v^{(i)}$). This candidate receives $2H - (h-1)$ vetoes from the first group of voters. Further, there are exactly $h-1$ voters in the third group that give one veto to $v^{(i)}$ each (these are the voters that correspond to the edges that connect $v^{(i)}$ with the other vertices of the clique). No other voter vetoes $v^{(i)}$. Now, for each pair of colors, i and j, $1 \leq i, j \leq h$, $i \neq j$, consider the two edge candidates, call them (u,v) and (v,u), whose corresponding edges are incident to vertices of color i (candidate u) and color j (candidate v). Both (u,v) and (v,u) still get $2H-1$ vetoes from the second group of voters. It is also easy to see that each of them receives one veto

from the third group of voters (for the case of (u,v), this veto comes from the first voter corresponding to the color choice (i,j), and in the case of v, this veto comes from the first voter corresponding to the color choice (j,i)).

Now we come to the reverse direction. Assume that it is possible to ensure p's victory be deleting at most k candidates. Prior to deleting any candidates, p has $2H$ vetoes and, of course, deleting candidates cannot decrease this number. Thus, we have to ensure that each non-deleted candidate has at least $2H$ vetoes.

Consider two colors i and j, $1 \leq i$, $j \leq h$, $i \neq j$. Each edge candidate (u,v) (where the corresponding vertex u has color i and the corresponding vertex v has color j) appears below p in $2H-1$ votes from the second group of voters and in two votes from the third one. If we keep two edge candidates, say (u',v') and (u'',v'') (where $u',u'' \in V_i(G)$ and $v',v'' \in V_j(G)$), then they are both ranked below p in the same $2H-1$ votes from the second group and in the same two votes from the third one. If neither (u',v') nor (u'',v'') is deleted, then one of them will receive fewer than $2H$ vetoes. This means that for each pair of colors i and j, we have to delete all except possibly one edge candidate of the form (u,v), where $u \in V_i(G)$ and $v \in V_j(G)$.

Similarly, for each color i, $1 \leq i \leq h$, each vertex candidate from $V_i(G)$ appears below p in $2H-(h-1)$ votes from the first group of voters and in $2(h-1)$ votes from the third group. Each two candidates of the same color are ranked below p in the same votes in the first group. Thus, if two vertex candidates of the same color were left in the election (after deleting candidates), then at least one of them would have fewer than $2H$ vetoes.

In consequence, and since we can delete at most $k = |V(G)| - h + 2|E(G)| - H$ candidates, which means that at least $h+H$ candidates except p must remain in the final election, if p is to become a winner, then after deleting the candidates the election must contain exactly one vertex candidate of each color, and exactly one edge candidate for each ordered pair of colors.

Assume that p is among the winners after deleting candidates and consider two remaining vertex candidates u and v, $u \in V_i(G)$ and $v \in V_j(G)$ $(i \neq j)$; they must exist by the previous observation. We claim that edge candidates (u,v) and (v,u) also must remain. Due to symmetry, it suffices to consider (u,v). Careful inspection of voters in the third group shows that if (u,v) is not among the remaining candidates, then (using the observation regarding orders $R(i,j)$ and $R'(i,j)$) we have that the two voters from the third group that correspond to the color pair (i,j) either both rank u last or both rank the same edge candidate last. In either case, a simple counting argument shows that either u has fewer than $2H$ vetoes or the edge candidate corresponding to the ordered color pair (i,j) has

fewer than $2H$ vetoes. In either case, p is not a winner. This shows that the remaining candidates correspond to an order-h multicolored clique. □

Our final example of the application of the multicolored clique technique is for t-Approval-Comb-DCAC for $t \geq 2$. We use an approach very similar to the one used in the preceding proofs, but since we are in the combinatorial setting, we use the bundling function to ensure consistency between the added edge candidates and the added vertex candidates. This is crucial since t-Approval-DCAC is FPT.

Theorem 3.4. *For each fixed integer $t \geq 2$, t-*Approval-Comb-DCAC *is* W[1]*-hard, parameterized by the number n of voters.*

Proof. Given an instance $I = (G, h)$ for the Multicolored Clique problem, we construct an instance of t-Approval-Comb-DCAC. For the combinatorial setting, it is more natural to create only one candidate for each edge, and not two "directed" ones. We let the set of registered candidates be $C = \{p, d\} \cup D$, where D is the following set of dummy candidates:

$$D = \{d_z^{\{i,j\}} : i \in [h] \neq j \in [h], z \in [t-1]\}$$

$$\cup \{d_z^{(i)} : i \in [h], z \in [t-1]\}$$

$$\cup \{e_z^{(i)} : i \in [h], z \in [t-1]\}.$$

Candidate d is the despised one whose victory we want to preclude. We let the set of the additional (unregistered) candidates be

$$A = V(G) \cup E(G).$$

That is, A contains all the vertex candidates and all the edge candidates. We set the bundling function κ so that for each edge candidate e whose corresponding edge is incident to u and v, we have $\kappa(e) = \{e, u, v\}$, and for each vertex candidate v we have $\kappa(v) = \{v\}$. We introduce the following voters:

1) For each pair $\{i, j\} \subset [h]$, $i \neq j$, of distinct colors, we have one voter with the following preference order, where we write $E(\{i, j\})$ to mean an arbitrarily chosen order over the edge candidates that link vertices of color i with those of color j; the first occurrence of "$\cdots$" regards the candidates in $\{d_z^{\{i,j\}} : z \in [t-1]\}$ only):

$$E(\{i, j\}) \succ d_1^{\{i,j\}} \succ \cdots \succ d_{t-1}^{\{i,j\}} \succ d \succ \cdots.$$

Note that in the initial election, d gets a point from this voter, but it is sufficient (and we will make sure that it is also necessary) to add one candidate from $E(\{i,j\})$ to prevent d from getting this point.

2) For each color i, $1 \le i \le h$, we have a voter with the following preference order (recall that $V_i(G)$ consists of all vertex candidates that correspond to the vertices of the same color i; the first occurrence of "$\cdots$" regards the candidates in $\{d_z^{(i)} : z \in [t-2]\}$ only):

$$V_i(G) \succ d_1^{(i)} \succ \cdots \succ d_{t-2}^{(i)} \succ p \succ d_{t-1}^{(i)} \succ \cdots .$$

Note that in the initial election p gets a point from this voter, but if more than one candidate from $V_i(G)$ is added, then p does not gain this point.

3) For each number $i \in [h]$, we have a voter with the following preference order (the first occurrence of "$\cdots$" regards the candidates in $\{e_z^{(i)} : z \in [t-1]\}$ only):

$$d \succ e_1^{(i)} \succ \cdots \succ e_{t-1}^{(i)} \succ \cdots .$$

Note that, altogether, d gets h points from the voters in this group.

First, prior to adding any candidates, d has $h + \binom{h}{2}$ points while p has h points, and each of the dummy candidates has one point. We show next that it is possible to ensure that d is not a winner of this election by adding at most $k := \binom{h}{2}$ (bundles of) candidates if and only if G has a multicolored clique of order h.

An easy calculation shows that if there is a multicolored clique in G, then adding the edge candidates corresponding to the edges of this clique ensures that d is not a winner.

For the reverse direction, assume that it is possible to ensure that d is not a winner by adding at most $\binom{h}{2}$ bundles. It is easy to see that p is the only candidate that can reach a higher score than d this way. For this to happen, d must lose all the points that d initially got from the first group of voters, and p must still get all the points from the second group of voters. Moreover, adding voters corresponding to vertices does not help. Thus, this must correspond to adding $\binom{h}{2}$ edge candidates whose bundles do not add two vertices of the same color. That is, these $\binom{h}{2}$ added edge candidates must correspond to a multicolored clique of order h. □

We conclude this section by mentioning that the following results also follow by applying the Multicolored Clique technique. The proofs are available in the appendix to this chapter.

Theorem 3.5. *For each fixed integer t, $t \geq 2$, t-*APPROVAL-CCAC *is* W[1]*-hard, parameterized by the number n of voters.*

Theorem 3.6. *For each fixed integer t, $t \geq 2$, t-*VETO-CCAC *is* W[1]*-hard, parameterized by the number n of voters.*

Theorem 3.7. *For each fixed integer $t \geq 1$, t-*VETO-CCDC *is* W[1]*-hard, parameterized by the number n of voters.*

Theorem 3.8. 2-APPROVAL-CCDC*, parameterized by the number n of voters, is* W[1]*-hard.*

Theorem 3.9. *For each fixed integer t, $t \geq 3$, t-*APPROVAL-CCDC *is* W[1]*-hard, parameterized by the number n of voters.*

3.5. Cubic Vertex Cover Proof Technique

We now move on to the Cubic Vertex Cover proof technique. Specifically, we use it to obtain the following results (again, all results are for the parameterization by the number of voters):

1) Borda-CCAC and Borda-CCDC are NP-hard (this holds already for elections with ten voters).

2) For each rational α, $0 \leq \alpha \leq 1$, Copeland$^\alpha$-CCAC and Copeland$^\alpha$-CCDC are NP-hard (this holds already for elections with twenty and twenty-six voters, respectively).

3) Maximin-CCAC is NP-hard (this holds already for elections with ten voters).

In other words, we use the Cubic Vertex Cover technique for all our non-combinatorial Para-NP-hardness results. Right in this section we provide proofs for the cases of Borda-CCDV and Maximin-CCAC, while the remaining ones are given in the appendix to this chapter. The reason for doing so is that the proofs for Borda-CCDV and Maximin-CCAC illustrate the essential elements of the technique (as applied both to an adding-candidates case and to a deleting-candidates case, and both to a scoring rule and a Condorcet consistent rule).

Recall the NP-hard CUBIC VERTEX COVER problem [84].

CUBIC VERTEX COVER
Input: A 3-regular undirected graph $G = (V(G), E(G))$ and an integer h.
Question: Is there a set of h vertices such that each edge is incident to at least one vertex from the set?

The general idea behind the Cubic Vertex Cover technique is to prove Para-NP-hardness via reductions from the CUBIC VERTEX COVER problem, using the fact that cubic graphs can be encoded using a constant number of votes.

All the reductions in this section use the following common setup. Let I be an instance of CUBIC VERTEX COVER with a graph G and an integer h. From the classic result by Vizing [141], we know that there is an edge-coloring of G with four colors (that is, it is possible to assign one out of four colors to each edge so that no two edges incident to the same vertex have the same color). Further, it is possible to compute this coloring in polynomial time [122]. This is equivalent to saying that it is possible to decompose the set of G's edges into four disjoint matchings. Our reductions start by computing this decomposition. We rename the edges of G so that these four disjoint matchings are:

$$E^{(1)} = \{e_1^{(1)}, \ldots, e_{m_1}^{(1)}\},$$

$$E^{(2)} = \{e_1^{(2)}, \ldots, e_{m_2}^{(2)}\},$$

$$E^{(3)} = \{e_1^{(3)}, \ldots, e_{m_3}^{(3)}\},$$

$$E^{(4)} = \{e_1^{(4)}, \ldots, e_{m_4}^{(4)}\}.$$

We set $m' = m_1 + m_2 + m_3 + m_4 = |E(G)|$ and $n' = |V(G)|$. For each edge e of the graph, we arbitrarily order its vertices and we write $v'(e)$ and $v''(e)$ to refer to the first vertex and to the second vertex, respectively. For each ℓ, $1 \leq \ell \leq 4$, we write $E^{(-\ell)}$ to mean $E(G) \setminus E^{(\ell)}$. We write $V^{(-\ell)}$ to mean the set of vertices that are not incident to any of the edges in $E^{(\ell)}$.

The crucial point of our approach is to use the above decomposition to create eight votes (two for each matching) that encode the graph. We will now provide

useful notation for describing these eight votes. For each edge e of the graph, we define the following four orders over e, $v'(e)$, and $v''(e)$:

$$P(e)\colon e \succ v'(e) \succ v''(e),$$

$$P'(e)\colon e \succ v''(e) \succ v'(e),$$

$$Q(e)\colon v'(e) \succ v''(e) \succ e,$$

$$Q'(e)\colon v''(e) \succ v'(e) \succ e.$$

For each ℓ, $1 \le \ell \le 4$, we define the following orders over $V(G) \cup E(G)$:

$$A(\ell)\colon P(e_1^{(\ell)}) \succ P(e_2^{(\ell)}) \succ \cdots \succ P(e_{m_\ell}^{(\ell)}),$$

$$A'(\ell)\colon P'(e_{m_\ell}^{(\ell)}) \succ \cdots \succ P'(e_2^{(\ell)}) \succ \cdots \succ P'(e_1^{(\ell)}),$$

$$B(\ell)\colon Q(e_1^{(\ell)}) \succ Q(e_2^{(\ell)}) \succ \cdots \succ Q(e_{m_\ell}^{(\ell)}),$$

$$B'(\ell)\colon Q'(e_{m_\ell}^{(\ell)}) \succ \cdots \succ Q'(e_2^{(\ell)}) \succ \cdots \succ Q'(e_1^{(\ell)}).$$

Note that since each $E^{(\ell)}$ is a matching, each of the above orders is well-defined. The first two of these families of orders (that is, $A(\ell)$ and $A'(\ell)$) will be useful in the hardness proofs for the cases of deleting candidates, and the latter two (that is, $B(\ell)$ and $B'(\ell)$) in the hardness proofs for the cases of adding candidates. The intuitive idea behind orders $A(\ell)$ and $A'(\ell)$ ($B(\ell)$ and $B'(\ell)$) is that, at a high level, they are reverses of each other, but they treat edges and their endpoints in a slightly asymmetric way (we will describe this in detail in the respective proofs).

We are ready to show examples of applying the Cubic Vertex Cover technique. We start with the case of Borda-CCDC, where we present the theorem and its proof first, followed by an example of applying the reduction.

Theorem 3.10. BORDA-CCDC *is* NP*-hard, even for elections with only ten voters.*

Proof. We give a reduction from the CUBIC VERTEX COVER problem. Let I be our input instance that contains graph $G = (V(G), E(G))$ and an integer h. We use the notation introduced in the beginning of the section. We form an election $E = (C, V)$, where $C = \{p, d\} \cup V(G) \cup E(G)$. We introduce the following ten voters:

1) For each ℓ, $1 \le \ell \le 4$, we have the following two voters:

$$\mu(\ell)\colon A(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$

$$\mu'(\ell)\colon p \succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ A'(\ell).$$

2) We have one voter with preference order $p \succ d \succ V(G) \succ E(G)$ and one voter with preference order $\overleftarrow{E(G)} \succ \overleftarrow{V(G)} \succ p \succ d$.

We claim that p can become a winner of this election by deleting at most $k := h$ candidates if and only if there is a vertex cover of size h for G.

Let us first calculate the scores of all the candidates:

1) Candidate p has $5(n' + m') + 6$ points (that is, $4(n' + m' + 1)$ points from the first eight voters and $n' + m' + 2$ points from the last two voters).

2) Each vertex candidate v has $5(n' + m') + 2$ points (for each of the three pairs of voters $\mu(\ell)$, $\mu'(\ell)$, $1 \le \ell \le 4$, such that v is incident to some edge in $E^{(\ell)}$, v gets $n' + m'$ points; v gets $n' + m' + 1$ points from the remaining pair of voters in the first group and, additionally, $n' + m' + 1$ points from the last two voters).

3) Each edge candidate e has $5(n' + m') + 7$ points (that is, $n' + m' + 3$ points from the pair of voters $\mu(\ell)$, $\mu'(\ell)$ such that $e \in E^{(\ell)}$, $n' + m' + 1$ points from each pair of the remaining three pairs of voters in the first group, and $n' + m' + 1$ points from the last two voters.

4) Candidate d has $5(n' + m') + 4$ points (that is, $4(n' + m' + 1)$ points from the voters in the first group and $n' + m'$ points from the last two voters.

Clearly, prior to deleting any of the candidates, p is not a winner since the edge candidates have higher scores. However, the score of p is higher than the score of the vertex candidates and the score of d.

We now describe how deleting candidates affects the scores of the candidates. Let v be some vertex candidate. Deleting v from our election causes the following effects: the score of each edge candidate e such that $v = v'(e)$ or $v = v''(e)$ decreases by six; the score of each remaining candidate decreases by five. This means that if we delete h vertex candidates that correspond to a vertex cover of G, then the scores of p, d, and all the vertex candidates decrease by $5h$, while the scores of

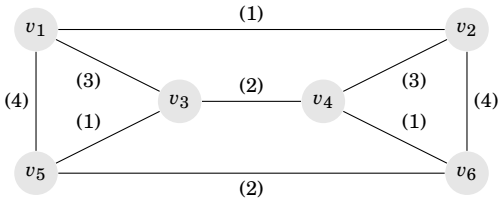


Figure 3.5. The input graph for our example for Theorem 3.10. The numbers in parentheses represent the colors of the edges according to an assumed partition to four colors. For example, the edge $\{v_1, v_3\}$ has the third color.

all edge candidates decrease by at least $5h+1$. As a result, we have p as a winner of the election.

For the reverse direction, assume that it is possible to ensure p's victory by deleting at most h candidates. Deleting candidate d decreases the score of p by six, whereas it decreases the scores of every other candidate by five. Thus, we can assume that there is a solution that does not delete d. Similarly, one can verify that if there is a solution that deletes some edge e, then a solution that is identical but instead of e deletes either $v'(e)$ or $v''(e)$ (it is irrelevant which one) is also correct. We conclude that it is possible to ensure p's victory by deleting at most h vertex candidates. However, by the discussion of the effects of deleting vertex candidates and the fact that prior to any deleting each edge candidate has one point more than p, we have that these at-most-h deleted vertex candidates must correspond to a vertex cover of G. This completes the proof. □

We provide an example for the reduction described in the proof of Theorem 3.10. The input graph is depicted in Figure 3.5 and we take $h := 4$. We present the constructed election in Figure 3.6. The election that results from deleting candidates $\{v_1, v_3, v_4, v_6\}$ that correspond to a vertex cover is presented in Figure 3.7.

Let us now show an application of the Cubic Vertex Cover technique to the case of adding candidates. Specifically, we consider Maximin-CCAC.

Theorem 3.11. MAXIMIN-CCAC *is* NP*-hard, even for elections with only ten voters.*

Proof. We give a reduction from CUBIC VERTEX COVER (we use the notation as provided at the beginning of this section). Given an instance $I = (G, h)$ for

$$\mu(1)\colon \{v_1, v_2\} \succ v_1 \succ v_2 \succ \{v_3, v_5\} \succ v_3 \succ v_5 \succ \{v_4, v_6\} \succ v_4 \succ v_6 \succ E^{(-1)} \succ V^{(-1)} \succ d \succ p$$

$$\mu'(1)\colon p \succ d \succ \overleftarrow{V^{(-1)}} \succ \overleftarrow{E^{(-1)}} \succ \{v_4, v_6\} \succ v_6 \succ v_4 \succ \{v_3, v_5\} \succ v_5 \succ v_3 \succ \{v_1, v_2\} \succ v_2 \succ v_1$$

$$\mu(2)\colon \{v_3, v_4\} \succ v_3 \succ v_4 \succ \{v_5, v_6\} \succ v_5 \succ v_6 \succ E^{(-2)} \succ V^{(-2)} \succ d \succ p$$

$$\mu'(2)\colon p \succ d \succ \overleftarrow{V^{(-2)}} \succ \overleftarrow{E^{(-2)}} \succ \{v_5, v_6\} \succ v_6 \succ v_5 \succ \{v_3, v_4\} \succ v_4 \succ v_3$$

$$\mu(3)\colon \{v_1, v_3\} \succ v_1 \succ v_3 \succ \{v_2, v_4\} \succ v_2 \succ v_4 \succ E^{(-3)} \succ V^{(-3)} \succ d \succ p$$

$$\mu'(3)\colon p \succ d \succ \overleftarrow{V^{(-3)}} \succ \overleftarrow{E^{(-3)}} \succ \{v_2, v_4\} \succ v_4 \succ v_2 \succ \{v_1, v_3\} \succ v_3 \succ v_1$$

$$\mu(4)\colon \{v_1, v_5\} \succ v_1 \succ v_5 \succ \{v_2, v_6\} \succ v_2 \succ v_6 \succ E^{(-4)} \succ V^{(-4)} \succ d \succ p$$

$$\mu'(4)\colon p \succ d \succ \overleftarrow{V^{(-4)}} \succ \overleftarrow{E^{(-4)}} \succ \{v_1, v_5\} \succ v_5 \succ v_1 \succ \{v_2, v_6\} \succ v_6 \succ v_2$$

$$\text{one voter}\colon p \succ d \succ V(G) \succ E(G)$$

$$\text{one voter}\colon \overleftarrow{E(G)} \succ \overleftarrow{V(G)} \succ p \succ d$$

Figure 3.6. The election constructed in the proof of Theorem 3.10 for the input graph depicted in Figure 3.5.

$\mu(1)$: $\{v_1, v_2\} \succ \cancel{v_1} \succ v_2 \succ \{v_3, v_5\} \succ \cancel{v_3} \succ v_5 \succ \{v_4, v_6\} \succ \cancel{v_4} \succ \cancel{v_6} \succ$
$E^{(-1)} \succ V^{(-1)} \succ d \succ p$

$\mu'(1)$: $p \succ d \succ \overleftarrow{V^{(-1)}} \succ \overleftarrow{E^{(-1)}} \succ$
$\{v_4, v_6\} \succ \cancel{v_6} \succ \cancel{v_4} \succ \{v_3, v_5\} \succ v_5 \succ \cancel{v_3} \succ \{v_1, v_2\} \succ v_2 \succ \cancel{v_1}$

$\mu(2)$: $\{v_3, v_4\} \succ \cancel{v_3} \succ \cancel{v_4} \succ \{v_5, v_6\} \succ v_5 \succ \cancel{v_6} \succ$
$E^{(-2)} \succ V^{(-2)} \succ d \succ p$

$\mu'(2)$: $p \succ d \succ \overleftarrow{V^{(-2)}} \succ \overleftarrow{E^{(-2)}} \succ$
$\{v_5, v_6\} \succ \cancel{v_6} \succ v_5 \succ \{v_3, v_4\} \succ \cancel{v_4} \succ \cancel{v_3}$

$\mu(3)$: $\{v_1, v_3\} \succ \cancel{v_1} \succ \cancel{v_3} \succ \{v_2, v_4\} \succ v_2 \succ \cancel{v_4} \succ$
$E^{(-3)} \succ V^{(-3)} \succ d \succ p$

$\mu'(3)$: $p \succ d \succ \overleftarrow{V^{(-3)}} \succ \overleftarrow{E^{(-3)}} \succ$
$\{v_2, v_4\} \succ \cancel{v_4} \succ v_2 \succ \{v_1, v_3\} \succ \cancel{v_3} \succ \cancel{v_1}$

$\mu(4)$: $\{v_1, v_5\} \succ \cancel{v_1} \succ v_5 \succ \{v_2, v_6\} \succ v_2 \succ \cancel{v_6} \succ$
$E^{(-4)} \succ V^{(-4)} \succ d \succ p$

$\mu'(4)$: $p \succ d \succ \overleftarrow{V^{(-4)}} \succ \overleftarrow{E^{(-4)}} \succ$
$\{v_1, v_5\} \succ v_5 \succ \cancel{v_1} \succ \{v_2, v_6\} \succ \cancel{v_6} \succ v_2$

one voter : $p \succ d \succ V(G) \succ E(G)$

one voter : $\overleftarrow{E(G)} \succ \overleftarrow{V(G)} \succ p \succ d$

Figure 3.7. The election from Figure 3.6 with the candidates corresponding to the vertex cover $\{v_1, v_3, v_4, v_6\}$ deleted.

CUBIC VERTEX COVER, we construct an instance for Maximin-CCAC. We let the registered candidate set C be $\{p\} \cup E(G)$, and we let $V(G)$ be the set of unregistered candidates. We construct the following ten voters:

1) For each ℓ, $1 \leq \ell \leq 4$, we have the following two voters:

$$\mu(\ell)\colon B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ p,$$

$$\mu'(\ell)\colon p \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

2) We have one voter with preference order $E(G) \succ p \succ V(G)$ and one voter with preference order $\overleftarrow{E(G)} \succ p \succ \overleftarrow{V(G)}$.

Let E be the thus-constructed election (including all registered and unregistered candidates). We have the following values of the $N_E(\cdot,\cdot)$ function (recall that this function represents the results of the head-to-head contests):

1) For each vertex $v \in V(G)$, we have $N_E(p,v) = 6$ (so $N_E(v,p) = 4$).

2) For each edge $e \in E(G)$, we have $N_E(p,e) = 4$ (so $N_E(e,p) = 6$).

3) For each vertex $v \in V(G)$ and each edge $e \in E(G)$ we have the following: if v is an endpoint of e, then $N_E(v,e) = 6$ (so $N_E(e,v) = 4$), and otherwise we have $N_E(v,e) = 5$ (so $N_E(e,v) = 5$).

4) For each pair of vertices, $v', v'' \in V(G)$, $N_E(v',v'') = 5$.

5) For each pair of edges, $e', e'' \in E(G)$, $N_E(e',e'') = 5$.

In effect, prior to adding the candidates, the score of p is four and the score of each edge candidate is five. Adding a vertex candidate v to the election does not change the score of p, but decreases the score of each edge candidate that has v as an endpoint to four. Further, this added vertex candidate has score four as well. Thus, it is easy to see that it is possible to ensure p's victory by adding at most h candidates if and only if there is a size-h vertex cover for G. □

We conclude the section by mentioning that the following results, whose proofs are given in the appendix to this chapter, also follow by applying the Cubic Vertex Cover technique.

Theorem 3.12. BORDA-CCAC *is* NP*-hard, even for elections with only ten voters.*

Theorem 3.13. *For each rational number α, $0 \leq \alpha \leq 1$,* COPELAND$^\alpha$-CCAC *is* NP*-hard, even for elections with only twenty voters.*

Theorem 3.14. *For each rational number α, $0 \leq \alpha \leq 1$,* COPELAND$^\alpha$-CCDC *is* NP*-hard, even for elections with only twenty-six voters.*

3.6. Set-Embedding Proof Technique for Combinatorial Variants

In this section, we present our Set-Embedding proof technique for the combinatorial variants of our control problems. Specifically, we prove the following statements (again, all results are for the parameterization by the number of voters):

1) For each fixed integer $t \geq 1$ and for each voting rule $\mathscr{R} \in \{t$-Approval, t-Veto, Borda, Copeland$^\alpha$ (for $0 \leq \alpha \leq 1$), Maximin$\}$, both $\mathscr{R}$-COMB-CCDC and $\mathscr{R}$-COMB-DCDC are Para-NP-hard.

2) For each voting rule $\mathscr{R} \in \{$Borda, Copeland$^\alpha$ (for $0 \leq \alpha \leq 1$)$\}$, Maximin$\}$, $\mathscr{R}$-COMB-CCAC is Para-NP-hard.

3) For each voting rule $\mathscr{R} \in \{$Borda, Copeland$^\alpha$ (for $0 \leq \alpha \leq 1$)$\}$, $\mathscr{R}$-COMB-DCAC is Para-NP-hard.

That is, in this section we provide all our Para-NP-hardness results for the combinatorial variants of our problems.

Recall the NP-hard SET COVER problem [84].

> SET COVER
> **Input:** A universe of elements X, a collection $\mathscr{S}$ of sets of elements of X, and a budget h.
> **Question:** Is there a subcollection $\mathscr{S}' \subseteq \mathscr{S}$ of sets such that $|\mathscr{S}'| \leq h$ and $\bigcup_{S \in \mathscr{S}'} S = X$?

All proofs in this section follow by reducing the SET COVER problem to the respective problem in a way which uses the bundling function to encode the sets from the SET COVER instances (hence the name of the technique). We start by providing some common notation and observations common to all of these results.

Let $I = (X, \mathcal{S}, h)$ be an input instance of SET COVER (which is NP-hard). We construct elections with candidate sets that include the elements from X and the sets from $\mathcal{S}$. Specifically, for each element $x_i \in X$, we introduce a candidate with the same name, and for each set $S_j \in \mathcal{S}$, we introduce a candidate named s_j. We denote the set of all element candidates by X_{cand} and denote the set of all set candidates by $\mathcal{S}_{\text{cand}}$. Further, we will typically have candidates p and d. For the constructive cases, p will be the preferred candidate while for the destructive cases, d will be the despised one.

Unless stated otherwise, in each of our proofs we use a bundling function κ defined as follows: for each set candidate s_j, we have $\kappa(s_j) = \{s_j\} \cup \{x_i : x_i \in S_j\}$, and for each non-set candidate c, we have $\kappa(c) = \{c\}$. We refer to this bundling function as the *set-embedding bundling function*.

The general idea of our proofs is that to ensure p's victory (for the constructive cases) or d's defeat (for the destructive cases), one has to add/delete all the candidates from X_{cand}, and due to the bound on the number of candidates that we can add/delete, this has to be achieved by adding/deleting the candidates from $\mathcal{S}_{\text{cand}}$ and relying on the bundling function.

With the above setup ready, we move on to proving our results. Most of the proofs are available in the appendix to this chapter, but for each type of problem (COMB-CCAC, COMB-CCDC, COMB-DCAC, COMB-DCDC) we give one sample proof.

3.6.1. Constructive Control by Deleting Candidates

We start by looking at constructive control by deleting candidates since in this case we obtain a very general hardness result that applies to all the voting rules which satisfy the *unanimity* principle. A rule satisfies the unanimity principle if in each election where a unique candidate c is ranked first by all the voters, this candidate c is the unique winner.

Theorem 3.15. *Let $\mathcal{R}$ be a voting rule that satisfies the unanimity principle. $\mathcal{R}$-*COMB-CCDC *is* NP*-hard, even for the case of elections with just a single voter.*

Proof. Let the notation be as in the introduction to this section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER, we create an instance I' of $\mathcal{R}$-COMB-CCDC as follows. We construct an election $E = (C, V)$ where $C = \{p\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$ and where V contains a single voter with the following preference order:

$$X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}.$$

We use the set-embedding bundling function. We claim that I is a yes-instance of Set Cover if and only if it is possible to ensure p's victory by deleting at most h (bundles of) candidates.

On the one hand, if I is a yes-instance of Set Cover, then I' is a "yes"-instance of $\mathscr{R}$-Comb-CCDC. Indeed, if $\mathscr{S}'$ is a subfamily of $\mathscr{S}$ such that $|\mathscr{S}'| \leq h$ and $\bigcup_{S_j \in \mathscr{S}'} S_j = X$, then it suffices to delete the candidates C' that correspond to the sets in $\mathscr{S}'$ from the election to ensure that p is ranked first (and, by the unanimity of $\mathscr{R}$, is a winner).

On the other hand, assume that I' is a yes-instance of $\mathscr{R}$-Comb-CCDC. Since $\mathscr{R}$ satisfies the unanimity property, the candidate ranked first by the only voter in our election is always the unique winner. This means that if I' is a yes-instance of $\mathscr{R}$-Comb-CCDC, then there is a subset C' of candidates such that $p \notin C'$ and $X \subseteq \bigcup_{c \in C'} \kappa(c)$.

Without loss of generality, we can assume that C' contains only candidates from the set $\mathscr{S}_{\text{cand}}$ (if C' contained some candidate x_i, we could replace x_i with an arbitrary set candidate s_j such that $x_i \in S_j$). However, this immediately implies that setting $\mathscr{S}' := \{S_j : s_j \in C'\}$ results in a set cover of size at most h. Therefore I is a yes-instance of I. □

As Plurality, Borda, Copeland$^\alpha$, and Maximin all satisfy the unanimity property, we conclude the following.

Corollary 3.1. *For each voting rule $\mathscr{R} \in$ {Plurality, Borda, Copeland$^\alpha$, Maximin}, $\mathscr{R}$-Comb-CCDC is* NP*-hard, even for elections with only a single voter.*

By applying minor tweaks to the above construction, we obtain the following results (for the proofs see Section 3.8).

Theorem 3.16. *For each fixed integer $t \geq 2$, t-Approval-Comb-CCDC is* NP*-hard, even for elections with only a single voter.*

Theorem 3.17. *For each fixed integer $t \geq 1$, t-Veto-Comb-CCDC is* NP*-hard, even for elections with only a single voter.*

3.6.2. Destructive Control by Deleting Candidates

While the very general proof for the combinatorial variant of constructive control by deleting candidates is very simple, occasionally our set-embedding proofs become slightly more involved.

For example, our proof that Maximin-COMB-DCDC is NP-hard, even for elections with only a few voters, requires a bit more care. Specifically, we need to define some further candidates to achieve the desired score differences between the current winner d and his defeater p.

Theorem 3.18. MAXIMIN-COMB-DCDC *is* NP*-hard, even for elections with only five voters.*

Proof. Given an instance $(X, \mathscr{S}, h)$ for SET COVER, we construct an instance $(E = (C,V),k)$ for Maximin-COMB-DCDC. We construct an election $E = (C,V)$ where $C := \{p,d,e\} \cup X_{\text{cand}} \cup \mathscr{S}_{\text{cand}}$ and where the voter set consists of the following five voters:

$$\text{one voter: } p \succ d \succ X_{\text{cand}} \succ e \succ \mathscr{S}_{\text{cand}},$$

$$\text{two voters: } d \succ X_{\text{cand}} \succ p \succ e \succ \mathscr{S}_{\text{cand}},$$

$$\text{two voters: } e \succ \overleftarrow{X_{\text{cand}}} \succ p \succ d \succ \overleftarrow{\mathscr{S}_{\text{cand}}}.$$

We set $k := h$. We use the set-embedding bundling functions. We claim that I is a yes-instance of SET COVER if and only if it is possible to ensure that d is not a winner by deleting at most $k = h$ (bundles of) candidates.

The values of the $N_E(\cdot,\cdot)$ function are given in the table below (the entry for row a and column b gives the value of $N_E(a,b)$, which stands for the number of voters which prefer a to b in the election E; we assume $i' \neq i''$ and $j' \neq j''$).

	p	d	e	$x_{i'}$	$s_{j'}$
p	-	3	3	1	5
d	2	-	3	3	5
e	2	2	-	2	5
$x_{i''}$	4	2	3	2 or 3	5
$s_{j''}$	0	0	0	0	2 or 3

We have the following scores of the candidates: p has one point (because of the members of X_{cand}), d has two points (because of p), e has two points (because of p, d, and the members of X_{cand}), the members of X_{cand} have two points each

(because of d), and the members of $\mathscr{S}_{\text{cand}}$ have zero points each (because of all other candidates).

It is easy to verify that if there is a set cover for I of size h, then deleting the set candidates corresponding to the cover deletes all the members of X_{cand} and ensures that p has three points, whereas d has only two. In effect, d certainly is not a winner.

Now consider the other direction. Since deleting a candidate can never decrease the score of any remaining candidate, the only way of making d lose is to increase some remaining candidate's score.

Since for each candidate other than p, at least three voters prefer d to this candidate, only p has any chance of getting a higher score than d. For this to happen, we need to ensure that all members of X_{cand} disappear. As in the previous set-embedding proofs, this is possible to do by deleting at most h candidates only if there is a set cover of size at most h for I. □

For most of the other results it suffices to use proofs very similar to that for Theorem 3.15. However, for the case of t-Approval-COMB-DCDC we have to use either two voters (if $t \geq 2$) or three voters (if $t = 1$ and we are dealing with Plurality). The reason is that if we have a single voter and candidate d is a t-Approval winner, then it is impossible to prevent d from winning by deleting candidates (no matter what we do, d will still have the highest possible score, one). A similar reasoning applies to the case of the Plurality rule with two voters. The proofs of the following results are given in the appendix to this chapter.

Theorem 3.19. PLURALITY-COMB-DCDC *is* NP*-hard, even for elections with only three voters.*

Theorem 3.20. *For each fixed integer $t \geq 2$, t-*APPROVAL-COMB-DCDC *is* NP*-hard, even for elections with only two voters.*

Theorem 3.21. *For each fixed integer $t \geq 1$, t-*VETO-COMB-DCDC *is* NP*-hard, even for elections with only a single voter.*

Theorem 3.22. BORDA-COMB-DCDC *is* NP*-hard, even for elections with only two voters.*

Theorem 3.23. COPELAND$^{\alpha}$-COMB-DCDC *is* NP*-hard, even for elections with only three voters.*

3.6.3. Constructive and Destructive Control by Adding Candidates

For the case of combinatorial control by adding candidates, we give sample proofs for the cases of the Borda rule.

Theorem 3.24. BORDA-COMB-CCAC *and* BORDA-COMB-DCAC *are both* NP*-hard, even for elections with only two voters.*

Proof. We first show the NP-hardness result for Borda-COMB-CCAC and then show how to modify the proof for Borda-COMB-DCAC.

Let the notation be as in the introduction to this section. Given an instance $I := (X, \mathscr{S}, h)$ for SET COVER with $n' := |X_{\text{cand}}|$, we create an instance I' of Borda-COMB-CCAC as follows. We construct the registered candidate set $C = \{d, p\} \cup D$, where $D = \{d_1, \ldots, d_{n'}\}$. We construct the unregistered candidate set $A = X_{\text{cand}} \cup \mathscr{S}_{\text{cand}}$. We construct two voters with the following preference orders:

$$\text{one voter: } d \succ D \succ p \succ \mathscr{S}_{\text{cand}} \succ X_{\text{cand}} \succ \cdots,$$

$$\text{one voter: } p \succ \overleftarrow{X_{\text{cand}}} \succ d \succ \mathscr{S}_{\text{cand}} \succ \overleftarrow{D} \succ \cdots.$$

We use the set-embedding bundling function. We claim that I is a yes-instance of SET COVER if and only if it is possible to ensure p's victory by adding at most h (bundles of) candidates. Note that d gets n' points more than p from the first voter. Given a set cover of size h, we add the corresponding s_j's to the election. Simple calculation shows that in this case p and d tie as winners.

For the reverse direction, note that the relative scores of p and d in the first vote do not change, irrespective of which candidates we add. The relative scores of p and d, however, do change in the second vote in the following way: For each unregistered candidate x_i added to the election, p's score increases by one but d's score remains unchanged. Thus, the only way to ensure that p is a winner is by bringing all the candidates from X_{cand} to the election. Doing so by adding at most h candidates is possible only if there is a size-h cover for I.

The construction for Borda-COMB-DCAC is the same, except that, first, we do not want p to win but d to lose (that is, we define d to be the despised candidate, and second, we define D to have only $n-1$ dummy candidates. □

The proofs for the remaining results are available in the appendix to this chapter. Note that, while technically similar Para-NP-hardness results already

follow from our discussion of the non-combinatorial variants, using the set-embedding technique we can give proofs that use fewer voters.

Theorem 3.25. *For each rational number* α, $1 \leq \alpha \leq 1$, COPELAND$^\alpha$-COMB-DCAC *and* COPELAND$^\alpha$-COMB-CCAC *are* NP*-hard, even for elections with only three voters.*

Theorem 3.26. MAXIMIN-COMB-CCAC *is* NP*-hard, even for elections with only six voters.*

3.7. Signature Proof Technique for Destructive Control

We now move on to the Signatures technique for obtaining FPT algorithms. Specifically, in this section we show the following results:

1) For each fixed integer $t \geq 1$ and for each voting rule $\mathscr{R} \in \{t\text{-Approval}, t\text{-Veto}\}$, $\mathscr{R}$-DCAC is FPT.

2) Plurality-Comb-DCAC and Veto-Comb-DCAC are FPT.

3) For each fixed integer $t \geq 1$ and for each voting rule $\mathscr{R} \in \{t\text{-Approval}, t\text{-Veto}\}$, $\mathscr{R}$-DCDC is FPT.

That is, we apply the technique to the case of destructive candidate control, under t-Approval and t-Veto elections. The main idea of the Signature technique is to identify certain properties of the candidates to be added or deleted that allow us to treat some of them as being equivalent. We refer to these properties as *signatures* and we build FPT algorithms based on the observation that the number of different signatures (in a given context) is a function dependent only on the number of voters.

The results from this section apply, in particular, to the cases of Plurality-DCDC and Veto-DCDC. However, these problems are simple enough that direct algorithms for them that are easier and faster; we provide such algorithms in Section 3.8.

3.7.1. Destructive Control by Adding Candidates

Let us consider an instance of the destructive control by adding candidates problem for the case of t-Approval and t-Veto (for now we focus on the non-

combinatorial variant). The instance consists of the set C of registered candidates, the set A of unregistered candidates, the collection V of n voters, the despised candidate $d \in C$, and an integer k, bounding the number of candidates that we can add. We assume that d is a winner of the given election (C, V) (otherwise we can trivially return True). The general scheme for our FPT algorithms (parameterized by the number n of the voters) is as follows:

1) We guess a candidate $p \in C \cup A \setminus \{d\}$. The role of p is to defeat d, that is, to obtain more points than d. Altogether, there are $|C| + |A| - 1$ candidates to choose from, and we repeat our algorithm for each possible choice of p.

2) For each choice of p, we "kernelize" the input instance. That is, we bound the number of "relevant" candidates by a function dependent only on the parameter n. Then, we search for an optimal solution, in a brute-force manner, over this "kernel" (indeed, the Signature technique regards computing this kernel).[3]

3) We return True if the best solution found adds at most k candidates, while we return False otherwise.

We now describe how to perform the kernelization step. Let us consider the registered candidates first. It turns out that it suffices to focus on a few relevant ones only.

Definition 3.1 (Relevant registered candidates)**.** Fix an integer t, $t \geq 1$, and consider an instance of t-Approval-DCAC. We call a registered candidate *relevant* if this candidate receives at least one point. For the case of t-Veto-DCAC, we call a registered candidate *relevant* if this candidate receives at least one veto. We refer to those candidates that are not relevant as *irrelevant*.

[3] We mention that this kind of kernelization is called *Turing kernelization*. See the works of Binkele-Raible et al. [16] and Schäfer et al. [135] for examples of this concept in the context of graph problems.

For the case of t-Approval-DCAC, we can safely remove all the irrelevant registered candidates. This is so for following two reasons. First, removing an irrelevant candidate does not change the score of any other registered (or later-added) candidate. Second, an irrelevant candidate can never obtain score higher than the despised one since, on the one hand, initially this candidate has score zero, and, on the other hand, under t-Approval, adding candidates never increases the scores of those already registered.

For the case of t-Veto-DCAC, if there are some irrelevant candidates, then we consider the following two possibilities. First, if d receives at least one veto, then d is already not a winner (since the irrelevant candidates receive no vetoes and, thus, defeat d). Second, if d does not receive any vetoes, then this will stay so, irrespective of which candidates we add, and d will remain a winner in any resulting election. Thus, in either case, we can immediately output the correct answer.

Thus, from now on we assume that all the registered candidates are relevant. Furthermore, it is clear that there are at most $t \cdot n$ relevant candidates for each instance of t-Approval-DCAC or t-Veto-DCAC with n voters.

To deal with the unregistered candidates, we introduce the notion of a $\{d,p\}$-*signature* (recall that d is the despised candidate and p is a candidate whose goal is to defeat d). Let c be some unregistered candidate. Each voter can rank c in the following three different ways, relative to p and d: either this voter rank c ahead of both p and d, or below both p and d, or between p and d. A $\{d,p\}$-*signature* for c is a vector which indicates, for each voter, which of these cases holds. Formally, we use the following definition.

Definition 3.2 ($\{d,p\}$-signature)**.** Consider an election $(C \cup A, V)$, a candidate $d \in C$, and a candidate $p \in C \cup A$. Let n be the number of voters in V. A $\{d,p\}$-signature of candidate $c \in (C \cup A) \setminus \{d,p\}$ is a vector $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n) \in [3]^n$ such that, for each voter $v_i \in V$, it holds that:

$$\gamma_i = \begin{cases} 3, & \text{if } v_i \text{ prefers } c \text{ to both } p \text{ and } d, \\ 1, & \text{if } v_i \text{ prefers both } p \text{ and } d \text{ to } c, \\ 2, & \text{otherwise.} \end{cases}$$

The crucial observation, is that, for a given choice of p, it is enough to consider only t candidates of the same $\{d,p\}$-signature.

Lemma 3.1. *Consider an instance $I := ((C,V),A,d \in C,k)$ of t-Approval-DCAC (respectively, t-Veto-DCAC), with the despised candidate d, and with some arbitrarily selected candidate $p \in C \cup A$. Let $\vec{\gamma}$ be some $\{d,p\}$-signature for this election.*

Adding t unregistered candidates with signature $\vec{\gamma}$ has the same effect on the relative scores of p and d as adding more than t candidates with this signature.

Proof. We begin with the case of t-Approval-DCAC. Let n be the number of voters in the instance I. We have $\vec{\gamma} = (\gamma_1, \ldots, \gamma_n)$. Consider the ith voter.

1) If $\gamma_i = 3$, then after adding t (or more) candidates with signature $\vec{\gamma}$, the ith voter will give zero points to both p and d.

2) If $\gamma_i = 1$, then number of points that the ith voter will give to p and d does not depend on the number of candidates with signature $\vec{\gamma}$ that we add.

3) If $\gamma_i = 2$, then for each candidate c with signature $\vec{\gamma}$, the ith voter ranks c between candidates p and d. We have the following two, quite similar, cases to consider: either the ith voter has preference order $p \succ c \succ d$, or the ith voter has preference order $d \succ c \succ p$. In the first case, adding t (or more) candidates with signature $\vec{\gamma}$ will ensure that the ith voter gives zero points to d and gives the same number of points to p as prior to adding these candidates. In the second case, the situation is the same, but with the roles of p and d swapped.

Summarizing the above discussion, we see that adding t candidates with a given signature $\vec{\gamma}$ has the same effect on the relative scores of p and d as adding more candidates of the same signature.

The arguments for the case of t-Veto-DCAC are analogous, and thus, omitted. □

In effect, it suffices to keep at most t candidates with each signature. This results in having at most $t \cdot 3^n$ unregistered candidates. We can now formally describe our kernelization process.

Theorem 3.27. *For each fixed integer t, $t \geq 1$, t-*APPROVAL*-DCAC and t-*VETO*-DCAC admit Turing kernels of size $O(t \cdot 3^n)$, where n is the number of voters.*

Proof. Consider an instance $I := ((C, V), A, d \in C, k)$ of t-Approval-DCAC (respectively, t-Veto-DCAC). Let d be the despised candidate and let n be the number of voters in the instance. As per our discussion above, we can assume that the instances are non-trivial and that all the registered candidates are relevant. Thus, there are at most $t \cdot n$ registered candidates. By Lemma 3.1, for each choice of p, it suffices to consider at most t candidates of the same $\{d, p\}$-signatures, and there are at most 3^n different $\{d, p\}$-signatures.

Altogether, for each choice of candidate p, we produce an instance of t-Approval-DCAC (respectively, t-Veto-DCAC) with at most $t \cdot n$ registered candidates and with at most $t \cdot 3^n$ unregistered ones (for each possible signature we keep up to t arbitrarily chosen unregistered candidates); in each instance we allow to add either the same number of candidates as in I (that is, k), or one less (that is, $k-1$), if p needs to be added to the election as well.

Finally, it holds that it is possible to preclude d from winning if and only if it is possible to do so in at least one of the produced instances. □

Using a brute-force approach on top of the kernelization given in Theorem 3.27, it is possible to solve both t-Approval-DCAC and t-Veto-DCAC in FPT-time: straight-forward application of a brute-force search to each instance produced by Theorem 3.27 results in running time $O^*(\binom{t \cdot 3^n}{k})$, where the O^* notation suppresses polynomial terms. It is easy to see that it never makes sense to add more than $t \cdot n$ candidates (intuitively, if we add more than $t \cdot n$ candidates, then at least one would be irrelevant, and thus, we could as well not add him). Thus we can assume that $k \leq t \cdot n$.

Therefore, we have that the straight-forward brute-force algorithm running on top of Theorem 3.27 has running time $O^*((t \cdot 3^n)^{t \cdot n})$. However, we can improve the time complexity by sacrificing the space complexity.

Theorem 3.28. PLURALITY-DCAC *can be solved in* $O(m \cdot n \cdot 2^n)$ *time, using* $O^*(2^n)$ *space, where* m *is the total number of candidates and* n *is the number of voters.*

Proof. Our algorithm uses a similar general structure as before. We assume that we have a non-trivial instance, where all the registered candidates are relevant. First, we guess a candidate p whose goal is to defeat d and from now on we focus on a situation where we have both p and d, and the goal is to ensure that p gets more points than d. (If p is an unregistered candidate, then we add p to the election, decrease the number of candidates that we can add by one, and proceed as if p was a registered candidate to begin with.)

We define a simplified notion of a candidate's signature. A *simplified-signature* for an unregistered candidate c is a binary vector $\vec{\tau} = (\tau_1, \ldots, \tau_n) \in \{0,1\}^n$ such that the following hold.

$$\tau_i = \begin{cases} 1, & \text{if } v_i \text{ prefers } c \text{ to all registered candidates,} \\ 0, & \text{otherwise.} \end{cases}$$

We define the *simplified-signature* $\vec{\tau}$ of a set A' of unregistered candidates analogously: value one at a given position i means that some candidate from

A' is ranked ahead of all the registered candidates while value zero means that some registered candidate is ranked ahead of all the members of A'.

Let k be the number of candidates that we are allowed to add. Using the notion of simplified-signatures, we maintain a table $\mathcal{Z}$ of size 2^n, such that each entry in this table corresponds to a simplified-signature $\vec{\tau} \in \{0,1\}^n$. Specifically, in the $\vec{\tau}'s$ entry of the table $\mathcal{Z}$, we will store the minimum number $z_{\vec{\tau}}$ of unregistered candidates, such that there is a subset $A^{(\vec{\tau})} \subseteq A \setminus \{p\}$ of exactly $z_{\vec{\tau}}$ ($z_{\vec{\tau}} \leq k$) candidates with the signature $\vec{\tau}$ (we use the value $k+1$ to indicate that there such set does not exist).

To describe our algorithm for computing the table $\mathcal{Z}$, we need the following additional piece of notation. For each pair of simplified-signatures, $\vec{\tau}$ and $\vec{\tau}'$, we define a "merged" simplified-signature $\vec{\tau} \oplus \vec{\tau}' = (\max\{\tau_i, \tau'_i\})_{i \in [n]}$. In other words, we apply the coordinate-wise max operator.

We compute the table $\mathcal{Z}$ as follows (our algorithm is slightly more complicated than necessary for the case of the Plurality rule since we will use it as a base for more involved settings as well):

1) We initiate the table by setting $z_{\vec{\tau}} := 1$ if at least one unregistered candidate with signature $\vec{\tau}$ exists, and we set $z_{\vec{\tau}} := k+1$ otherwise.

2) Iteratively, for each unregistered candidate a, we perform the following operations:
   a) We compute a's simplified-signature $\vec{\tau}_a$.
   b) We compute a new table $\mathcal{Z}'$, with the entries $z'_{\vec{\tau}}$, by setting, for each simplified-signature $\vec{\tau}$:

$$z'_{\vec{\tau}} = \min(\{z_{\vec{\tau}}\} \cup \{z_{\vec{\tau}} + 1 : \vec{\tau} = \vec{\tau}' \oplus \vec{\tau}_a\} \cup \{k+1\}).$$

   c) We copy the contents of $\mathcal{Z}'$ to $\mathcal{Z}$. (At this point, for each signature $\vec{\tau}$, we have that $z_{\vec{\tau}}$ is the number of candidates in the smallest set (of size up to k) which is composed of only the so-far processed candidates, such that, jointly, have this simplified-signature, or is $k+1$ if no such set exists.)

After we have computed the table $\mathcal{Z}$ as described above, we check whether there is at least one simplified-signature $\vec{\tau}$ for which the value $z_{\vec{\tau}}$ is at most k (recall that k is the number of candidates that we are allowed to add), and such that adding the corresponding candidate set $A_{\vec{\tau}}$ which implements this

simplified-signature $\vec{\tau}$ ensures that p has more points than d. It is easy to check whether a simplified-signature ensured that p has more points than d: given a simplified-signature $\vec{\tau}$, if the ith component τ_i is zero, then the ith voter gives one point to whomever this voter ranks first among the registered candidates; otherwise, if τ_i is one, then the ith voter gives one point to a candidate from $A_{\vec{\tau}}$, that is, neither to p nor d).

We first consider the algorithm's running time. The most time-consuming part of the algorithm is iteratively updating the contents of the table $\mathcal{Z}$. If, in each iteration, we first copy the then-current contents of $\mathcal{Z}$ to $\mathcal{Z}'$, and only then perform the remaining updates, then the time we need is $O(m \cdot n \cdot 2^n)$ (recall that there are 2^n simplified-signatures). This part dominates the running time of the remaining parts of the algorithm.

Let us consider the correctness of the algorithm. Assume that we have guessed the correct candidate p and that there is a subset of $l \leq k$ unregistered candidates $A' = \{a_1, \ldots, a_\ell\}$ such that, after we add the candidates from A', p has more points than d. If $\vec{\tau}$ is the simplified-signature of the set A', then the algorithm indeed computes a value $z_{\vec{\tau}} \leq \ell$. Further, if the algorithm returns True, then it must correspond to such set. □

We can generalize the above ideas to the cases of t-Approval and t-Veto as well. The proofs are given in the appendix to this chapter.

Theorem 3.29. *For each fixed integer $t \geq 2$, t-*APPROVAL*-DCAC can be solved in $\min\{O(m \cdot (t \cdot 3^n)^{t \cdot n}), O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})\}$ time, where m is the total number of candidates and n is the number of voters.*

We can adapt the algorithms used for Theorem 3.28 and Theorem 3.29 in a straight-forward way (basically, by reversing the signatures) to show analogous results for the case of t-Veto-DCAC.

Corollary 3.2. *For each fixed integer $t \geq 1$, t-*VETO*-DCAC can be solved in $\min\{O(m \cdot (t \cdot 3^n)^{t \cdot n}), O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})\}$ time, where m is the total number of candidates and n is the number of voters.*

To conclude the discussion of the Signature technique for the case of destructive control by adding candidates, we consider the combinatorial variant of the problem. The situation is more complicated since we should add bundles of candidates instead of individual candidates. In effect, both for t-Approval-Comb-DCAC and for t-Veto-Comb-DCAC), we cannot upper-bound the number of bundles to add. This is so, since bundles with the same signature, but with different sizes,

may have different effects on the score difference between the despised candidate d and a specific guessed candidate p (indeed, t-Approval-Comb-DCAC is W[1]-hard for $t \geq 2$, as shown in Section 3.4). Yet, for the Plurality rule (for the Veto rule), only the first (respectively, the last) position gets a point (respectively, a veto). This structural observation allows us to generalize our non-combinatorial algorithms.

Corollary 3.3. PLURALITY-COMB-DCAC *and* VETO-COMB-DCAC *are fixed-parameter tractable with respect to the number n of voters.*

Proof. For the case of Plurality, it suffices to use, for example, the same algorithm as in Theorem 3.28, but with the following changes:

1) For each choice of candidate p, we also consider each way of adding p to the election if p were unregistered (p might belong to several different bundles and we try each possibility).

2) Each unregistered candidate's signature is replaced by the signature of the set of candidates in his bundle.

Since under Plurality each voter gives a point only to whomever this voter ranks first, this strategy suffices. The case of the Veto rule is handled analogously. □

3.7.2. Destructive Control by Deleting Candidates

Let us now move on to the case of destructive control by deleting candidates. The (Turing) kernelization approach from the previous section does not easily transfer to the case of deleting candidates. The problem is that we cannot upper-bound (by a function dependent only on the number n of voters) the number of candidates that have to be deleted.

However, by applying our Signature technique, followed by casting the remaining task as an integer linear program, we can show fixed-parameter tractability. We present the proof of the following result in the appendix to this chapter (while the proof is quite interesting technically, we believe that the previous proofs have presented the Signatures technique sufficiently well).

Theorem 3.30. *For each fixed integer $t \geq 1$, both t-*APPROVAL-DCDC *and t-*VETO-DCDC *can be solved in time $O^*(m \cdot 4^n \cdot (3^n)^{3^n})$, where m is the total number of candidates and n is the number of voters.*

3.8. Outlook

Several possible research directions, motivated by the work presented in this chapter, are listed below.

- We still do not know the exact complexity of 2-Veto-Comb-DCAC. This open question is marked by a question mark (?) in Table 3.1. It looks as if none of the proof techniques developed in this chapter fits to this case, thus, it might be that a new proof technique is needed for it.

- There are several NP-hardness results in this chapter which hold for contant number of voters (that is, Para-NP-hardness results). Besides some results which hold even when there is only one voter in the election, the other reductions use 2, 3, and up to 26 voters. It is not clear where is the borderline; one might try to find the lowest possible numbers for which the respective problems are still hard, and then, hopefully, also find matching algorithms for the other lower numbers.

- It is natural to consider an even more diverse set of voting rules. This might allow for understanding our general techniques better, and might help in devising new techniques as well. For example, it would be interesting to consider the Bucklin rule: on the one hand, candidate control problems for Bucklin are NP-complete [68] and, on the other hand, this rule can be seen as an adaptive variant of t-Approval. Thus, it would be interesting to see if our techniques can be applied to the case of the Bucklin rule (see, for example, the paper by [68], for the formal definition of the Bucklin voting rule).

- One might experiment with real-world elections to understand the practical relevance of our theoretical findings or heuristically solve our proven intractable cases. One possible starting point for such an analysis would be the experimental paper of Erdélyi et al. [69].

  At least for the easier rules, such as the Plurality rule and the Veto rule, there is quite some hope that our signature-based algorithms would prove to be useful in practice. Then, one could try to extend the applicability of these algorithms to the more complex rules, namely t-Approcal and t-Veto, at least for small values of t. It would be also interesting whether these algorithms could be extended to the combinatorial case, which is shown

in this chapter to be fixed-parameter tractable for the destructive case of deleting candidates for both the Plurality rule and the Veto rule.

- It is interesting to consider some game-theoretic aspects of candidate control, where several agents perform the control actions. So far, doing this even for the simplest rules such as the Plurality rule was hampered by the fact that these control problems are NP-hard. Our (partial) tractability results might help in overcoming this obstacle in a non-trivial way.

- It might be worthwhile to study the multimode control framework of Faliszewski et al. [77] for the case of few voters. In multimode control one can perform control actions of several types at the same time (for example, one can add candidates and also delete voters). Faliszewski et al. expected that combining two types of easy control actions would lead to a possibly computationally hard multimode control problem, but they did not observe such effects among natural voting rules. One possible explanation for this fact is that they did not have enough easy control problems available to combine. We have shown that many candidate control problems become easy (FPT) when they are parametrized by the number of voters and, thus, there are more opportunities for studying multimode control problems.

- Generally, we believe that the case of few voters did not receive sufficient attention in the computational social choice literature and many other problems can (and should) be studied with respect to this parameter. The main two justifications for this kind of study are that, first, as discussed in Section 3.2.1, it is very well motivated, and, second, in our control problems we observe a rich (parameterized) complexity landscape, which we hope might be observed in other voting problems.

Appendix for Chapter 3

We provide remaining results and proofs deffered from Chapter 3.

3.A. Deffered Proofs for the Multicolored Clique Proof Technique

Theorem 3.5. *For each fixed integer t, $t \geq 2$, t-*APPROVAL-CCAC *is* W[1]*-hard, parameterized by the number n of voters.*

Proof. We use the same proof as in the case of Theorem 3.1, but for each voter we introduce $t-1$ additional registered dummy candidates which this voter ranks first (each voter ranks all the remaining dummy candidates last). In this way, each dummy candidate has exactly one point. The reasoning for the correctness proof works in the same way. □

Theorem 3.6. *For each fixed integer t, $t \geq 2$, t-*VETO-CCAC *is* W[1]*-hard, parameterized by the number n of voters.*

Proof. We use the same proof as in Theorem 3.2, but we introduce $t-1$ additional registered dummy candidates whom every voter ranks last. In this way, each dummy candidate receives exactly one veto from each voter, while p and d receive the same number of vetoes as in the election constructed in the proof for Theorem 3.2. The arguments from that proof apply here as well. □

Theorem 3.7. *For each fixed integer $t \geq 1$, t-*VETO-CCDC *is* W[1]*-hard, parameterized by the number n of voters.*

Proof. We use almost the same proof as in Theorem 3.3, but we add sufficiently many dummy (padding) candidates to ensure that we can only delete vertex and edge candidates. Let $I = (G,h)$ be an input instance of MULTICOLORED CLIQUE. Let $E' = (C',V')$ be the election created by the reduction from the proof of Theorem 3.3 on input I and set $k \coloneqq |V(G)| - h + 2|E(G)| - H$.

We modify this election by extending C' to contain a set D of t dummy candidates, $D = \{d_1, \ldots, d_t\}$, and modify the voter collection V' as follows (recall that the number $|V'|$ of voters is polynomially upper-bounded by h; set $n' := |V'|$):

1) For each voter v in V' except the last group of voters, we modify v's preference order to rank the dummies $d_1, \ldots, d_{t-1}$ last and d_t first.

2) For each voter v in the last group of V', we rank all candidates from D such that v will have preference order of the form

$$d_t \succ \cdots \succ (D \setminus \{d_t\}) \succ p.$$

3) We add n' voters, all with preference order of the form

$$\cdots \succ p \succ D.$$

One can verify that, for each $1 \leq i \leq t-1$, each newly added candidate d_i has $2n'$ vetoes and d_t has n' vetoes. Since we assume the input graph to be connected and to have at least two vertices, at least one candidate from the edge and vertex candidates receives fewer vetoes than p. Thus, p is not a winner initially.

We claim that p (the preferred candidate from the proof of Theorem 3.3) can become a winner by deleting at most k candidates if and only if I is a yes-instance.

First, note that if we delete any of the new dummy candidates from $D \setminus \{d_t\}$, then p certainly does not become a winner since p will have at least $n' + 2H$ vetoes and d_t will have exactly n' vetoes. Second, if we delete dummy candidate d_t, then p will receive $2n'$ vetoes, but there will be at least one remaining vertex or edge candidate which is not vetoed by the last group of voters and, therefore, will have less than $2n'$ vetoes. It follows that no dummy candidate can be deleted. Thus, none of the dummy candidates will have fewer vetoes than p and (ignoring the dummy candidates) the election will behave as if it was held according to the Veto rule. The remaining arguments from the proof of correctness in Theorem 3.3 hold. □

Theorem 3.8. 2-APPROVAL-CCDC*, parameterized by the number n of voters, is* W[1]*-hard.*

Proof. The proof is quite similar to that for the case of Veto-CCDC, but the construction is more involved. Again, we give a parameterized reduction from the MULTICOLORED CLIQUE problem. Let $I = (G, h)$ be our input instance with

graph G and an integer h, and let the notation be as described in the introduction to Section 3.4. Based on I, we form an instance I' of 2-Approval-CCDC, as follows.

We set $T = |V(G)| + |E(G)|$ with the intended meaning that T is an integer larger than the number of candidates that we can delete. We set $H \coloneqq 2\binom{h}{2} = (h-1) \cdot h$. We build our candidate set C as follows.

1) We introduce the preferred candidate p.
2) We introduce T candidates $b_1, \ldots, b_T$. These are the *blocker* candidates, whose role, on the one hand, is to ensure that p will have to obtain a given number of points and, on the other hand, is to prevent the possibility of deleting too many candidates of other types.
3) For each vertex $v \in V(G)$, we introduce a candidate v.
4) For each edge $\{u, v\} \in E(G)$, we introduce two candidates, (u, v), and (v, u).
5) We introduce two sets, $D = \{d_1, \ldots d_h\}$ and $F = \{f_{(i,j)} : 1 \leq i, j \leq h, i \neq j\}$, of dummy candidates.

The set of voters consists of the following groups (we write B to refer to the arbitrary ordering of $\{b_1, b_2, b_T\}$).

1) We have $h + 3H$ voters, each with preference order of the form
$$B \succ \cdots \succ p.$$
2) For each color i, $1 \leq i \leq h$, we create $3H + 1$ voters, where the first of them has preference order of the form
$$V_i(G) \succ p \succ B \succ \cdots,$$
and the remaining ones have preference order of the form
$$V_i(G) \succ d_i \succ B \succ \cdots.$$
3) For each pair i, j of distinct colors ($1 \leq i, j \leq h$, $i \neq j$), we create $3H + h - 1$ voters, where the first of them has preference order of the form
$$E(i,j) \succ p \succ B \succ \cdots,$$
and the remaining ones have preference order of the form
$$E(i,j) \succ f_{(i,j)} \succ B \succ \cdots.$$

4) For each pair i,j of distinct colors ($1 \le i,j \le h$, $i \ne j$), we introduce two voters with preference orders of the following forms

$$p \succ R(i,j) \succ B \succ \cdots,$$

$$p \succ R'(i,j) \succ B \succ \cdots.$$

Note that the total number of constructed voters is polynomially bounded by h:

$$h + 3H + (3H+1) \cdot h + (3H+h-1) \cdot H + 2H = 2h + 4H + 4H \cdot h + 3H^2.$$

We set the number of candidates that can be deleted to $k := |V(G)| - h + 2|E(G)| - H$, with the intention that p can become a winner if and only if it is possible to delete all vertex candidates and all edge candidates, except for the ones corresponding to a multicolored clique of order h.

For the "if" direction, note that if G indeed contains an multicolored clique Q of size h, then deleting all candidates in $V(G) \setminus Q$ and all edge candidates of the form (u,v), where either $u \notin Q$ or $v \notin Q$, indeed ensures that p is a winner (in this case, p, and all the vertex and edge candidates, have $h+3H$ points each, and all the blocker candidates have at most $h+3H$ points each).

For the "only if" direction, assume that it is possible to ensure p's victory by deleting at most k candidates and let $C' \subseteq C$ be a set of at most k candidates such that p is a winner of $E' = (C \setminus C', V)$. Note that $k < T - 1$ and so there are at least two blocker candidates that receive $h+3H$ points each from the first group of voters. The only voters that p can obtain points from, after deleting at most k candidates, are the ones in the second and third group and there are exactly $h+H$ of them (h in the second group and H in the third group).

However, p can obtain the points from the second and the third groups of voters without, at the same time, increasing the score of the highest-scoring blocker candidate if and only if we delete all but one vertex candidate of each color, and, for each pair i,j of distinct colors ($1 \le i,j \le h$, $i \ne j$), we delete all but one edge candidate of the form (u,v), where $u \in V_i(G)$ and $v \in V_j(G)$. Indeed, this means deleting exactly k candidates. Further, we claim that if p is a winner of E', then for each pair of vertex candidates u and v which were not deleted, it must be the case that both edge candidates (u,v) and (v,u) remain in the election, meaning that there is an edge between u and v in the original graph. It suffices to consider the case of (u,v) (the case of (v,u) is symmetric). If, instead of (u,v), the only remaining edge candidate for the pair of colors u and v is some edge candidate (u',v') (where $(u',v') \ne (u,v)$), then one of the two following cases must

happen: either u and v would receive more than $h-1$ points from the fourth group, therefore would have more than $h+2\binom{h}{2}$ points, causing p not to be a winner, or (u',v') would receive more than one point from the fourth group, again causing p to not be a winner. Thus, p can become a winner by deleting at most H candidates if and only if G contains a multicolored clique of order h.

It is clear that the given reduction can be computed in polynomial time and that it is a parameterized reduction. □

Theorem 3.9. *For each fixed integer t, $t \geq 3$, t-*APPROVAL-CCDC *is* W[1]*-hard, parameterized by the number n of voters.*

Proof. Let $E' = (C', V')$ be the election constructed in the proof for Theorem 3.8. We use the same proof as for Theorem 3.8 except that now, for each voter $v_i \in V'$, we introduce a group of $t-2$ new dummy candidates, $d_1^i, d_2^i, \ldots, d_{t-2}^i$. These dummy candidates are ranked first by their corresponding voter, and for each such introduced group, we introduce two new dummies, c_1^i and c_2^i, as well as additional $|V'|-1$ voters with preference order of the form (we write D_i to refer to the preference order $d_1^i \succ d_2^i \succ \ldots \succ d_{t-2}^i$):

$$D_i \succ c_1^i \succ c_2^i \succ B \succ \cdots.$$

These voters ensure that none of the new dummy candidates can be deleted without increasing the score of the highest-scoring blocker candidate. If a score of a highest-scoring blocker candidate increases, then the preferred candidate has no longer any chance to win the election. If none of the new dummy candidates can be deleted, then the correctness proof works the same as the one given for Theorem 3.8.

The number of voters is still polynomially bounded by the clique order h. □

3.B. Deffered Proofs for the Cubic Vertex Cover Proof Technique

Theorem 3.12. BORDA-CCAC *is* NP*-hard, even for elections with only ten voters.*

Proof. We give a reduction from the CUBIC VERTEX COVER problem (we use the notation as provided at the beginning of Section 3.5). Given an instance $I = (G, h)$ for CUBIC VERTEX COVER, we construct an instance for Borda-CCAC.

We let the registered candidate set C be $\{p, d\} \cup E(G)$, and we let $V(G)$ be the set of unregistered candidates. We construct the following voters:

1) For each ℓ, $1 \le \ell \le 3$, we have the following two voters:

$$\mu(\ell)\colon B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$

$$\mu'(\ell)\colon p \succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

2) For $\ell = 4$, we have the following two voters:

$$\mu(\ell)\colon B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$

$$\mu'(\ell)\colon d \succ p \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

3) We have two voters with preference orders

$$E(G) \succ p \succ d \succ V(G),$$

$$p \succ \overleftarrow{E(G)} \succ d \succ \overleftarrow{V(G)}.$$

We claim that it is possible to ensure p's victory by adding $k := h$ candidates if and only if there is a vertex cover of size h for G.

Let $m' := |E(G)|$ be the number of edges in $E(G)$. Note that initially, p has $5m'+5$ points, d has $4m'+5$ points, and each edge candidate has $5m'+6$ points. Thus, p is not a winner. Adding each unregistered vertex candidate v causes the scores of all candidates to increase: for the edge candidates that include v as an endpoint this increase is by five points, whereas for all other candidates this increase is by six points. Note that the last two voters always prefer the registered candidates to any vertex candidate. Thus, by simple counting, each of these h vertex candidates may obtain at most $4m'+5h+7$ points and will never obtain more points than p as long as $m'+h \ge 2$.

Thus, if we have a vertex cover of size h, then it is possible to ensure p's victory by adding all vertex candidates that correspond to this vertex cover. For the other direction, assume that it is possible to ensure p's victory by adding at most h candidates and let S be such a set of candidates.

Now, if there would be an edge candidate e which is not covered by some vertex candidate in S, then it would follow that the score of e is greater than the score of p, Thus, S must correspond to a vertex cover in G. □

Theorem 3.13. *For each rational number α, $0 \le \alpha \le 1$,* COPELAND$^\alpha$-CCAC *is* NP*-hard, even for elections with only twenty voters.*

Proof. We give a reduction from the CUBIC VERTEX COVER problem (we use the notation as provided at the beginning of Section 3.5). Given an instance $I = (G, h)$ of CUBIC VERTEX COVER, we construct an instance for Copeland$^\alpha$-CCAC. Let the registered candidate set C be $\{p, d\} \cup E(G)$ and let $V(G)$ be the set of unregistered candidates. We introduce the following voters:

1) For each ℓ, $1 \le \ell \le 4$, we construct four voters, two voters with the following preference order:
$$B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$
and two voters with the following preference order:
$$p \succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

2) One voter with the preference order $E \succ V \succ d \succ p$, and one voter with the preference order $d \succ p \succ \overleftarrow{E} \succ \overleftarrow{V}$.

3) One voter with the preference order $p \succ V \succ E \succ d$, and one voter with the preference order $\overleftarrow{E} \succ d \succ p \succ \overleftarrow{V}$.

We illustrate the results of head-to-head contests between the candidates in Figure 3.8. We claim that there is a vertex cover of size at most h for G if and only if p can become a winner of the election by adding at most $k := h$ candidates.

During the proof, when we say that a vertex candidate and an edge candidate are *adjacent* to each other, we mean that the corresponding vertex and edge are adjacent to each other. Consider a situation where we have added some subset A' of k candidates ($k \le h$; take $k = 0$ to see the situation prior to adding any of the unregistered candidates). The candidates have the following scores:

1) p has score $\alpha m' + k$ (p ties head-to-head contests with all edge candidates and wins all head-to-head contests with the vertex candidates).

2) d has score $1 + \alpha k$ (d wins the head-to-head contest with p and ties all head-to-head contests with the vertex candidates).

3) Each added vertex candidate v has score $3 + \alpha k$ (v ties the head-to-head contests with d and the remaining $k - 1$ vertex candidates and wins the head-to-head contests with the three edge candidates that are adjacent to v).

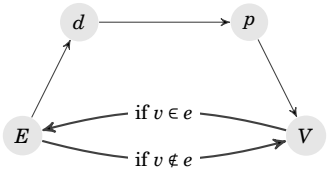


Figure 3.8. Illustration for the reduction used in the proof of Theorem 3.13. Each vertex in the graph corresponds to a candidate or a set of candidates, and there is an arc going from a vertex u_1 to a vertex u_2 if u_1 beats u_2 in a head-to-head contest. Edges indicating ties are omitted. The main idea is that an edge candidate beats a vertex candidate if and only if the vertex candidate is one of the endpoints of the edge candidate.

4) Each edge candidate e has score $\alpha m' + k + 1 - c(e)$, where $c(e)$ is the number of vertex candidates from A' that are adjacent to e (e ties head-to-head contests with p and the remaining edge candidates and wins head-to-head contests with d and all added vertex candidates except those that are adjacent to e).

In effect, it is easy to see that p is a winner of the election if and only if A' corresponds to a vertex cover of G. □

Theorem 3.14. *For each rational number α, $0 \le \alpha \le 1$,* Copeland$^\alpha$-CCDC *is* NP*-hard, even for elections with only twenty-six voters.*

Proof. We give a reduction from Cubic Vertex Cover (we use the notation as provided at the beginning of Section 3.5). Given an instance $I = (G, h)$ for Cubic Vertex Cover, we construct an instance for Copeland$^\alpha$-CCDC. The candidate set contains the edge candidates, the vertex candidates, the preferred candidate p, a dummy candidate d, and a set of additional dummy candidates $Z = \{z_1, \ldots, z_{m'+n'}\}$ (recall that $m' := |E(G)|$ denotes the number of edges in $E(G)$ and that $n' := |V(G)|$ denotes the number of vertices in $V(G)$). We construct the following voters:

1) For each ℓ, $1 \le \ell \le 4$, we construct two voters with preference order:

$$A(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ Z \succ d \succ p,$$

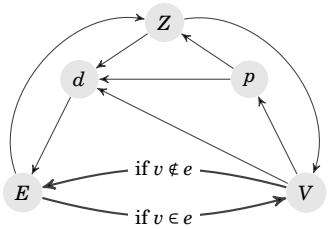


Figure 3.9. Illustration for the reduction used in the proof of Theorem 3.14. Each vertex in the graph corresponds to a candidate or a set of candidates, and there is an arc going from a vertex u_1 to a vertex u_2 if u_1 beats u_2 in a head-to-head contest. Edges indicating ties are omitted. The main idea is that an edge candidate beats a vertex candidate if and only if the vertex candidate is one of the endpoints of the edge candidate.

and two voters with preference order:

$$p \succ d \succ \overleftarrow{Z} \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ A'(\ell).$$

2) We also construct the following ten voters:

$$v_1 \colon V \succ E \succ Z \succ d \succ p,$$

$$v_1' \colon p \succ d \succ \overleftarrow{Z} \succ \overleftarrow{V} \succ \overleftarrow{E},$$

$$v_2 \colon V \succ p \succ d \succ E \succ Z,$$

$$v_2' \colon \overleftarrow{E} \succ \overleftarrow{Z} \succ \overleftarrow{V} \succ p \succ d,$$

$$v_3 \colon p \succ Z \succ d \succ V \succ E,$$

$$v_3' \colon \overleftarrow{E} \succ \overleftarrow{V} \succ p \succ \overleftarrow{Z} \succ d,$$

$$v_4 \colon d \succ E \succ Z \succ V \succ p,$$

$$v_4' \colon p \succ \overleftarrow{V} \succ \overleftarrow{Z} \succ d \succ \overleftarrow{E},$$

$$v_5 \colon Z \succ V \succ E \succ d \succ p,$$

$$v_5' \colon p \succ d \succ \overleftarrow{E} \succ \overleftarrow{Z} \succ \overleftarrow{V}.$$

Figure 3.9 illustrates the results of the head-to-head contests among the candidates. Prior to deleting any of the candidates, we have the following scores:

1) each edge candidate e has $m' + n' + \alpha m' + 2$ points (e wins head-to-head contests against all candidates in Z due to voters v_2 and v_2', wins head-to-head contests against its "incident" vertex candidates due to the first group of voters, and ties with p and the remaining edge candidates),

2) each vertex candidate u has $\alpha(n'-1) + m' - 1$ points (u wins head-to-head contests against all edge candidates that are *not* "incident" to u due to voters from the first group, and ties with the remaining vertex candidates),

3) each candidate z from Z has $n'+1+\alpha(m'+n'-1)$ points (z wins head-to-head contests against all vertex candidates and d due to voters v_3, v_3', v_5, v_5', and ties with the remaining candidates from Z),

4) d has m' points (d wins head-to-head contests against all edge candidates due to voters v_4 and v_4'), and

5) p has $m' + n' + \alpha m' + 1$ points (p wins head-to-head contests against all candidates from Z due to voters v_3 and v_3', wins head-to-head contests against d due to voters v_2, v_2', v_3, v_3', and ties with all edge candidates).

Thus, all edge candidates are co-winners, and p is not a winner since each edge candidate has one point more than p. However, p has more points than any other non-edge candidate. Note that in the input graph it holds that $m' = 3n'/2$.

We claim that it is possible to ensure that p is a winner by deleting at most $k := h$ candidates if and only if there is a vertex cover of size h for G.

If there is a vertex cover for G of size h, then deleting the corresponding h vertices ensures that p is a winner. To see why this is the case, note that after deleting vertices corresponding to a vertex cover the score of p does not change, but the score of each edge candidate decreases by at least one. The scores of other candidates cannot increase, so p is a winner.

For the reverse direction, assume that it is possible to ensure that p is a winner by deleting at most h candidates. Deleting candidates cannot increase p's score, so it must be the case that each edge candidate loses at least one point.

Observe that deleting candidates other than the vertex candidates will not make the edge candidates lose more than one point than p. The only possibility of deleting a candidate such that an edge candidate e loses a point but p does not is by deleting one of the vertex candidates $v'(e)$ or $v''(e)$.

Thus, if it is possible to ensure that p is a winner, then we must delete vertices that correspond to a vertex cover. □

3.C. Deffered Proofs for the Set-Embedding Proof Technique

Theorem 3.16. *For each fixed integer $t \geq 2$, t-*Approval-Comb-CCDC *is* NP*-hard, even for elections with only a single voter.*

Proof. We build upon the proof of Theorem 3.15, but add $t-1$ dummy candidates. Specifically, given an instance $I := (X, \mathscr{S}, h)$ for Set Cover, we create an instance I' of t-Approval-Comb-CCDC as follows. We construct an election $E = (C, V)$ where $C = \{p\} \cup X_{\text{cand}} \cup \mathscr{S}_{\text{cand}} \cup D$, where $D = \{d_1, \ldots, d_{t-1}\}$, and where V contains a single voter with the following preference order:

$$D \succ X_{\text{cand}} \succ p \succ \mathscr{S}_{\text{cand}}.$$

We use the bundling function as described in the introduction to Section 3.6. We claim that I is a yes-instance of Set Cover if and only if it is possible to ensure p's victory by deleting at most h (bundles of) candidates.

To see the correctness of the argument, note that if there is a solution that ensures p is a winner by deleting a specific number of candidates, then there is also a solution that achieves the same and does not delete any of the dummy

candidates (it is always at least as useful to delete one of the set candidates instead of a dummy one). □

Theorem 3.17. *For each fixed integer $t \geq 1$, t-*VETO-COMB-CCDC *is* NP*-hard, even for elections with only a single voter.*

Proof. Let the notation be as in the introduction to Section 3.6. Given an instance $I := (X, \mathscr{S}, h)$ for SET COVER, we create an instance I' of t-Veto-COMB-CCDC as follows. We construct an election $E = (C, V)$ with candidate set:

$$C = \{p, z\} \cup X_{\text{cand}} \cup \mathscr{S}_{\text{cand}} \cup D,$$

where $D = \{d_1, \ldots, d_{t-1}\}$ is a set of dummy candidates (indeed, for $t = 1$, that is, for Veto, $D = \emptyset$), and with the voter collection V containing a single voter with the following preference order:

$$z \succ X_{\text{cand}} \succ \mathscr{S}_{\text{cand}} \succ D \succ p.$$

We use the set-embedding bundling function, with the added feature that $\kappa(z) = \mathscr{S}_{\text{cand}}$. We claim that I is a "yes"-instance of SET COVER if and only if it is possible to ensure p's victory by deleting at most $h + 1$ bundles.

Using similar reasoning as used in Theorem 3.16, it is easy to see that the only way of ensuring that p is a winner is to let all the remaining candidates receive no points at all. The only way to achieve this is to first delete up to h candidates from $\{s_1, \ldots, s_m\}$ that correspond to a cover of the ground set and then to delete z. □

Theorem 3.19. PLURALITY-COMB-DCDC *is* NP*-hard, even for elections with only three voters.*

Proof. Let the notation be as in the introduction to Section 3.6. Given an instance $I := (X, \mathscr{S}, h)$ for SET COVER, we create an instance I' of Plurality-COMB-DCDC as follows. We construct an election $E = (C, V)$ where $C = \{p, d\} \cup X_{\text{cand}} \cup \mathscr{S}_{\text{cand}}$, and where V contains three voters with the following preference orders:

$$X_{\text{cand}} \succ p \succ \mathscr{S}_{\text{cand}} \succ d,$$

$$d \succ X_{\text{cand}} \succ p \succ \mathscr{S}_{\text{cand}},$$

$$p \succ d \succ X_{\text{cand}} \succ \mathscr{S}_{\text{cand}}.$$

We use the set-embedding bundling function. We claim that the despised candidate d can be precluded from winning by deleting at most h (bundles of) candidates if and only if there is a set cover of size h for I.

Prior to deleting any of the candidates, d, p, and one of the candidates from X are tied as winners. Since deleting candidates cannot make any candidate lose points and since deleting p will make d a unique winner, the only way of defeating d is by ensuring that the first voter gives her point to p. This means that all element candidates have to be removed from the election. By the same argument as in the previous proofs, doing so by deleting at most h candidates is possible if and only if I is a yes-instance of SET COVER. □

Theorem 3.20. *For each fixed integer $t \geq 2$, t-*APPROVAL-COMB-DCDC *is* NP*-hard, even for elections with only two voters.*

Proof. Let the notation be as in the introduction to Section 3.6. Given an instance $I := (X, \mathscr{S}, h)$ for SET COVER, we create an instance I' of t-Approval-COMB-DCDC as follows. We construct an election $E = (C, V)$ with candidate set:

$$C = \{p, d\} \cup X_{\text{cand}} \cup \mathscr{S}_{\text{cand}} \cup D \cup F,$$

where $D = \{d_1, \ldots, d_{t-2}\}$ and $F = \{f_1, \ldots, f_{t-1}\}$ are two sets of dummy candidates (note that D can be empty), and with the voter collection V containing two voters with the following preference orders:

$$d \succ X_{\text{cand}} \succ D \succ p \succ \mathscr{S}_{\text{cand}} \succ F \text{ and}$$

$$p \succ F \succ d \succ X_{\text{cand}} \succ \mathscr{S}_{\text{cand}} \succ D.$$

We use the set-embedding bundling function. We claim that I is a yes-instance of SET COVER if and only if it is possible to preclude d from winning by deleting at most h (bundles of) candidates.

Initially, both d and p are winners (as well as some members of $X_{\text{cand}} \cup F$). Deleting p will make d gain one more point (from the second voter), making it impossible for d to lose. The same holds for the dummy candidates from set F. In other words, if we change the set of candidates that gain a point from the second voter, then d will obtain two points and will certainly be a winner. This implies that the only way of making d lose is to let either p or at least one candidate from F gain one point from the first voter. By construction of the first voter's preference order, this is possible only for p if and only if we delete all members of X_{cand}.

As in the previous proofs, deleting them (through deleting at most h bundles of candidates) is possible if and only if I is a yes-instance of SET COVER. □

Theorem 3.21. *For each fixed integer $t \geq 1$, t-*VETO-COMB-DCDC *is* NP*-hard, even for elections with only a single voter.*

Proof. We use the same construction as used in Theorem 3.16 for t-Approval-COMB-CCDC but we reverse the preference order and replace p with d, the despised candidate:

$$\mathscr{S}_{\text{cand}} \succ d \succ X_{\text{cand}} \succ D.$$

The crucial observation here is that with only one voter, the only way of preventing d from winning is to rank her within the last t positions. This means that all element candidates have to "disappear" from the election (one could also try deleting the dummy candidates, but it is never a mistake to "make disappear" the members of X_{cand} instead, through deleting the appropriate candidates in $\mathscr{S}_{\text{cand}}$). Thus we can conclude that the set of deleted candidates contains the set candidates only. Clearly, if d is to be precluded from winning by deleting at most h candidates, this set must correspond to a set cover of size h. Since we can assume that $h < |\mathscr{S}_{\text{cand}}|$, there is at least one set element not deleted, and this will be a winner. □

Theorem 3.22. BORDA-COMB-DCDC *is* NP*-hard, even for elections with only two voters.*

Proof. Let the notation be as in the introduction to Section 3.6. Given an instance $I := (X, \mathscr{S}, h)$ for SET COVER, we create an instance I' of Borda-COMB-DCDC as follows. We construct an election $E = (C, V)$ where $C = \{p, d, z\} \cup X_{\text{cand}} \cup \mathscr{S}_{\text{cand}}$ and where V contains two voters with the following preference orders:

$$d \succ X_{\text{cand}} \succ p \succ \mathscr{S}_{\text{cand}} \succ z \text{ and}$$

$$p \succ z \succ d \succ \overleftarrow{X_{\text{cand}}} \succ \overleftarrow{\mathscr{S}_{\text{cand}}}.$$

We use the set-embedding bundling function. We claim that I is a yes-instance of SET COVER if and only if it is possible to preclude d from winning by deleting at most h (bundles of) candidates.

For convenience, we calculate the scores of all candidates:

1) d has $2|\mathscr{S}_{\text{cand}}| + 2|X_{\text{cand}}| + 2$ points.

2) p has $2|\mathscr{S}_{\text{cand}}| + |X_{\text{cand}}| + 3$ points.

3) Each element candidate x_i has $2|\mathscr{S}_{\text{cand}}| + |X_{\text{cand}}| + 1$ points.

4) z has $|\mathscr{S}_{\text{cand}}| + |X_{\text{cand}}| + 1$ points.

5) Each set candidate s_j has $|\mathscr{S}_{\text{cand}}|$ points.

Clearly, d has the highest number of points and, thus, is a winner.

Since both voters rank d ahead of the candidates in the set $X_{\text{cand}} \cup \mathscr{S}_{\text{cand}}$, no member of this set can have the score higher than d, irrespective of which other candidates we delete. Similarly, irrespective of which candidates we delete, z will never have score higher than d. We conclude that candidate p is the only candidate that has a chance of defeating d.

Since deleting candidates does not increase the scores of any of the remaining candidates, to ensure that d is not a winner, we have to guarantee that he loses at least $|X_{\text{cand}}|$ points (relative to p). This means that it is possible to ensure that d is not a winner if and only if it is possible to remove all candidates from X_{cand}. However, this is possible if and only if I is a yes-instance of SET COVER. □

Theorem 3.23. COPELAND$^\alpha$-COMB-DCDC *is* NP*-hard, even for elections with only three voters.*

Proof. Let the notation be as in the introduction to Section 3.6. Given an instance $I := (X, \mathscr{S}, h)$ for SET COVER, we construct an instance for Copeland$^\alpha$-COMB-DCDC. Since our reduction will produce an instance with an odd number of voters, the particular value of α is immaterial. We form the set of candidates:

$$C = \{d, p\} \cup X_{\text{cand}} \cup \mathscr{S}_{\text{cand}}.$$

We have three voters with the following preference orders:

$$p \succ d \succ X_{\text{cand}} \succ \mathscr{S}_{\text{cand}},$$

$$d \succ X_{\text{cand}} \succ p \succ \mathscr{S}_{\text{cand}},$$

$$\overleftarrow{X_{\text{cand}}} \succ p \succ d \succ \overleftarrow{\mathscr{S}_{\text{cand}}}.$$

We use the set-embedding bundling function. We claim that I is a yes-instance of SET COVER if and only if it is possible to preclude d's victory by deleting at most h (bundles of) candidates.

The initial scores are:

1) d receives $|\mathscr{S}_{\text{cand}}| + |X_{\text{cand}}|$ points (d wins head-to-head contests against all other candidates but p).

2) p receives $|\mathscr{S}_{\text{cand}}| + 1$ point (p wins head-to-head contests against d and all the members of $\mathscr{S}_{\text{cand}}$).

3) Each member x_i of X_{cand} receives at most $|\mathscr{S}_{\text{cand}}| + |X_{\text{cand}}|$ (from head-to-head contests with p, all members of $\mathscr{S}_{\text{cand}}$, and the other members of X_{cand}).

4) Each member s_j of $\mathscr{S}_{\text{cand}}$ receives at most $|\mathscr{S}_{\text{cand}}| - 1$ points (from head-to-head contests with the other members of $\mathscr{S}_{\text{cand}}$).

Since deleting candidates cannot make any candidate gain more points, the only way of ensuring that d is not a winner is to make sure that d's score decreases relative to some other candidate. By the above list of scores, it is easy to see that the only candidate that may end up with a score higher than d is p. This happens only if we remove all the members of X_{cand}. As in the previous proofs using the set-embedding technique, doing so by deleting at most h candidates is possible if and only if there is a set cover of size at most h for I. □

Theorem 3.25. *For each rational number α, $1 \leq \alpha \leq 1$,* COPELAND$^\alpha$-COMB-DCAC *and* COPELAND$^\alpha$-COMB-CCAC *are* NP*-hard, even for elections with only three voters.*

Proof. Let the notation be as in the introduction to Section 3.6. Given an instance $I \coloneqq (X, \mathscr{S}, h)$ for SET COVER with $n' \coloneqq |X_{\text{cand}}|$, we construct an instance for Copeland$^\alpha$-COMB-DCAC. Since our reduction will produce an instance with an odd number of voters, the particular value of α is immaterial. We form the set of registered candidates:

$$C = \{d, p\} \cup D \cup F,$$

where d is the despised candidate (and we will want to ensure that p wins over d), and where $D \coloneqq \{d_1, \ldots, d_{n-2}\}$ and $F \coloneqq \{f_1, \ldots, f_{n-1}\}$ are two sets of dummy

candidates. We let the set of unregistered candidates be $A = X_{\text{cand}} \cup \mathscr{S}_{\text{cand}}$. Finally, we construct three voters with the following preference orders:

$$d \succ D \succ p \succ F \succ X_{\text{cand}} \succ \mathscr{S}_{\text{cand}},$$

$$p \succ \overleftarrow{F} \succ \overleftarrow{X_{\text{cand}}} \succ \overleftarrow{D} \succ d \succ \overleftarrow{\mathscr{S}_{\text{cand}}},$$

$$X_{\text{cand}} \succ d \succ D \succ F \succ p \succ \mathscr{S}_{\text{cand}}.$$

We use the set-embedding bundling function. We claim that I is a yes-instance of SET COVER if and only if it is possible to preclude d's victory by adding at most h (bundles of) candidates.

Prior to adding any of the candidates, we have the following scores:

1) d receives $2n' - 2$ points (d wins head-to-head contests with all the remaining registered candidates).

2) p receives $n' - 1$ points (p wins head-to-head contests with the members of F).

3) Every dummy candidate $d_i \in D$ receives at most $2n' - 3$ points (d_i wins head-to-head contests with all the members of F, with p, and—at most—all remaining members of D).

4) Every dummy candidate $f_i \in F$ receives at most $n' - 2$ points (f_i wins head-to-head contests with—at most—the remaining members of F).

It is easy to verify through simple calculation that if there is a set cover for I of size at most h, then adding the members of $\mathscr{S}_{\text{cand}}$ that correspond to the cover ensures that d is not a winner (relative to d, p gets additional n' points).

For the reverse direction, note that adding candidates to the election cannot decrease the score of any existing candidate. Thus, in order to beat d, we must add candidates to increase (relative to d) the score of some candidate. We make several observations:

1) The candidates in $\mathscr{S}_{\text{cand}}$ themselves do not contribute to the increase of a score of any candidate relative to p since all other candidates (including d) win head-to-head contests against them.

2) The scores of the members of D do not change relative to the score of d, irrespective of which other candidates join the election.

3) By the first observation in this enumeration, the maximum possible increase of a score of candidate is by n' points (if this candidate defeats all members of X_{cand} and members of X_{cand} join the election). Since all members of set F have score at most $n'-2$, none of them can obtain score higher than d, irrespective of which candidates we add.

As a conclusion, we have that the only candidate that can possibly defeat d is p, and this happens only if all members of X_{cand} join the election. It is possible to ensure that this happens by adding at most h bundles of candidates if and only if there is a set cover for I of size at most h.

We use the same construction for the case of Copeland$^\alpha$-CCAC, except that now p is the preferred candidate and we increase the size of D by one. □

Theorem 3.26. MAXIMIN-COMB-CCAC *is* NP*-hard, even for elections with only six voters.*

Proof. Let the notation be as in the introduction to Section 3.6. Given an instance $I \coloneqq (X, \mathscr{S}, h)$ of SET COVER with $n' \coloneqq |X_{\text{cand}}|$, we construct an instance for Maximin-COMB-CCAC. We let the set of registered candidates be $C \coloneqq \{p\} \cup D$, where p is the preferred candidate and where $D \coloneqq \{d_1, \dots, d_{n'}\}$ is a set of dummy candidates. The unregistered candidate set is $A \coloneqq X_{\text{cand}} \cup \mathscr{S}_{\text{cand}}$. We construct six voters with the following preference orders:

$$v_1 : p \succ x_1 \succ d_1 \succ \cdots \succ x_{n'} \succ d_{n'} \succ \mathscr{S}_{\text{cand}},$$

$$v_2 : p \succ x_{n'} \succ d_{n'} \succ \cdots \succ x_1 \succ d_1 \succ \mathscr{S}_{\text{cand}},$$

$$v_3 : x_1 \succ \cdots \succ x_{n'} \succ d_1 \succ \cdots \succ d_{n'} \succ p \succ \mathscr{S}_{\text{cand}},$$

$$v_4 : d_{n'} \succ \cdots \succ d_1 \succ p \succ x_{n'} \succ \cdots x_1 \succ \mathscr{S}_{\text{cand}},$$

$$v_5 : x_1 \succ \cdots \succ x_{n'} \succ d_1 \succ \cdots \succ d_{n'} \succ p \succ \mathscr{S}_{\text{cand}},$$

$$v_6 : d_{n'} \succ \cdots \succ d_1 \succ p \succ x_{n'} \succ \cdots \succ x_1 \succ \mathscr{S}_{\text{cand}}.$$

(Note that v_3 and v_5 have the same preference order and that v_4 and v_6 have the same preference order.) We use the set-embedding bundling function. We claim that I is a yes-instance of SET COVER if and only if it is possible to ensure p's victory by adding at most h (bundles of) candidates.

Prior to adding any of the candidates, p has two points and each candidate in D has three points. All the voters rank the members of $\mathscr{S}_{\text{cand}}$ last, so the presence of these candidates in the election does not change the scores of p and members of D. More so, members of $\mathscr{S}_{\text{cand}}$ themselves receive zero points each. If some candidate x_i appears in the election, however, then we have the following effects:

1) This candidate's score is at most two (since only voters v_3 and v_5 prefer x_i to p).

2) The score of d_i becomes at most two (since only voters v_4 and v_6 prefer d_i to x_i).

3) The score of p does not change (since already v_1 and v_2 prefer p to x_i).

This means that if there is a set cover of size at most h for I, then adding the set candidates that correspond to this cover will bring all members of X_{cand} to the election and p will be among the winners.

First, note that it is impossible to increase the score of p by adding candidates, and that for each d_i, the only way to decrease his score to at most two is to bring x_i into the election.

For the reverse direction, note that in order to let p win, we must add candidates to the election to decrease the score of every element candidate x_i, and the only way of achieving this by adding at most k bundles is by adding the s_j corresponding to the set cover. This means that if it is possible to ensure p's victory by adding at most h candidates, then it must be possible to add all members of X_{cand} into the election, and this means that there is a set cover of size at most h. □

3.D. Deffered Proofs for the Signature Proof Technique

Theorem 3.29. *For each fixed integer $t \geq 2$, t-*APPROVAL-DCAC *can be solved in $\min\{O(m \cdot (t \cdot 3^n)^{t \cdot n}), O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})\}$ time, where m is the total number of candidates and n is the number of voters.*

Proof. We have two ways to solve this problem. We can either run the brute-force algorithm on top of Theorem 3.27, obtaining running time $O(m \cdot (t \cdot 3^n)^{t \cdot n}))$, or we can use a variant of the algorithm from Theorem 3.28. Below we describe how

to adapt the algorithm from Theorem 3.28, as it allows us to achieve a better running time.

We use the same algorithm as in Theorem 3.28, but with a more involved notion of a signature and with a more involved merging operator $\oplus$. Indeed, the algorithm to be described next is a generalization of the algorithm described in Theorem 3.28.

Since we have n voters, we define the *unbounded-signature* of a set A' of unregistered candidates to be a vector $\vec{\tau}$ with n entries, such that the ith entry is a vector τ_i with t values, defined as follows. The jth entry of τ_i, for $1 \leq j \leq t$ (and $1 \leq i \leq n$), contains the number of candidates in A' that the ith voter prefers to all but $j-1$ registered candidates. Based on the definition of an unbounded-signature, we define the *bounded-signature* (in short, a signature) of a set A' as its unbounded signature where all entries greater than t are replaced by t. Altogether, there are $(t+1)^{t \cdot n}$ signatures.

Given two signatures, $\vec{\tau}'$ and $\vec{\tau}''$, we define their merge, $\vec{\tau} = \vec{\tau}' \oplus \vec{\tau}''$, as follows. For each i, $1 \leq i \leq n$, vector τ_i is computed by first calculating the component-wise sum of vectors $\vec{\tau}'$ and $\vec{\tau}''$, and then replacing with t each entry greater than t. It is easy to see that, if A' and A'' are two disjoint sets of candidates with signatures $\vec{\tau}_{A'}$ and $\vec{\tau}_{A''}$, then $\vec{\tau}_{A'} \oplus \vec{\tau}_{A''}$ is a signature of their union. (Note that in our algorithm we apply the operator $\oplus$ to "signatures of disjoint sets of candidates" only.)

It is straight-forward to verify that, given a signature of a subset A' of unregistered candidates, we can compute the scores of candidates p and d. This suffices to describe our algorithm and to justify its correctness.

The running time is $O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})$ (it is calculated in the same way as in the proof of Theorem 3.28, except that now we have more signatures and the components of the signatures are t-dimensional vectors). □

An example for the algorithm described above is provided next.

Example 4. Consider the following election.

$$v_1 : \mathbf{d} \succ c \succ a \succ \mathbf{e} \succ b \succ \mathbf{p}$$

$$v_2 : b \succ c \succ \mathbf{p} \succ \mathbf{d} \succ \mathbf{e} \succ a$$

$$v_3 : a \succ c \succ \mathbf{d} \succ \mathbf{p} \succ b \succ \mathbf{e}$$

The registered candidates are $\{d, p, e\}$ and the unregistered candidates are $\{a, b, c\}$. We consider 2-Approval (that is, $t = 2$), therefore d gets 3 points, p gets 2 points, and e gets 1 point.

We have the following signatures.

$$a\text{'s signature: } \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \quad b\text{'s signature: } \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \quad c\text{'s signature: } \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}$$

Combining a and c together, we have the following signature.

$$(a\text{'s signature }) \oplus (c\text{'s signature:}) \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 0 \end{pmatrix}$$

Indeed, it can be computed from the above signature, that $\{a, c\}$ is a solution, causing p to win the election. △

Theorem 3.30. *For each fixed integer $t \geq 1$, both t-*APPROVAL*-DCDC and t-*VETO*-DCDC can be solved in time $O^*(m \cdot 4^n \cdot (3^n)^{3^n})$, where m is the total number of candidates and n is the number of voters.*

Proof. We begin with the case of the t-Approval rule and later explain how to adapt the proof to apply to the case of the t-Veto. Let us fix a positive integer t and let $((C, V), d, k)$ be an instance of t-Approval-DCDC, where $V = (v_1, \ldots, v_n)$.

We guess a candidate p, whose role is to defeat the despised candidate d. For each such candidate p we do the as follows. First, for each voter, we "guess" one of at most four possible possibilities for the score that d and p would gain from this voter, in the resulting election (after we delete the candidates):

1) First possibility: only d receives one point.

2) Second possibility: only p receives one point.

3) Third possibility: both candidates receive one point.

4) Fourth possibility: neither p nor d receive a point.

We record our guesses in the vector $\vec{\delta}$. Note that we do not consider some of the possibilities: for example, the second possibility is not possible for a voter which prefers d to p. For each guessed p and $\vec{\delta}$, we check whether giving the points according to our guesses in $\vec{\delta}$ guarantees that p has more points than d. If so, then we run an integer linear program to find out whether it is possible to ensure that every voter gives points to candidates p and d as described by the vector $\vec{\delta}$, while respecting the budget.

To this end, for each signature $\vec{\gamma} \in [3]^n$, we create an integer variable $x_{\vec{\gamma}}$, representing the number of candidates of signature $\vec{\gamma}$ that we delete. For each signature $\vec{\gamma}$, we add the following constraint:

$$x_{\vec{\gamma}} \leq z_{\vec{\gamma}},$$

where $z_{\vec{\gamma}}$ denotes the current number of candidates with signature $\vec{\gamma}$. We add a budget constraint:

$$\sum_{l \in [3]^n} x_l \leq k.$$

We now treat each voter v_i individually, as follows. We have several cases to consider, based on which possibility we guessed for v_i.

We begin with the first and second possibility. That is, if, we guessed the first or second possibility for v_i (that is, if v_i prefers d to p and we also guessed that only d should gain a point from v_i, or, if v_i prefers p to d and we also guessed that only p should gain a point from v_i), then we add two constraints. The first constraint is as follows:

$$\sum_{\vec{\gamma}:\gamma_i=3} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \leq t-1.$$

To understand this constraint, recall the definition of a $\{d,p\}$-signature; specifically, a candidate will have $\gamma_i = 3$ if v_i prefers c to both p and d. This constraint is added since, if we would add more than $t-1$ candidates which are preferred to both p and d by v_i, then neither p or d would gain a point from v_i. The second constraint is as follows:

$$\sum_{\vec{\gamma}:\gamma_i \in \{2,3\}} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \geq t-1.$$

To understand this constraint, recall the definition of a $\{d,p\}$-signature again; specifically, a candidate will have $\gamma_i = 2$ if v_i prefers one of $\{d,p\}$ to c while c is preferred to the other candidate from $\{d,p\}$. This constraint is added since, if we would add less than $t-1$ candidates with this behavior, then p and d would both gain a point from v_i.

We continue with the third possibility. That is, if we guessed the third possibility for v_i (that is, if both d and p should gain a point from v_i), then we add the following constraint:

$$\sum_{\vec{\gamma}:\gamma_i \in \{2,3\}} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) + 2 \leq t.$$

Now, this constraint is added since, if we would add more than t candidates such that, for each candidate c out of these t candidates, v_i prefers c to at least one of d or p, then at least one of d or p would not gain a point from v_i.

We continue with the forth possibility. That is, if we guessed the fourth possibility for v_i (that is, if both d and p should not gain a point from v_i), then we add the following constraint:

$$\sum_{\vec{\gamma}:\gamma_i = 3} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \geq t.$$

This finishes the description of the ILP and the description of the algorithm.

The running time is easy to verify: since we run an ILP for each choice of a defeater candidate p and a possible way of giving p and d points, the running time is $O(m \cdot 4^n)$ multiplied by the cost of running the ILP. The running time then follows by applying the result by Lenstra [104], and since that the number of variables in the ILP is 3^n variables.

For the case of the t-Veto rule, we use the same approach as described above for the t-Approval rule, but we first reverse all preference orders and consider a candidate as a winner if his score is the lowest (in essence, this is equivalent to replacing "points" with "vetoes" in the above reasoning). □

3.E. Remaining Results

In this section we present some remaining algorithms, showing membership in XP, FPT, and P, for some cases. For XP and FPT membership, our algorithms use a simple brute-force approach.

3.E.1. Fixed-Parameter Tractability Results

We now show simple, fast FPT algorithms for Plurality-CCDC, Plurality-DCDC, and Veto-DCDC. The main idea for the algorithms in this section is to guess a subset of voters that will give a specific candidate one point under the relevant voting rule. The key observation is that, in the case of deleting candidates, after

guessing this subset of voters, it is trivial to find the set of candidates to delete to realize this guess.

Theorem 3.31. PLURALITY-CCDC *can be solved in* $O(m \cdot n \cdot 2^n)$ *time, where* m *is the total number of candidates and* n *is the number of voters.*

Proof. Let $I \coloneqq ((C,V),p,k)$ be a Plurality-CCDC instance. If I is a yes-instance, then after deleting at most k candidates, there must be a subset of voters, each giving one point to candidate p, while no other candidate has more points than p. Observe that, in order to let p gain one point from a voter, one has to delete all the candidates this voter prefers to p. Our algorithm, based on these observations, proceeds as follows.

We consider all 2^n subsets of n voters. For each considered set V' we do the following. For each voter $v' \in V'$, we delete all candidates that v' prefers to p. As a result, all members of V' rank p first. Then, we keep deleting all candidates that have more than $|V'|$ points (note that deleting some candidate that has more than $|V'|$ points may result in some other candidate exceeding this bound).

If, at the end of this process, no candidate has more than $|V'|$ points and we deleted at most k candidates, then we return True. Otherwise, we proceed to the next subset of voters. If we did not return True for any of the subsets of voters, then we return False. □

It is straight-forward to see how to adapt the algorithm from Theorem 3.31 to the destructive case. In essence, it suffices to try all choices of a candidate p whose goal is to defeat the despised candidate d and, for each such choice, guess a subset of voters that are to give points to p. If after deleting the candidates that these voters prefer to p (assuming that none of them prefers d to p) the despised candidate d has fewer points than p, then we return True. In the destructive case there is no need to have the additional loop of deleting candidates with higher score than p.

Corollary 3.4. *Plurality-DCDC can be solved in* $O(m^2 \cdot n \cdot 2^n)$ *time, where* n *is the number of voters and* m *is the number of candidates in the input election.*

We provide an analogous result for the Veto rule.

Theorem 3.32. VETO-DCDC *can be solved in* $O(m \cdot n \cdot 2^n)$ *time, where* m *is the total number of candidates and* n *is the number of voters.*

Proof. We use almost the same approach as for Theorem 3.31. First, we guess candidate p whose goal is to have fewer vetoes than d. Deleting candidates can

only increase the number of vetoes that a remaining candidate has. Thus, our algorithm proceeds as follows.

We consider every subset V' of voters that prefer p to d. For each subset V', and for each voter v' in the subset V', we delete all candidates that this voter ranks below d (by choice of V', p is never deleted). If, as a result, d has more vetoes than p, then we return True. Otherwise, we try the next subset of voters. If we did not return True for any of the subsets of voters, then we return False. □

3.E.2. XP Results

We now establish XP results for our W[1]-hard problems. Indeed, this implies that, if the number of voters is a constant, then the problems are polynomial-time solvable.

Theorem 3.33. *For each fixed integer t, $t \geq 1$, and for each control type $\mathcal{K} \in \{CCAC, CCDC\}$, t-Approval-$\mathcal{K}$ and t-Veto-$\mathcal{K}$ can be solved in time $O(m^{tn} \cdot m \cdot n)$, where m is the total number of candidates and n is the number of voters.*

Proof. We consider the CCAC and the CCDC cases jointly, in parallel for both t-Approval and t-Veto. Our algorithm first guesses, for each voter, the set of t candidates that this voter will rank first (for the case of t-Approval) or last (for the case of t-Veto). There are $O(m^{tn})$ possible different guesses.

For each guess, for each voter, we verify which candidates have to be added (for the case of CCAC) or deleted (for the case of DCAC) to ensure that the voter ranks the guessed t candidates on top. If it suffices to add or delete k candidates to implement the guess, and, as a result of implementing the guess, the preferred candidate is a winner, then we return True. Otherwise, we proceed to the next guess. If we did not return True for any guess, then we return False. □

Theorem 3.34. *For each fixed integer t, $t \geq 1$, and each control type $\mathcal{K} \in \{CCAC, DCAC\}$, t-Approval-Comb-$\mathcal{K}$ and t-Veto-Comb-$\mathcal{K}$ can be solved in time $O(m^{2tn} \cdot m \cdot n)$, where m is the total number of candidates and n is the number of voters.*

Proof. We use the same approach as described in the proof of Theorem 3.33, but in addition to guessing the first t candidates for each vote, we also guess, for each added candidate c, which bundle it belongs to. Guessing one bundle is enough since, by guessing these bundles, we indeed guess the solution. □

3.E.3. Polynomial-time Solvable Case

We show the last remaining case, of Maximin-COMB-DCAC, next. It turns out that the polynomial-time algorithm for Maximin-DCAC, presented by Faliszewski et al. [77], can be generalized for the combinatorial case.

Theorem 3.35. MAXIMIN-COMB-DCAC *can be solved in* $O(m^3 \cdot n)$ *time, where* m *is the total number of candidates and* n *is the number of voters.*

Proof. It was shown by Faliszewski et al. [77] that Maximin-DCAC is polynomial-time solvable. The same strategy can be applied for the combinatorial case as well. To this end, let $((C,V),A,d,\kappa,k)$ be an instance of Maximin-COMB-DCAC.

The algorithm is simple and can be described as follows: we guess up to two candidates, add their bundles to the election, and check whether the despised candidate d is no longer a winner; if so, then we return True, and otherwise, we return False.

To see why this simple algorithm is correct, consider a solution, that is, a set A' of at most k unregistered candidates whose bundles are to be added. If A' consists of at most two candidates, then we are done. Otherwise, let us take a closer look at the set A'.

It is clear that, in the resulting election $E' \coloneqq (C \cup \kappa(A'),V)$, d is not a winner. Therefore, there must be at least one other candidate p that has higher score than d. Consider the bundle b_p of some candidate in $\kappa(A')$ such that b_p includes p (indeed, there might be several such candidates; we can choose any one of them arbitrarily; it is also possible that p is present in the original election, in which case we take b_p to be an "empty" bundle). Further, consider some candidate z in $\kappa(A')$ such that the Maximin score of candidate d in the election E' is exactly $N_{E'}(d,z)$. There might be several such candidates and we choose one arbitrarily. Finally, we choose an arbitrary candidate from A' whose bundle b_z includes z (in fact, it is possible that z is present in the original election, in which case we take b_z to be an "empty" bundle).

It is clear that p defeats d in the election $(C \cup \kappa(x,y),V)$ where x and y are the candidates corresponding to the bundles b_p and b_z, respectively (if either of these bundles is "empty", then we simply disregard it). Thus, each yes-instance of Maximin-COMB-DCAC has a solution that consists of at most two candidates and, consequently, it is sufficient to guess and test at most two unregistered candidates. □

4. Combinatorial Shift Bribery

In this chapter, similarly to Chapter 3, we study the possibility of manipulating an election which occurs on top of an underlying social network. In Chapter 3 we consider manipulating a given election by adding or deleting groups of candidates from the election. Here, however, we consider manipulating a given election by changing the way groups of voters vote.

Technically, we study a combinatorial variant of the SHIFT BRIBERY problem in elections. In the standard SHIFT BRIBERY problem, we are given an election where each voter has a preference order over the set of candidates and where an external agent, the briber, can pay each voter to rank the briber's favorite candidate a given number of positions higher. The goal is to ensure the victory of the briber's preferred candidate.

The combinatorial variant of the problem models settings where it is possible to affect the position of the preferred candidate in multiple votes, either positively or negatively, with a single bribery action. That is, there is some underlying structure connecting the voters, such that whenever the external agent changes the position of its preferred candidate in one of the votes, the position of the briber's preferred candidate is changed in some other (related) votes. This variant of the problem is particularly interesting in the context of large-scale campaign management problems (which, from the technical side, are modeled as bribery problems).

We show that the combinatorial variant of the SHIFT BRIBERY problem is highly intractable in general (that is, NP-hard, hard in the parameterized sense, and hard to approximate), and also provide some parameterized algorithms and approximation algorithms for natural restricted cases.

4.1. Illustrating Example

Consider the following example.

Example 5. Recall the group of people discussed in Section 3.1, the social network representing their friendship relationships (Figure 3.1, and given also

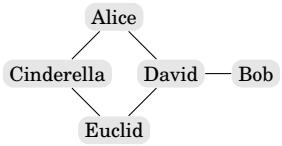


Figure 4.1. The social network used in the illustrating example.

Alice	:	Alice	$\succ$	Cinderella	$\succ$	David	$\succ$	Bob	$\succ$	Euclid
Bob	:	Euclid	$\succ$	Bob	$\succ$	David	$\succ$	Alice	$\succ$	Cinderella
Cinderella	:	Cinderella	$\succ$	Alice	$\succ$	David	$\succ$	Bob	$\succ$	Euclid
David	:	David	$\succ$	Cinderella	$\succ$	Euclid	$\succ$	Alice	$\succ$	Bob
Euclid	:	Cinderella	$\succ$	David	$\succ$	Euclid	$\succ$	Alice	$\succ$	Bob

Figure 4.2. The election used in the illustrating example.

in Figure 4.1 for convenience), and the election performed over their social network (Figure 3.2, and given also in Figure 4.2 for convenience).

In Section 3.1, corresponding to the computational problem considered in Chapter 3, we assume that Bob can choose one person from the group and persuade him (or her) not to participate as a candidate in the election, and as a result, all his (or her) friends would also not participate as candidates in the election.

Here, however, corresponding to the computational problem considered in this chapter, we assume that Bob can persuade some people from the group to change their ranking of Euclid. Specifically, a set of *shift actions* is given, and Bob can choose some shift actions from this set. Each shift action results in shifting the position of Euclid in the preference orders of several voters. A particular set of shift actions is depicted in Figure 4.3 (for example, shift action f_1 would result in Euclid being shifted backwards by one position in the vote of Bob, while shifted forward by one position in the vote of David and Euclid).

We ask the following question: “which shift actions shall Bob choose?”

effect on $Alice$	:	$\begin{pmatrix} 1 \\ -1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ -2 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 0 \\ 1 \\ -1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$
effect on Bob	:					
effect on $Cinderella$	:					
effect on $David$	:					
effect on $Euclid$	:					
		f_1	f_2	f_3	f_4	f_5

Figure 4.3. The shift actions used in the illustrating example.

Alice	:	Alice	$\succ$	Cinderella	$\succ$	David	$\succ$	Bob	$\succ$ Euclid
Bob	:	Bob	$\succ$	David	$\succ$	Alice	$\succ$	Euclid	$\succ$ Cinderella
Cinderella	:	Cinderella	$\succ$	Alice	$\succ$	David	$\succ$	Euclid	$\succ$ Bob
David	:	Euclid	$\succ$	David	$\succ$	Cinderella	$\succ$	Alice	$\succ$ Bob
Euclid	:	Euclid	$\succ$	Cinderella	$\succ$	David	$\succ$	Alice	$\succ$ Bob

Figure 4.4. The resulting election.

In the current example, the answer is f_1, f_3, and f_5. Indeed, the resulting election would be as shown in Figure 4.4, and, under the Plurality voting rule, we will have Euclid as the winner, exactly as desired by Bob. △

4.2. Introduction

In this chapter we study a scenario of election campaign management, for the case where campaign actions (such as airing a TV advertisement, launching a web-based campaign, or organizing meetings with voters) may have large-scale effects which affect multiple voters. Further, we are interested in settings where

these actions can have both positive effects (for example, some voters may choose to rank the promoted candidate higher because they find arguments presented in a given advertisement appealing) as well as negative ones (for example, because some other voters find the advertisement too aggressive). Thus, in our setting, the two major issues faced by a campaign manager are (a) choosing actions that positively affect as many voters as possible and (b) balancing the negative effects of campaigning actions (for example, by concentrating these negative effects on voters who disregard the promoted candidate anyway).

Within computational social choice, the term *campaign management* (introduced by Elkind et al. [61, 62]) is an alternative name for the bribery family of problems (introduced by Faliszewski et al. [74]) for the case where one focuses on modeling actions available during election campaigns: as a result of money spent by a campaign manager, some of the voters change their votes. In this chapter, we study campaign management through the SHIFT BRIBERY problem [27, 61, 62]. In the SHIFT BRIBERY problem, we have a candidate p whom we want to win, for each voter v we have a price $\pi_v(i)$ for which this voter is willing to shift p forward by i positions in her preference order (of course, this price does not necessarily reflect a direct money transfer to the voter, but rather the cost of convincing the voter to change his or her mind), and we ask for the lowest cost of ensuring that p is a winner (see Section 4.2.1 for references to other campaign management problems).

The SHIFT BRIBERY problem has one major drawback as a model for campaign management: it is incapable of capturing large-scale effects of campaign actions. In particular, if one puts forward a TV spot promoting a given candidate, then some voters will react positively and rank the candidate higher, some will be oblivious to it, and some will react negatively, by ranking the candidate lower. The SHIFT BRIBERY problem cannot model such correlated effects. In this chapter we study the COMBINATORIAL SHIFT BRIBERY problem, allowing campaign actions to have effects, positive and negative, on whole groups of voters. Indeed, one might think of these correlated effects as being dependent on some underlying social network, connecting the voters.

We consider two voting rules, the Plurality rule (where we pick the candidate who is ranked first by most voters) and the Borda rule (where each candidate c gets from each voter v as many points as there are candidates that v prefers c to, and we pick the candidate with the most points). These rules are chosen since the Plurality rule is the most widespread rule in practice and since the Borda rule is very well-studied in the context of campaign management.

We are interested in understanding how such a more realistic model of campaign management affects the complexity of the problem. Indeed, the SHIFT BRIBERY problem is, computationally, a very well-behaved problem. For example, for the Plurality rule it is solvable in polynomial time and for the Borda rule it is NP-complete [62], but there is a polynomial-time 2-approximation algorithm [61, 62] and there are parameterized algorithms, either exact or capable of finding solutions arbitrarily close to the optimal ones [27]. In this chapter, we ask to what extent do we retain these good computational properties when we allow large-scale effects. The results are surprising both positively and negatively:

1) COMBINATORIAL SHIFT BRIBERY becomes both NP-complete and W[1]-hard even for the Plurality rule, even for very restrictive choice of parameters, even if the correlated effects of particular campaign actions are limited to at most two voters. Moreover, our hardness results imply that good, general approximation algorithms do not exist when we allow negative effects of campaign actions.

2) In spite of the above, it is still possible to derive relatively good (approximation) algorithms, both for the Plurality rule and for the Borda rule, provided that we restrict the effects of the campaign actions to be only positive and to either only involve few voters each, or to only involve groups of consecutive voters (given an ordering of the voters).

Our results are summarized in Table 6.1. With the generality of our problem and its combinatorial nature it is natural that we obtain many hardness results. Yet, their extent and strength is surprising, and so is the fact that we also find a non-trivial landscape of tractable cases.

4.2.1. Related Work

Our work builds on top of two main research ideas. First, on studying campaign management and bribery problems, and, second, on studying combinatorial variants of election problems (indeed, studying combinatorial variants of election problems is partially the subject of Chapter 3 as well).

The study of the computational complexity of bribery in elections was initiated by Faliszewski et al. [74], and continued by a number of researchers [75, 93, 115, 117]. Elkind et al. [61, 62] realized that the formalism of election bribery problems is useful from the point of view of planning election campaigns. In particular, they defined the SWAP BRIBERY problem and its restricted variant,

SHIFT BRIBERY. In the former, it is possible, at a given price, to swap any two adjacent candidates in a given vote. In the latter, we are only allowed to shift the preferred candidate forward. Various problems, modeling different flavors of campaign management, have been studied, including, for example, the possibility to alter the number of approved candidates [6, 80, 136].

Moreover, different (positive) applications of bribery problems have been revealed. For example, in the MARGIN OF VICTORY view of the problem, the goal of the briber is to prevent some candidate from winning; if it is possible to do so at low cost, then this suggests that the election could have been tampered with [38, 113, 133, 143].

From our point of view, the most related works are those of Elkind et al. [61, 62], Bredereck et al. [27], and Dorn and Schlotter [56]. The former ones study the SHIFT BRIBERY problem (with the work of Bredereck et al. [27] focusing on parameterized complexity of SHIFT BRIBERY), which we generalize, whereas the work of Dorn and Schlotter [56] pioneers the use of parameterized complexity analysis for bribery problems.

Similarly to Chapter 3, the combinatorial variant considered in this chapter is largely inspired by the work of Chen et al. [41], who introduced and studied a combinatorial variant of voter control. Control is a very well-studied topic in computational social choice, initiated by Bartholdi et al. [5]. Control problems, especially those related to adding and deleting voters, are quite relevant to issues of campaign management. Indeed, in Section 4.5 we do show a connection between COMBINATORIAL SHIFT BRIBERY and (combinatorial) control by adding voters [33] (control is the subject of Chapter 3; for a literature review on election control, see Section 3.2.3).

We stress that our use of the term "combinatorial variants of election problems" is different from the one used in the well-established line of work regarding combinatorial candidate spaces (see the survey of Lang and Xia [101] and further work, for example [17, 51, 115]; in short, we refer to the combinatorial structure connecting the voters or the candidates of the given elections, while they refer to voting in multi-issue domains, where the voters vote on combinatorial candidates). In this chapter, we use the term "combinatorial" to refer to the "combinations" of voters affected by each bribery action.

4.3. Specific Preliminaries

In this section, we first define the COMBINATORIAL SHIFT BRIBERY problem in its full generality. Then, we describe why and how we simplify it for the remainder of our study.

4.3.1. The Definition

Let $\mathscr{R}$ be some voting rule. The formal definition of $\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY is somewhat involved, therefore we first define some necessary components. We are given an election $E = (C, V)$ and a preferred candidate $p \in C$. The goal is to ensure that p is an $\mathscr{R}$-winner of the election. To this end, we have a number of possible actions to choose from.

Recall that $m := |C|$ denotes the number of candidates in E and that $n := |V|$ denotes the number of voters in E. A *shift action* f is an n-dimensional vector of (possibly negative) integers, $f = (f^{(1)}, \ldots, f^{(n)})$. In $\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY we are given a family $F = (f_1, \ldots, f_\zeta)$ of shift actions. Each particular shift action models a possible campaigning action, such as airing a TV spot or organizing a meeting with the voters. The components of a given shift action measure the effect of the corresponding campaigning action on the particular voters. For a given subset $F' \subseteq F$ of shift actions, we define the effect of F' on voter v_i $(1 \leq i \leq n)$ as $\mathscr{E}^{(i)}(F') = \sum_{f_j \in F'} f_j^{(i)}$. Further, each shift action f_j $(1 \leq j \leq \zeta)$ comes with a non-negative integer cost $w(f_j)$ for applying it.

Each voter v_i $(1 \leq i \leq n)$ has her individual *threshold function* $\pi_i : \mathbb{Z} \to \mathbb{Z}$ describing how shift actions affect this voter. We require that $\pi_i(0) = 0$ and that π_i is non-decreasing. Let F' be a collection of shift actions. After applying the shift actions from F', each voter v_i $(1 \leq i \leq n)$ shifts the preferred candidate p by $t > 0$ positions forward if (a) $\mathscr{E}^{(i)}(F') > 0$, and (b) $\pi_i(t) \leq \mathscr{E}^{(i)}(F') < \pi_i(t+1)$. The shift is by $t > 0$ positions back if (a) $\mathscr{E}^{(i)}(F') < 0$ and (b) $\pi_i(-t) \geq \mathscr{E}^{(i)}(F') > \pi_i(-t-1)$.

Finally, we are given a non-negative integer B, the *budget*. We ask for the existence of a collection $F' \subseteq F$ of shift actions with total cost $\sum_{f_j \in F'} w(f_j)$ at most B and such that, after applying them, p will be an $\mathscr{R}$-winner of the given election. If this is the case, then we say that F' is *successful*. Consider the following example.

Example 6. Consider the election below, where the set of candidates is $C = \{a, b, c, p\}$, the set of voters is $V = (v_1, v_2, v_3)$, and p is the preferred candidate. There are three available shift actions, each with the same unit cost (that is, $w(f_1) = w(f_2) = w(f_3) = 1$).

$$\begin{array}{cc} \textbf{election} & \textbf{shift actions} \\ v_1: \quad c \succ b \succ p \succ a & \begin{pmatrix} 2 \\ 4 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 6 \\ 0 \\ -3 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \\ v_2: \quad b \succ a \succ c \succ p & \\ v_3: \quad p \succ a \succ b \succ c & \\ & f_1 \quad f_2 \quad f_3 \end{array}$$

The threshold functions are such that:

1) $\pi_1(-1) = -4$, $\pi_1(0) = 0$, $\pi_1(1) = 6$, $\pi_1(2) = 100$.

2) $\pi_2(0) = 0$, $\pi_2(1) = 2$, $\pi_2(2) = \pi_2(3) = 100$.

3) $\pi_3(-3) = \pi_3(-2) = -100$, $\pi_3(-1) = -3$, $\pi_3(0) = 0$.

We use the Borda rule. Candidates a, b, c, and p have, respectively, 4, 6, 4, and 4 points. It is easy to see that applying any single shift action does not ensure p's victory. However, applying shift actions $F' = \{f_2, f_3\}$ results in p being a winner. The total effect of these two shift actions is $(6, 2, -3)$. According to the threshold functions, this means that p is shifted forward by one position in v_1 and v_2, and is shifted back by one position in v_3. After these shifts, the modified election looks as follows:

$$\begin{array}{c} \textbf{election} \\ v_1': \quad c \succ p \succ b \succ a \\ v_2': \quad b \succ a \succ p \succ c \\ v_3': \quad a \succ p \succ b \succ c \end{array}$$

That is, after we apply the shift actions $F' = \{f_2, f_3\}$, we have that candidate c has 3 points, while all other candidates have 5 points each. Thus, a, b, and p are tied as co-winners and F' is indeed a successful set of shift actions. △

Formally, given a voting rule $\mathscr{R}$, we define the $\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY problem as follows.

$\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY
Input: An election $E = (C,V)$, where $C = \{c_1,\ldots,c_m\}$ is the set of candidates and $V = (v_1,\ldots,v_n)$ is the collection of voters, a set $F = \{f_1,\ldots,f_\zeta\}$ of shift actions with costs $w(f_1),\ldots,w(f_\zeta)$, threshold functions $\pi_1,\ldots,\pi_n$, and a non-negative integer budget B. One of the candidates is designated as the preferred candidate p.
Question: Is there a subset $F' \subseteq F$ of shift actions with total cost at most B such that, after we apply the shift actions from F', candidate p is an $\mathscr{R}$-winner of the resulting election?

While this definition is quite complicated, it captures some important features of campaigning. For example, the use of threshold functions allows us to model voters who are unwilling to change the position of the preferred candidate beyond a certain range, irrespective of the strength of the campaign. The fact that different shift actions have different costs models the fact that particular actions (for example, airing TV spots or organizing meetings) may come at different costs.

In this chapter we study also the approximability of the COMBINATORIAL SHIFT BRIBERY problem. Specifically, by approximate solution we mean a solution which causes p to be a winner of the election but possibly uses more budget than given.

4.3.2. Relation to Standard Shift Bribery

It is important to discuss the relation between the COMBINATORIAL SHIFT BRIBERY problem and its non-combinatorial variant, the SHIFT BRIBERY problem [61, 62]. If we restrict our shift actions such that each shift action has a positive entry for exactly one voter, then—in effect—we obtain SHIFT BRIBERY for the case of *convex price functions* [27]. This is a very general variant of the SHIFT BRIBERY problem for which, naturally, all known NP-hardness results hold, but nonetheless not the most general one (a more general variant would allow for arbitrary prices). We decided not to complicate our definition further to obtain a full generalization. As we will see below, doing so would obfuscate the problem without visible gain.

4.3.3. A General Hardness Result

It turns out that the COMBINATORIAL SHIFT BRIBERY problem, as defined above, is so general that it allows for the following, sweeping, hardness result. Note,

however, that we prove weak NP-hardness. That is, our result may not hold if we assume that all occurring numbers are encoded in unary. In contrast, all other hardness proofs in this chapter prove strong NP-hardness results and are independent of such number encoding issues.

Theorem 4.1. *For both the Plurality rule and the Borda rule,* COMBINATORIAL SHIFT BRIBERY *is* NP*-hard even for five voters, two candidates, and no budget constraints. For the Borda rule,* COMBINATORIAL SHIFT BRIBERY *is* NP*-hard even for three voters and four candidates.*

Proof. We reduce from the following (weakly NP-hard) variant of the SUBSET SUM problem (it is an easy exercise to show its NP-hardness through a reduction from the classic SUBSET SUM problem [84]):

> SUBSET SUM (ZERO VARIANT)
> **Input:** A set $A := \{a_1, \dots, a_n\}$ of integers.
> **Question:** Is there a non-empty set $A' \subseteq A$ such that $\sum_{a_i \in A'} a_i = 0$?

Given an instance $A = \{a_1, \dots, a_n\}$ of SUBSET SUM (ZERO VARIANT), we construct an instance of Plurality-COMBINATORIAL SHIFT BRIBERY with two candidates. Since the Plurality rule and the Borda rule coincide for elections with two candidates, our hardness result transfers to Borda-COMBINATORIAL SHIFT BRIBERY (and, in fact, to almost all natural voting rules).

We construct the following election:

election		**shift actions**		
v_1:	$p \succ d$	a_1		a_n
v_2:	$p \succ d$	$-a_1$		$-a_n$
v_3:	$d \succ p$	1	$\cdots$	1
v_4:	$d \succ p$	0		0
v_5:	$d \succ p$	0		0
		f_1	$\cdots$	f_n

That is, for each element $a_i \in A$, F contains one shift action f_i with effect a_i on v_1, effect $-a_i$ on v_2, effect 1 on v_3, and no effect on the other two voters.

The voter threshold functions are as follows. Candidate p is shifted to the last position for v_1 and v_2 if the effect on these voters is negative (that is, we have that $\pi_1(-1) = \pi_2(-1) = -1$). Candidate p is shifted to the top position for the third voter if the effect is positive (that is, we have that $\pi_3(1) = 1$). We set the cost of each shift action to be one and we set our budget to be n. Thus the budget allows us to pick any combination of the shift actions.

For the "if" direction, let $A' \subseteq A$ be a non-empty subset whose element-wise sum equals zero. After applying $F' := \{f_i : a_i \in A'\}$, it holds that p is a winner: since A' sums up to zero, there is no effect on the first two voters. Moreover, the effect on the third voter is positive, since A' is non-empty. Thus, p is preferred by three voters (out of five) and wins the election.

For the "only if" direction, let $F' \subseteq F$ be a subset of shift actions that makes p a winner. Then, F' must be non-empty since p does not win the initial election. We claim that the element-wise sum of $A' := \{a_i : f_i \in F'\}$ is zero. Towards a contradiction, assume that $\sum_{a_i \in A'} a_i \neq 0$. If the sum were negative, then there would be a negative effect on the first voter, d would be preferred by three voters out of five, and d would win the election. If the sum were positive, then we would have the same effect with the second voter taking the role of the first one.

Using a very similar idea, we can show how to reduce SUBSET SUM (ZERO VARIANT) to Borda-COMBINATORIAL SHIFT BRIBERY with three voters and four candidates. Given the same input as before, we construct the following instance:

	election	**shift actions**		
v_1:	$p \succ d_1 \succ d_2 \succ d_3$	$\begin{pmatrix} a_1 \\ -a_1 \\ 1 \end{pmatrix}$	$\cdots$	$\begin{pmatrix} a_n \\ -a_n \\ 1 \end{pmatrix}$
v_2:	$p \succ d_1 \succ d_2 \succ d_3$			
v_3:	$d_1 \succ d_2 \succ d_3 \succ p$			
		f_1	$\cdots$	f_n

That is, for each element $a_i \in A$, F contains one shift action f_i with effect a_i on v_1, effect $-a_i$ on v_2, and effect 1 on v_3. Each voter v_i has the same threshold function $\pi_i(t) = t$. In effect, p is shifted to the last position of the first and of the second voter if the effect on these voters is negative, and is shifted to the top position of the third vote if the effect there is positive. Each shift action has the same unit cost, and we set the budget to n (specifically, we can pick any combination of the shift actions).

Observe that d_1 is the original winner of the election and obtains seven points, whereas p obtains only six points.

For the "if" direction, let $A' \subseteq A$ be a non-empty subset whose element-wise sum equals zero. If we apply the set of shift actions $F' := \{f_i : a_i \in A'\}$ then p becomes a winner: since A' sums up to zero, there is no effect on the first two voters. Moreover, the effect on the third voter is positive since A' is non-empty. Thus, p is the most preferred candidate for all voters and wins the election.

For the "only if" direction, let $F' \subseteq F$ be a subset of shift actions that makes p a winner. Then, F' must be non-empty since p does not win the initial election. We show that the element-wise sum of $A' := \{a_i : f_i \in F'\}$ is zero. Towards a contradiction, assume that $\sum_{a_i \in A'} a_i \neq 0$. If the sum were negative, then there would be a negative effect on the first voter and p would obtain six points, whereas d_1 would obtain seven. If the sum were positive, we would have the same effect with the roles of the first and the second voter switched. □

Effectively, Theorem 4.1 shows that studying large-scale effects of campaign actions through the full-fledged $\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY problem leads to a hopelessly intractable problem: we have hardness even for elections with both a fixed number of candidates and a fixed number of voters.

4.3.4. Restricted Variants of Combinatorial Shift Bribery

Given the hardness results from Theorem 4.1, throughout the remainder of this chapter we focus on restricted variants of the COMBINATORIAL SHIFT BRIBERY problem. We assume the individual threshold functions to be the identity functions (that is, for each voter i and each integer t, we assume that $\pi_i(t) = t$), we assume each shift action to have the same unit cost, and we consider restricted types of shift actions. All these assumptions require some additional discussion.

The restrictions on the threshold functions and on the costs of shift actions seem to be very basic, and, in fact, are even satisfied by the instances built in the proof of Theorem 4.1. The reason for assuming them is that, on the one hand, it seems beyond point to study instances more involved than those from Theorem 4.1, and, on the other hand, they interact with other restrictions, leading to tractable cases. Having said this, they do have important consequences.

First, using identity threshold functions means that we model societies that are prone to propaganda. With identity threshold functions we cannot differentiate between voters based on their responsiveness to our actions. Second, assuming that every shift action has the same unit cost models settings where the costs of

particular campaign actions are similar enough that small differences between them are irrelevant; the actual number of actions we choose to perform is a sufficiently good approximation of the real cost. This is true, for example, for the case of organizing meetings with voters, which often have comparable prices. It is also likely to be the case when shift actions model actions such as airing TV spots: each spot has a similar cost to produce or to broadcast. The greatest disadvantage of assuming unit costs is that we no longer can model mixed campaigns that use actions of several different types (meetings with voters, TV spots, web campaigns, etc.).

The restrictions on the types of allowed shift actions have even greater impact on the nature of campaigns that we study. We study the following classes of shift actions:

Unrestricted Shift Actions. Here we put no restrictions on the allowed shift actions; this models the most general (and, naturally, the least tractable) setting.

Bounded-Effect Shift Actions. Here we consider a parameter Γ and require that, for each shift action $f = (f^{(1)}, \ldots, f^{(n)})$, it holds that for each j $(1 \leq j \leq n)$, we have $|f^{(j)}| \leq \Gamma$. This is still a very general setting, where we assume that each campaigning action has only a limited impact Γ on each voter.

Unit-Effect Shift Actions. This is a class of bounded-effect shift actions for $\Gamma = 1$. For each given voter, applying a given shift action can either leave the preferred candidate p unaffected or it can shift p one position up or down.

Interval Shift Actions. This a subclass of unit-effect shift actions that never affect voters negatively, and where for each shift action there is an interval of voters that are affected positively (the interval is with respect to an ordering of the voters). This class of shift actions models, for example, campaigns associated with a time window where certain voters can be reached or campaigns that are local to given neighborhoods[1] (for example, that include putting up multiple posters, organizing meetings, etc.). We often speak of 1^z-interval shift actions to mean interval shift actions where each shift action affects at most z voters.

[1] In the neighborhood scenario, we take the simplified view that a society of the voters lives on a line. Of course, it would be more natural to take two-dimensional neighborhoods into account. We view this as an interesting direction for future research, but for the time being we consider as simple settings as possible. In the time window scenario, a natural ordering of the voters is the point of time when they cast their votes or can be affected by the campaign.

Unit-Effect on Two Voters Shift Actions. This is a subclass of unit-effect shift actions that affect two voters at most. We focus on shift actions that affect both voters positively, denoted as $(+1,+1)$-shift actions, and that affect one voter positively and one voter negatively, denoted as $(+1,-1)$-shift actions. The reason for studying these families is not because they model particularly natural types of election campaigns, but rather to establish the limits of tractability for our the COMBINATORIAL SHIFT BRIBERY problem. For example, we consider $(+1,-1)$-shift actions to understand how intractable are shift actions that have negative effects; $(+1,-1)$-shift actions are the simplest shift actions of this type that may be useful in the campaign (one would never deliberately use a shift action that only affects the preferred candidate negatively).

Figure 4.5 presents graphically the difference between bounded-effect shift actions, unit-effect shift actions, unit-effect on two voters shift actions, and interval shift actions. As we discuss in the next section, the type of allowed shift actions has a huge impact on the complexity of our problem.

4.4. Overview of Our Results

We now provide a high-level overview of our results. It turns out that even with rather strong restrictions in place (that is, the restrictions defined in Section 4.3.4), COMBINATORIAL SHIFT BRIBERY is computationally hard in most settings. What we present here is our quest for understanding the border between tractability and intractability of COMBINATORIAL SHIFT BRIBERY. To this end, we employ the following techniques and ideas.

1) We seek both regular complexity results (NP-hardness results) and parameterized complexity results (FPT algorithms, W[1]-hardness and W[2]-hardness results, and XP algorithms).

2) We consider structural restrictions on the sets of available shift actions.

3) We seek approximation algorithms and inapproximability results (that is, approximation hardness results).

For our parameterized complexity results, we consider the following parameters: (a) the number n of voters, (b) the number m of candidates, (c) the budget B, (d) the maximum effect Γ of a single shift action, and (e) the maximum number Λ

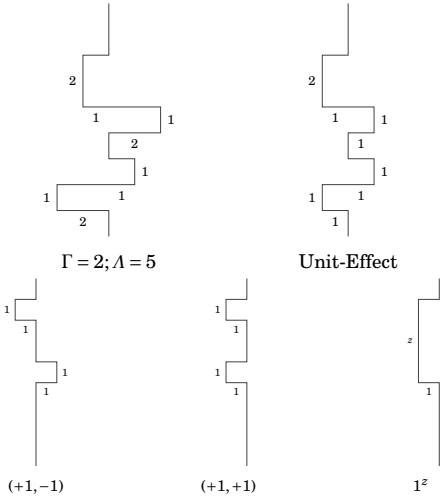


Figure 4.5. Restrictions on the shift actions. We visualize (from left to right and top to bottom) a shift action with maximum effect $\Gamma = 2$ of a single shift action and maximum number $\Lambda = 5$ of voters affected by a single shift action; a unit-effect shift action; a shift action with effect of $+1$ on one voter and effect of -1 on another voter "$(+1, -1)$"; a shift action with effect of $+1$ on two voters "$(+1, +1)$"; and a shift action with effect of $+1$ on an interval of size z "1^z". The intended interpretation is that voters are listed vertically, from top to bottom.

of voters affected by a single shift action. All our discussions of (in)approximability of COMBINATORIAL SHIFT BRIBERY regard the task of approximating the cost of ensuring the preferred candidate's victory (that is, we allow to exceed the budget). This means that a 2-approximation algorithm has to decide if it is possible to ensure the preferred candidate's victory at all and, if so, it has to output a successful set of shift actions whose total cost is at most twice as high as the cost of the optimal one.

We summarize our results in Table 6.1. These results show that COMBINATORIAL SHIFT BRIBERY is highly intractable. Theorem 4.3, Theorem 4.4, and Theorem 4.5, show that the problem is computationally hard (in terms of NP-hardness, W[2]-hardness, and inapproximability even by FPT algorithms) for both the Plurality rule and the Borda rule, even for various very restricted forms

Table 4.1. Overview of our results. We show exact algorithms and approximation algorithms for Plurality-COMBINATORIAL SHIFT BRIBERY and for Borda-COMBINATORIAL SHIFT BRIBERY, for different restrictions on the shift actions (see Figure 4.5). We consider parameterizations by the number n of voters, the number m of candidates, and the budget B. Note that all variants considered in this chapter are XP when parameterized by the budget B (Observation 4.1).

Shift actions	Rule	Exact complexity	Approximability
Regular SHIFT BRIBERY (convex prices)	Plurality	polynomial-time solvable ([62])	—
	Borda	NP-complete but FPT for B [27, 62]	2-approximable in poly. time [61, 62] FPT-approximation scheme for n [27]
Unit effect	Both	W[2]-h for B even if $m = 2$ (Thm. 4.3) XP for n (Prop. 4.1)	inapproximable even in FPT-time for B and $m = 2$ (Thm. 4.4)
$(+1, -1)$	Plurality	FPT for n (Thm. 4.9)	—
	Borda	W[1]-hard for n (Thm. 4.7)	inapproximable even in FPT-time for n (Cor. 4.2)
	Both	NP-h even if $m = 2$ (Thm. 4.5) W[1]-h for B and m combined (Thm. 4.6)	inapproximable even if $m = 2$ (Thm. 4.5)
$(+1, +1)$	Plurality	FPT for n (Thm 4.9)	—
	Both	W[1]-h for B and m combined (Thm. 4.6)	2-approximable in poly. time (Thm. 4.11)
1^z-intervals	Plurality	FPT for n (Thm. 4.9)	z-approximable in poly. time (Thm. 4.10)
	Borda	—	$2z$-approximable in poly. time (Thm. 4.10)
	Both	W[1]-h for B (Thm. 4.8)	2-approximable in m^z time (Thm. 4.12)

of unit-effect shift actions, even for two candidates. This means that, in essence, the problem is hard for all natural voting rules, since for two candidates all natural voting rules boil down to the Plurality rule.

Further, Theorem 4.6 and Theorem 4.8 show that our problems are W[1]-hard even if we take the number of candidates and the budget as a joint parameter, even for extremely restricted shift actions. The problem remains hard (for the case of the Borda rule) when parameterized by the number of voters, as shown in Theorem 4.7. In contrast, for the case of Plurality we obtain tractability, when parameterizing by the number n of voters.

We obtain several approximability results. In essence, these results are possible only for the cases where shift actions do not have negative results. An intuitive reason for this fact is that when shift actions have negative effects then it is computationally hard to check whether the preferred candidate can win even without any restrictions on the budget. Our approximability have one more interesting feature: essentially all of them are based on results for the non-combinatorial variant of the problem, due to Elkind et al. [61, 62]. Either we use the non-combinatorial algorithm directly as a subroutine in our algorithms, or we derive our results by plugging COMBINATORIAL SHIFT BRIBERY-specific blocks into the framework described by Elkind et al. [61, 62].

4.4.1. Organization of this Chapter

The remainder of this chapter has the following structure. First, in Section 4.5, we show a relation between Plurality-COMBINATORIAL SHIFT BRIBERY and the problem of combinatorial control by adding voters, establishing quite general hardness results (which already apply to unit-effect shift actions). Then, in Section 4.6, we present a series of strong hardness results covering all the classes of shift actions discussed in this chapter, for very restrictive sets of parameters (for example, many of our results already apply to the case of two candidates). Then, in Section 4.7 and in Section 4.8, we present some ways of dealing with our hardness results. Some of our proofs are available in the appendix to this chapter (either when a given proof relies on ideas already presented in other proofs, or—as in the case of Theorem 4.7—when the proof is particularly involved).

4.5. Connection to Combinatorial Control

The study of combinatorial variants of problems modeling ways of affecting election results was initiated by Chen et al. [33], who considered combinatorial control by adding voters (COMBINATORIAL-CCAV), for the Plurality rule and for the Condorcet rule. It turns out that for the Plurality rule we can reduce the problem of (COMBINATORIAL) CCAV to that of (COMBINATORIAL) SHIFT BRIBERY. For the non-combinatorial variants of these problems this does not give much since both problems are easily seen to be polynomial-time solvable. However, there are strong hardness results for Plurality-COMBINATORIAL-CCAV which we can transfer to the case of Plurality-COMBINATORIAL SHIFT BRIBERY. Formally, Plurality-COMBINATORIAL-CCAV is defined as follows [33].

> PLURALITY-COMBINATORIAL-CCAV
> **Input:** A set C of candidates with a preferred candidate $p \in C$, a collection V of registered voters (having preference orders over C), a collection W of unregistered voters (having preference orders over C), a bundling function $\kappa\colon W \to 2^W$ (for each $w \in W$ it holds that $w \in \kappa(w)$), and a budget k.
> **Question:** Is there a collection $W' \subseteq W$ of at most k voters such that p is a winner of the modified election $(C, V \cup \bigcup_{w' \in W'} \kappa(w'))$?

Intuitively, for each unregistered voter $w \in W$, we have her bundle, $\kappa(w)$ (given explicitly in the input), such that when we add w to the election (for example, by somehow convincing her to vote), all the voters in her bundle also join the election (for example, people choose to vote under an influence of a friend). Indeed, the problem discussed in Chapter 3 is the analog of COMBINATORIAL-CCAV, when we control the set of candidates in the election, and not the set of voters.

We next show a relation between COMBINATORIAL-CCAV and COMBINATORIAL SHIFT BRIBERY.

Theorem 4.2. PLURALITY-COMBINATORIAL-CCAV *is polynomial-time many-one reducible to* PLURALITY-COMBINATORIAL SHIFT BRIBERY. *For an instance of* PLURALITY-COMBINATORIAL-CCAV *with m candidates, the reduction outputs an instance of* PLURALITY-COMBINATORIAL SHIFT BRIBERY *with $m+1$ candidates.*

Proof. Consider an input instance of Plurality-COMBINATORIAL-CCAV with candidate set C, collection of registered voters V, collection of unregistered voters W, bundling function κ, preferred candidate $p \in C$, and a budget k. We create an instance of Plurality-COMBINATORIAL SHIFT BRIBERY, as follows.

We form a candidate set $C' = C \cup \{d\}$, where d is some new candidate. We form the set of voters V' in the following way.

1) For each voter $v \in V$, we include v in V', with the preference orders extended to rank d last.

2) For each voter $w \in W$ that ranks p first, we include in V' two voters: x_w, with preference order of the form $d \succ p \succ \cdots$, and x'_w, with preference order of the form $p \succ d \succ \cdots$.

3) For each voter $w \in W$ that ranks some candidate $c \in C \setminus \{p\}$ first, we include in V' voter x_w with preference order $p \succ c \succ \cdots$, and voter x'_w with preference order $d \succ p \succ \cdots$.

4) We include $4|W||C|$ voters in V' with preference orders such that: (a) for each $c \in C$ with a given score $s(c)$ in election (C,V), we have that c is ranked first by $4|W| + s(c)$ voters in V', and (b) d is ranked first by exactly $2|W|$ voters in V'. To achieve this, for each $c \in C \setminus \{p\}$, we include $4|W|$ voters that rank c first, we include $3|W|$ voters that rank p first, and we include $|W|$ voters that rank d first.

For each voter $w \in W$, we introduce a shift action f_w with the following effects: for each $w' \in \kappa(w)$, if w' ranks p first then f_w has effect 1 on $x_{w'}$ (but not on $x'_{w'}$) and if w' ranks some candidate in $C \setminus \{p\}$ first, then f_w has effect -1 on $x_{w'}$ and effect $+1$ on $x'_{w'}$. This finishes the construction.

We provide the proof of correctness after the following example of the reduction.

Example 7. Consider the following input to Plurality-COMBINATORIAL-CCAV, where the preferred candidate is p and the budget k is 1.

registered voters	**unregistered voters**	**bundling function**
v_1 : $p \succ a$	w_1 : $p \succ a$	$\kappa(w_1) = \{w_1, w_3\}$
v_2 : $a \succ p$	w_2 : $a \succ p$	$\kappa(w_2) = \{w_2\}$
v_3 : $a \succ p$	w_3 : $p \succ a$	$\kappa(w_3) = \{w_2, w_3\}$

The input for Plurality-COMBINATORIAL SHIFT BRIBERY which we construct is depicted in Figure 4.6. (Note that the number of entries of each shift action is 33.)

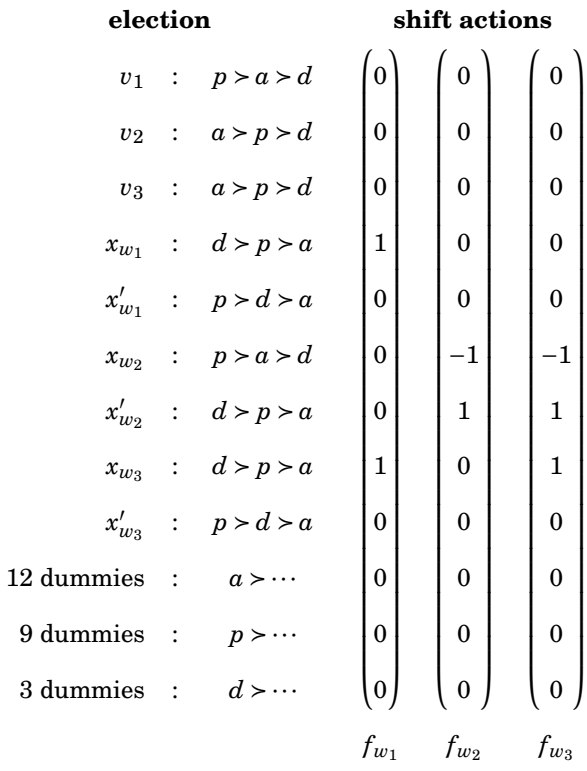


Figure 4.6. The input for Plurality-COMBINATORIAL SHIFT BRIBERY which we construct in Example 7. (Note that the number of entries of each shift action is 33.)

Note that adding the voter w_1 to the input election for Plurality-COMBINATORIAL-CCAV results in p being a winner of the election. Correspondingly, applying shift action f_{w_1} results in p being a winner of the input election for Plurality-COMBINATORIAL SHIFT BRIBERY. △

To see the correctness of the reduction, note that applying a shift action corresponding to a bundle of a voter $w \in W$ has the same effect on the differences between the scores of the candidates in C as adding the bundle $\kappa(w)$ has in the original control instance. More specifically, disregarding the score of d for now, we have the following. For each $w' \in \kappa(w)$ which ranks p first, we have an increase of the score of p by one, while for each $w' \in \kappa(w)$ which ranks some candidate $c \in C \setminus \{p\}$ first, we have an increase of the score of c by one. Further, the score of candidate d can never grow beyond $4|W|$ in our Plurality-COMBINATORIAL SHIFT BRIBERY instance and the score of p can never fall below $4|W|$. Therefore, d can never prevent p from being a winner.

Thus, the reduction is correct. Furthermore, the reduction can be computed in polynomial time and it outputs a Plurality-COMBINATORIAL SHIFT BRIBERY instance with one candidate more than the input Plurality-COMBINATORIAL-CCAV instance. We also observe that the output instance uses unit-effect shift actions that affect at most twice as many voters as the largest bundle in the input instance. □

Based on the proof of Theorem 4.2 and results of Chen et al. [33], we obtain the following result.

Corollary 4.1. *Plurality-*COMBINATORIAL SHIFT BRIBERY *is* W[2]*-hard with respect to the budget B even if $m = 3$, it is* W[1]*-hard with respect to B even for shift actions with unit effect on up to* 6 *voters, and it is* NP*-hard even for shift actions with unit effects on up to* 4 *voters.*

Proof. The result follows by applying the reduction from the proof of Theorem 4.2 to the Plurality-COMBINATORIAL-CCAV instances produced in the reductions from Theorem 2, Theorem 1, and Theorem 4, of Chen et al. [33], respectively. □

4.6. Hardness Results

The results from the previous section show that we are bound to hit hard instances for COMBINATORIAL SHIFT BRIBERY even in very restricted settings. In this

section we explore how restrictive these hard settings are. Our results are organized by the type of shift actions allowed.

4.6.1. Results for General Unit-Effect Shift Actions

We start by considering unit-effect shift actions. If the allowed effects are positive only, then we obtain NP-hardness and W[2]-hardness when parameterizing by the budget B. If we allow also negative unit-effects, we go beyond any hope for an approximation algorithm, even if the approximation algorithm were allowed to run in FPT-time when parameterizing by the budget B. Quite strikingly, these results hold even if we only have two candidates.

Theorem 4.3. *Both for the Plurality rule and for the Borda rule,* COMBINATORIAL SHIFT BRIBERY *is* NP*-hard and* W[2]*-hard for the budget B, even for two candidates and even if each shift action has an effect of either $+1$ or 0 on each voter.*

Proof. We provide a reduction from the W[2]-complete SET COVER problem parameterized by the solution size [57].

> SET COVER
> **Input:** A universe of elements X, a collection $\mathcal{S}$ of sets of elements of X, and a budget h.
> **Question:** Is there a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ of sets such that $|\mathcal{S}'| \leq h$ and $\bigcup_{S \in \mathcal{S}'} S = X$?

Let $(\mathcal{S}, X, h)$ be an instance of SET COVER. We construct an instance of Plurality-COMBINATORIAL SHIFT BRIBERY with two candidates. Note that, since the Borda rule and the Plurality rule coincide on elections with two candidates, our hardness result transfers to Borda-COMBINATORIAL SHIFT BRIBERY.

The construction is as follows. We have only two candidates, d and p. For each element $x_i \in X$, we create an *element voter* v_i with preference order $d \succ p$. We create another set of n *dummy voters*, all with preference order $d \succ p$. The set F of shift actions contains, for each set $S_j \in \mathcal{S}$, a function f_j having an effect of $+1$ on the element voters corresponding to the elements of the set (that is, $f_j[i] = 1$ if $x_i \in S_j$ and $f_j[i] = 0$ otherwise).

Finally, we set the budget B to be h. This finishes the construction. Clearly, the reduction can be computed in polynomial time. Consider the following example of applying it.

Example 8. Let the input to SET COVER be such that $X = \{x_1, x_2, x_3, x_4, x_5\}$ and $S = \{S_1, S_2, S_3\}$, with $S_1 = \{1, 2, 5\}$, $S_2 = \{2, 3\}$, $S_3 = \{3, 4\}$, and $h = 2$. We construct the following input for Plurality-COMBINATORIAL SHIFT BRIBERY.

election			**shift actions**		
v_1	:	$d \succ p$	$\begin{pmatrix}1\\1\\0\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}0\\1\\1\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\\1\\0\\0\end{pmatrix}$
v_2	:	$d \succ p$			
v_3	:	$d \succ p$			
v_4	:	$d \succ p$			
v_5	:	$d \succ p$			
5 dummies	:	$d \succ p$			
			f_1	f_2	f_3

Note that $\{S_1, S_3\}$ is a set cover, and, analogously, choosing f_1 and f_3 results in p being a winner of the election. △

It remains to show that there is a set cover of size h if and only if there is a successful set of shift actions of size h.

For the "if" direction, assume that there is a set cover $\mathscr{S}'$ of size at most h. Then, applying $F' = \{f_j : S_j \in \mathscr{S}'\}$ makes p win the election: since $\mathscr{S}'$ is a set cover, p will be the preferred candidate of all the element voters and, hence, a winner of the election.

For the "only if" direction, assume that there is a set of shift actions $F' \subseteq F$ of size at most h whose application makes p win the election. Then, p must be the preferred candidate of all the element voters in the bribed election, since no shift action has effect on any dummy voter. Since there are n element voters and n dummy voters, $\mathscr{S}' := \{S_j : f_j \in F'\}$ is a set cover. Finally, since $B = h$, it follows that S' is of size at most h. □

Allowing also negative (but unit) effects on the voters, we can adapt our reduction from Theorem 4.3 to show a strong inapproximability result. The inapproximability result follows since in the corresponding reduction, for yes-instances, the only correct solutions use the exact given budget. Notice that, in order to

get this inapproximability result, we allow negative effects and not only positive effects, as we do in the last result.

Theorem 4.4. *Assuming* W[2] ≠ FPT, COMBINATORIAL SHIFT BRIBERY *is inapproximable, even in* FPT*-time with respect to the budget B, both for the Plurality rule and for the Borda rule, even for elections with only two candidates and when allowing only unit-effect shift actions.*

Proof. We modify the reduction from Theorem 4.3 to show our inapproximability result. Let $(\mathcal{S}, X, h)$ be a SET COVER instance where $\mathcal{S} = \{S_1, \ldots, S_m\}$ and $X = \{x_1, \ldots, x_n\}$. Without loss of generality, we assume that $|\mathcal{S}| > h$. We construct an instance of Plurality-COMBINATORIAL SHIFT BRIBERY with two candidates as follows. (Since we have two candidates only, the proof applies to the case of Borda-COMBINATORIAL SHIFT BRIBERY as well.)

For each element $x_i \in X$, we create $|\mathcal{S}|$ *element voters* $v_i^1, \ldots, v_i^{|\mathcal{S}|}$, each with preference order $d \succ p$, and for each set $S_j \in \mathcal{S}$ we create a *set voter* v_j^0 with preference order $p \succ d$. We create $|\mathcal{S}| \cdot |X| + |\mathcal{S}| - 2h$ *dummy voters*, each with preference order $d \succ p$. The set F of shift actions contains, for each set S_j, a shift action f_j having an effect of 1 on each element voter corresponding to an element of the set and an effect of -1 on the set voter corresponding to the set. Finally, we set the budget B to be h. This completes the construction, which is clearly computable in polynomial time.

Next, we show that there is a successful set of shift actions of size h if and only if there is a set cover of size h.

For the "if" direction, assume that there is a set cover $\mathcal{S}'$ of size at most h. Then, $F' = \{f_j : S_j \in \mathcal{S}'\}$ is a successful set of shift actions: since $\mathcal{S}'$ is a set cover, p will be the preferred candidate of all $|\mathcal{S}| \cdot |X|$ element voters and also the preferred candidate for at least $|\mathcal{S}| - h$ set voters (corresponding to the sets not from the set cover). Moreover, d will be the preferred candidate for all $|\mathcal{S}| \cdot |X| + |\mathcal{S}| - 2h$ dummy voters and also the preferred candidate for at most h set voters (corresponding to the sets from the set cover). Hence, either p wins or p and d tie as winners.

For the "only if" direction, assume that there is a successful set of shift actions $F' \subseteq F$ of size at most h. Then, p must be the preferred candidate for all element voters in the bribed election: if there was an element voter with $d \succ p$, then there would be at least $|\mathcal{S}| - 1$ further element voters with $d \succ p$ (the element voters corresponding to the same element). Thus, there would be in total at most $|\mathcal{S}|(|X| - 1)$ element voters and $|\mathcal{S}|$ set voters that prefer p, but at least $|\mathcal{S}| \cdot |X| + |\mathcal{S}| - 2h$ dummy voters and $|\mathcal{S}|$ element voters that prefer d.

Since we assumed that $|\mathscr{S}| > h$, this would mean that p is not a winner. Thus, it must be that $\mathscr{S}' := \{S_j : f_j \in F'\}$ is a set cover, and, due to the budget constraint, it follows that $|\mathscr{S}'| \leq h$.

Finally, we show that Plurality-COMBINATORIAL SHIFT BRIBERY is inapproximable even in FPT-time when parameterized by the budget B. Assume, towards a contradiction, that a successful set of shift actions $F' \subseteq F$ with $|F'| > B$ exists. Then, in the bribed election, at least $|\mathscr{S}| \cdot |X| + |\mathscr{S}| - 2h$ dummy voters and also $|F'| \geq h+1$ set voters prefer d, but at most $|\mathscr{S}| \cdot |X|$ element voters and at most $|\mathscr{S}| - (h+1)$ set voters prefer p. Thus, d is the unique winner. Hence, *any* successful bribery action must be optimal with respect to the budget and any FPT-algorithm for Plurality-COMBINATORIAL SHIFT BRIBERY (parameterized by the budget B) would solve the W[2]-hard problem SET COVER (parameterized by the solution size) in FPT-time; a contradiction to the assumption that FPT $\neq$ W[2]. □

4.6.2. Results for Shift Actions with Unit Effect on Two Voters

In the previous section we did not limit the number of voters affected by each shift action. Now we focus on the case where each unit-effect shift action can affect at most two voters. First, we show that COMBINATORIAL SHIFT BRIBERY remains NP-hard and hard to approximate for $(+1,-1)$-shift actions. Then, we provide parameterized hardness results for both $(+1,-1)$ and $(+1,+1)$-shift actions. The proof of the next theorem is relatively similar to the one for Theorem 4.4 and so we present it in the appendix to this chapter.

Theorem 4.5. *Assuming* P $\neq$ NP, COMBINATORIAL SHIFT BRIBERY *is inapproximable (in polynomial time) both for the Plurality rule and for the Borda rule, even for elections with only two candidates and when allowing only* $(+1,-1)$*-shift actions.*

As opposed to Theorem 4.4, the above result does not yield W[2]-hardness for the parameter budget B. This is so since our proof uses a reduction from the SET COVER problem in which the value of the budget is the size of the universe set X. If we insist on parameterized hardness for unit-effects on two voters, then we have to accept larger sets of candidates. This increase, however, is not too large: below we show W[1]-hardness of COMBINATORIAL SHIFT BRIBERY jointly parameterized by the budget and the number of candidates.

Theorem 4.6. *Both for the Plurality rule and for the Borda rule,* COMBINATORIAL SHIFT BRIBERY *is* W[1]*-hard for the combined parameter* (m, B)*, even if we either only allow* $(+1,-1)$*-shift actions or only allow* $(+1,+1)$*-shift actions.*

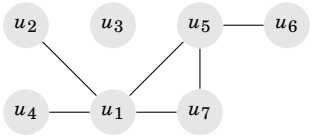
Proof. We have four cases to consider. We begin with the Plurality rule and $(+1,+1)$-shift actions and continue to the other cases.

The Plurality Rule with $(+1,+1)$-Shift Actions. We describe a reduction from the W[1]-hard CLIQUE problem parameterized by the clique size [57].

> CLIQUE
> **Input:** An undirected graph G and an integer h.
> **Question:** Is there a set of h pairwise adjacent vertices?

Let (G, h) be an instance of CLIQUE with $V(G) = \{u_1, \ldots, u_{n'}\}$ and $E(G) = \{e_1, \ldots, e_{m'}\}$. We create the following instance of Plurality-COMBINATORIAL SHIFT BRIBERY with $(+1,+1)$-shift actions, parameterized by (m, B). The set of candidates we create is $\{p\} \cup D$, where $D = \{d_1, \ldots, d_{h-1}\}$. For each vertex $u_i \in V(G)$, we create a *vertex voter* v_i with preference order $\overrightarrow{D} \succ p$. Moreover, we create $n' - 2h$ *dummy voters* with preference order $p \succ \overrightarrow{D}$ each. For each edge $\{u_i, u_j\} \in E(G)$, we create a shift action $f_{\{u_i,u_j\}}$ with effect 1 on the vertex voters v_i and v_j, and effect 0 on all other voters. Finally, we set the budget to $B := \binom{h}{2}$. This completes the construction, which is computable in polynomial time. Consider the following example.

Example 9. We have the following graph, where we are looking for a clique of size $h = 3$.



We construct the following input for Plurality-COMBINATORIAL SHIFT BRIBERY.

election			**shift actions**					
v_1	:	$d_1 \succ d_2 \succ p$	$\begin{pmatrix}1\\1\\0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\\1\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\\0\\0\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\1\\1\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\1\\0\\1\\0\end{pmatrix}$
v_2	:	$d_1 \succ d_2 \succ p$						
v_3	:	$d_1 \succ d_2 \succ p$						
v_4	:	$d_1 \succ d_2 \succ p$						
v_5	:	$d_1 \succ d_2 \succ p$						
v_6	:	$d_1 \succ d_2 \succ p$						
v_7	:	$d_1 \succ d_2 \succ p$						
1 dummy	:	$p \succ d_1 \succ d_2$						
			f_{u_1,u_2}	f_{u_1,u_4}	f_{u_1,u_5}	f_{u_1,u_7}	f_{u_5,u_6}	f_{u_5,u_7}

Note that (v_1, v_5, v_7) form a clique of size 3 in the input graph for CLIQUE, and, accordingly, applying the set of shift actions $\{f_{u_1,u_5}, f_{u_1,u_7}, f_{u_5,u_7}\}$ results in p being the winner of the election for Plurality-COMBINATORIAL SHIFT BRIBERY. △

Without loss of generality, assume that d_1 is ranked first in the (arbitrary but fixed) order $\vec{D}$. Observe that we have n' vertex voters and h dummy voters which rank d_1 first. We also have $n' - h$ dummy voters which rank p first. Hence, to make p win the election, one needs h additional voters to rank p first (and, in effect, not rank d_1 first).

It remains to show that our constructed instance contains a successful set of shift actions F' of size h if and only if (G, h) contains a clique of size h.

For the "if" direction, let $H \subseteq V(G)$ be a set of h vertices forming a clique and let $E' \subseteq E(G)$ be the set of edges between the vertices from H. Then, observe that $F' = \{f_{\{u_i,u_j\}} : \{u_i, u_j\} \in E'\}$ is a successful set of shift actions: for each vertex voter v_i corresponding to a clique vertex $u_i \in H$, candidate p is shifted $h-1$ positions forward. This means that, in total, we have h vertex voters rank p first and p ties as a winner of the election.

For the "only if" direction, let F' be a successful set of shift actions. Since dummy voters are not affected by any shift action, it follows that, in order to

make p a winner of the election, p must be shifted to the top position in at least h vertex voters. That is, in total, p must be shifted $h \cdot (h-1)$ positions forward. Since the size of F' is at most $B = \binom{h}{2} = h \cdot (h-1)/2$ and since each shift action affects only two vertex voters, it follows that F' must be of size exactly $\binom{h}{2}$, affecting exactly h vertex voters. By construction, this implies that there are $\binom{h}{2}$ edges in G incident to exactly h different vertices which is only possible if these h vertices form a clique. This finishes the proof for the Plurality rule with $(+1,+1)$-shift actions.

The remaining cases of the proof are, technically, more involved, but are quite similar in nature, and we present them in the appendix to this chapter. □

It is quite natural to also consider COMBINATORIAL SHIFT BRIBERY from a different perspective. Instead of asking what happens for a small number of candidates, we might ask about the complexity of COMBINATORIAL SHIFT BRIBERY for a small number of voters (see Chapter 3, which concentrates on such elections, for some motivation as to why looking at elections with few voters is interesting). In this case we obtain hardness only for the Borda rule. Indeed, later we will show that Plurality-COMBINATORIAL SHIFT BRIBERY is FPT for this parameter. Quite interestingly, the complexity of the non-combinatorial variant of Borda-SHIFT-BRIBERY, parameterized by the number of voters was only recently settled by Bredereck et al. [23], which showed that the is W[1]-hard. Bredereck et al. [27] show that there is an FPT approximation scheme for this case, but do not give hardness result for it (however, they do show W[1]-hardness of SHIFT-BRIBERY for the Copeland rule).

The proof of the next theorem is quite involved and we present it in the appendix to this chapter. It builds upon a reduction from a variant of the MULTICOLORED CLIQUE problem. It is worth mentioning that, while the reduction is quite different from the reductions described in Section 3.4, it shares some common ideas with those reductions, mainly the fact that we create a constant number of voters for each color and for each color pair.

Theorem 4.7. *Borda-*COMBINATORIAL SHIFT BRIBERY *is* W[1]*-hard with respect to the number n of voters, even when there are no budget constraints and even if we only allow* $(+1,-1)$*-shift actions.*

The proof of Theorem 4.7 describes a reduction from the STRONGLY REGULAR MULTICOLORED CLIQUE problem. Importantly, the reduction does not use budget constraints. Thus, it follows that any approximation algorithm for Borda-COMBINATORIAL SHIFT BRIBERY (running in FPT-time when parameterized by

the number of voters) would yield an FPT algorithm for STRONGLY REGULAR MULTICOLORED CLIQUE, parameterized by the solution size. In effect, we have the following corollary.

Corollary 4.2. *Unless* W[1] = FPT*,* *Borda*-COMBINATORIAL SHIFT BRIBERY *is inapproximable even in* FPT*-time for the parameter n, even for* $(+1,-1)$*-shift actions.*

4.6.3. Results for Interval Shift Actions

We conclude the discussion of the hardness results by considering COMBINATORIAL SHIFT BRIBERY with interval shift actions. In the previous section we allowed shift actions to have non-zero effects on two voters each, but these two voters could have been chosen arbitrarily. Next we show a hardness result for the case where we can positively affect multiple voters, but these voters have to form a consecutive interval in the input election.

Theorem 4.8. *Both for the Plurality rule and for the Borda rule,* COMBINATORIAL SHIFT BRIBERY *is* NP*-hard even if we only allow interval shift actions.*

Proof. We consider the Plurality rule first and give a many-one reduction from the following variant of the strongly NP-hard NUMERICAL MATCHING WITH TARGET SUMS problem.

RESTRICTED NUMERICAL MATCHING WITH TARGET SUMS
Input: Three sets of integers $A = \{a_1,\ldots,a_t\}$, $B = \{b_1,\ldots,b_t\}$, and $X = \{x_1,\ldots,x_t\}$, where (1) the numbers are encoded in unary, (2) all the $3t$ numbers are pairwise distinct, and (3) no two numbers that are both from A or both from B sum up to any number in X.
Question: Can the elements of A and B be paired so that, for each $i \in [t]$, the sum of the ith pair is exactly x_i?

The standard variant of the problem, as presented in the classic text of Garey and Johnson [84], does not have any restrictions on the integers in sets A, B, and X. We can assume that the numbers are encoded in unary since the problem is strongly NP-hard. Further, Hulett et al. [99] have shown that the problem remains NP-hard for the case where all the $3t$ integers are distinct. Finally, to see that the third restriction does not change the complexity of the problem, it suffices to consider the following transformation: given an instance (A,B,X) of RESTRICTED NUMERICAL MATCHING WITH TARGET SUMS, we increment each integer in B and in X by $2 \cdot \max(A \cup B \cup X) + 1$. This produces an equivalent instance where no two numbers, both from A or both from B, sum up to any number in X.

The Plurality Rule. Let (A,B,X) be an instance of RESTRICTED NUMERICAL MATCHING WITH TARGET SUMS and let y denote the largest integer in $A \cup B \cup X$. We create an instance of Plurality-COMBINATORIAL SHIFT BRIBERY as follows. The set of candidates is:

$$C := \{p, d, c_1^a,\ldots,c_t^a, c_1^b,\ldots,c_t^b, c_1^x,\ldots,c_t^x\}.$$

We create the following voters:

1) For each pair of integers $a_i \in A$ and $x_\ell \in X$, we introduce:
   a) One voter with preference order
   $$c_i^a \succ p \succ \overrightarrow{C \setminus \{p, c_i^a\}},$$
   b) a_i voters each with preference order
   $$c_\ell^x \succ p \succ \overrightarrow{C \setminus \{p, c_\ell^x\}},$$

c) $2y - (a_i + 1)$ voters each with preference order

$$d \succ p \succ \overrightarrow{C \setminus \{p, d\}}.$$

These voters are called the (a_i, x_ℓ)-voters and there are exactly $2y$ of them. For each pair (a_i, x_ℓ), we construct a shift action $f_{a_i}^{x_\ell}$ with effect 1 on exactly the set of (a_i, x_ℓ) voters.

2) For each pair of integers $b_j \in B$ and $x_\ell \in X$, we introduce:

a) One voter with preference order

$$c_j^b \succ p \succ \overrightarrow{C \setminus \{p, c_j^b\}},$$

b) b_j voters each with preference order

$$c_\ell^x \succ p \succ \overrightarrow{C \setminus \{p, c_\ell^x\}},$$

c) $2y - (b_j + 1)$ voters each with preference order

$$d \succ p \succ \overrightarrow{C \setminus \{p, d\}}.$$

These voters are called the (b_j, x_ℓ)-voters and there are exactly $2y$ of them. For each pair (b_j, x_ℓ), we construct a shift action $f_{b_j}^{x_\ell}$ with effect 1 on exactly the set of (b_j, x_ℓ) voters.

3) Let $q := 4ty$. We create sufficiently many *dummy voters* to ensure that, altogether, the candidates have the following scores:

a) p has q points,

b) for each i, c_i^a and c_i^b have $q + 4ty + 1$ points each, and

c) for each $\ell \in [t]$, c_ℓ^x has $q + 4ty + x_\ell$ points.

No shift action affects any of the dummy voters.

Finally, we set the budget $B := 2t$. This completes the construction. It is easy to see that it is computable in polynomial time (since all numbers are encoded in unary) and that we can order the voters so that each shift action affects a consecutive interval of $z := 2y$ voters.

It remains to show that our constructed instance of Plurality-COMBINATORIAL SHIFT BRIBERY contains a successful set F' of shift actions of size at most $2t$ if and only if (A,B,X) is a yes-instance of RESTRICTED NUMERICAL MATCHING WITH TARGET SUMS.

For the "if" direction, let $S := \{(a_{i_1}, b_{j_1}), \ldots, (a_{i_t}, b_{j_t})\}$ be a solution for RESTRICTED NUMERICAL MATCHING WITH TARGET SUMS, that is, a set of integer pairs such that each integer from $A \cup B$ occurs exactly once in S and such that $a_{i_\ell} + b_{j_\ell} = x_\ell$ holds for each $\ell \in [t]$. Observe that $F' := \{f_{a_{i_\ell}}^{x_\ell}, f_{b_{j_\ell}}^{x_\ell} : (a_{i_\ell}, b_{j_\ell}) \in S\}$ is a successful set of shift actions: since each integer from $A \cup B$ occurs exactly once in (some pair of) S, it holds that each candidate c_i^a and each candidate c_j^b loses one point. Moreover, since $a_{i_\ell} + b_{j_\ell} = x_\ell$ for each $\ell \in [t]$, it holds that each candidate c_ℓ^x loses x_ℓ points. By construction, p gains $4ty$ points from any set of shift actions of size $2t$. Thus, p wins the election.

For the "only if" direction, let F' be a successful set of shift actions of size $2t$ (if there was a successful set of shift actions of smaller size, then we could extend it to size $2t$ since our shift actions do not have negative effects). After applying the shift actions from F', p gains $4ty$ points. In order to make p a winner of the election, each candidate c_i^a and each candidate c_j^b needs to lose one point, and each candidate c_ℓ^x needs to lose x_ℓ points. Thus, for each $a_i \in A$, there is exactly one $f_{a_i}^{x_{\ell_i}} \in F'$ and, for each $b_j \in B$, there is exactly one $f_{b_j}^{x_{\ell_i}} \in F'$. Since all the integers in $A \cup B \cup X$ are distinct and since no two integers both from A or both from B sum up to any integer from X, it holds that, for each $x_\ell \in X$, there is at least one shift action $f_{a_{i_\ell}}^{x_\ell}$ with effect on a_{i_ℓ} voters who prefer c_l^x, and one shift action $f_{b_{j_\ell}}^{x_\ell}$ with effect on b_{j_ℓ} voters who prefer c_ℓ^x. Since there are t candidates c_ℓ^x and since $|F'| = 2t$, it follows that there are exactly two shift actions with effect on some voters preferring c_ℓ^x. Since c_ℓ^x has to lose at least x_ℓ points, it holds that $a_{i_\ell} + b_{j_\ell} \geq x_\ell$. In fact, by the pigeonhole principle, it holds that $a_{i_\ell} + b_{j_\ell} \geq x_\ell$. Hence, if there is a successful set of $2t$ shift actions, then there is a solution for our RESTRICTED NUMERICAL MATCHING WITH TARGET SUMS instance.

The Borda Rule. For the Borda rule, almost the same reduction works. Specifically, there still exists some integer q for which the set of requirements which were required in the proof for the Plurality rule will now hold for the Borda rule (with respect to a different q).

Importantly, since p is in the second position in the preference profiles of all of the voters, it holds that the score differences, when applying some shift

actions, are similar for the Plurality rule and the Borda rule. Thus, the proof of correctness for the Plurality rule transfers to the Borda rule. □

4.7. Exact Algorithms

In spite of the pessimism looming from the previous section, in this section we show two exact FPT and XP algorithms for $\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY. Then, in Section 4.8, we present several efficient approximation algorithms.

We begin by observing that $\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY can be solved in polynomial time, provided that we assume the budget B to be a constant. The reason is that we need to choose at most B shift actions out of all the available ones, but, clearly, the number of shift actions available is upper-bounded by the input size.

Observation 4.1. *Both Plurality-*COMBINATORIAL SHIFT BRIBERY *and Borda-*COMBINATORIAL SHIFT BRIBERY *are* XP *when parameterized by the budget B.*

If we restrict the instances to contain only bounded-effect shift actions, then $\mathscr{R}$-COMBINATORIAL SHIFT BRIBERY can be solved in polynomial time, provided that the number n of voters is a constant.

Proposition 4.1. *If the maximum effect of every shift action is upper-bounded by some constant, then both Plurality-*COMBINATORIAL SHIFT BRIBERY *and Borda-*COMBINATORIAL SHIFT BRIBERY *are* XP *when parameterized by the number n of voters.*

Proof. Let Γ be the value bounding, component-wise (that is, voter-wise), the effect of each shift action. First, observe that there are at most $(2\Gamma+1)^n$ types of different shift actions. Second, observe that once one knows the budget spent on each type of shift actions, one can easily check whether a corresponding set of shift actions makes p a winner of the election. Thus we use the following algorithm: we try all possibilities of distributing the budget B among the (at most $(2\Gamma+1)^n$) possible types of shift actions and check whether one of them makes p a winner. If so, then we return True; otherwise, we return False. □

Proposition 4.1 holds even if each shift action comes at an individual cost and also each voter has an individual threshold function, since we can, given some budget, always assume that we select the cheapest set of shift actions of a given type. Further, by expressing our problem as an integer linear program (ILP)

and by using a famous result of Lenstra [104], we can strengthen the above XP-membership to FPT-membership, for the case of the Plurality rule.

Theorem 4.9. *For bounded-effect shift actions (where we treat the bound as a constant),* PLURALITY-COMBINATORIAL SHIFT BRIBERY *is fixed-parameter tractable when parameterized by the number n of voters.*

Proof. Given an instance of Plurality-COMBINATORIAL SHIFT BRIBERY with n voters, our algorithm proceeds as follows. First, we guess a subset of the voters for whom we will guarantee that p is ranked first (there are 2^n guesses to try). For each guessed set of voters, we test whether p would be a winner of the election if p was shifted to the top position by the guessed voters and was not ranked first by the remaining voters. For each such guessed subset V' of voters for which this test is positive, we check whether it is possible to ensure (by applying sets of shift actions whose cost do not exceed the budget) that the voters from V' rank p first. We do so as follows.

Let Γ be the constant bounding, component-wise, the effect of each shift action. Observe that there are at most $(2\Gamma+1)^n$ types of different shift actions. For each shift action type z, we introduce a variable x_z denoting the number of times a shift action of type z is present in the solution. For each voter v_i, we denote the position of p in the original preference order of v_i by $s_{v_i}(p)$. For each voter $v_i \in V'$ we add the following constraint:

$$\sum_{\gamma\in[-\Gamma,\Gamma]} \left(\gamma \sum_{\{z:f_z \text{ has an effect of } \gamma \text{ on } v_i\}} x_z\right) \geq s_{v_i}(p).$$

This ensures that p is indeed shifted to the top position in the preference list of v_i. We add the following budget constraint:

$$\sum x_z \leq B,$$

ensuring that the solution respects the budget. Finally, for each shift action type z we add a constraint ensuring that we use at most as many shift actions of type z as there are shift actions of type z available in the input. This finishes the description of the ILP. By a result of Lenstra [104], we can solve this ILP in FPT-time, since we have at most $(2\Gamma+1)^n$ integer variables. □

Roughly speaking, Theorem 4.9 is the reason why (the hardness proven in) Theorem 4.7 does not apply to the Plurality rule; in this setting, Plurality-COMBINATORIAL SHIFT BRIBERY is tractable. Note that Theorem 4.9 applies to the case where each shift action has the same unit cost, that is, the case which

we focus on in this chapter. Nonetheless, it seems possible to lift Theorem 4.9 to the case where each shift action has its individual cost, by applying ideas from Bredereck et al. [30].

4.8. Approximation Algorithms

We now explore the possibility of finding approximate solutions for COMBINATORIAL SHIFT BRIBERY. We focus on approximating the cost of shift actions necessary to ensure p's victory (for example, a 2-approximate algorithm finds a solution that ensures p's victory whenever it is possible, and uses at most twice as many shift actions as necessary). By Theorem 4.4 and Theorem 4.5, we know that we cannot hope to find approximate algorithms for the cases of COMBINATORIAL SHIFT BRIBERY where the shift actions can have negative effects. Thus, in this section, we focus on unit-effect shift actions with only positive effects. This also simplifies our situation in that we can always check if it is possible to ensure p's victory: it suffices to apply all available shift actions and check if p is a winner (indeed, not being able to perform such a check is at the heart of our inapproximability results from Section 4.6).

Essentially, all our approximation algorithms are based on the framework of approximation algorithms for the non-combinatorial variant of SHIFT BRIBERY, as described by Elkind and Faliszewski [61], Elkind et al. [62]: either by directly using their framework (as a "black-box"), or by plugging some algorithms into their framework. We begin by exploring the first possibility and then describe the second.

Theorem 4.10. *If each shift action has effects of either* 0 *or* +1 *on each voter, then* PLURALITY-COMBINATORIAL SHIFT BRIBERY *can be* Λ*-approximated in polynomial-time and* BORDA-COMBINATORIAL SHIFT BRIBERY *can be* 2Λ*-approximated in polynomial time, where* Λ *denotes the maximum number of voters affected by a shift action.*

Proof. The general idea of these approximation algorithms is to split each shift action that affects some $\Lambda' \leq \Lambda$ voters into Λ' shift actions, each affecting a single voter only. In effect, we construct a non-combinatorial instance of the SHIFT BRIBERY problem that we solve exactly, for the case of Plurality rule, or 2-approximately, for the case of the Borda rule.

Specifically, our construction goes as follows. Let $\lambda(i)$ denote the number of shift actions affecting voter i. Given an instance of COMBINATORIAL SHIFT BRIBERY,

we create an instance of SHIFT BRIBERY that is identical, except that instead of having shift actions, we have price functions for the voters: we set the price function for each voter i in such a way which ensures that, for $j \leq \lambda(i)$, shifting p by j positions costs j, and for $j > \lambda(i)$, shifting p by j positions costs $(2B+1)^j$ (where B is the total number of shift actions available; note that the exponential function $(2B+1)^j$ ensures that the price functions are convex and that we can easily identify situations where one shifts p by more than $\lambda(i)$ positions).[2]

Below we describe how to use this construction for the case of the Plurality rule and for the case of the Borda rule.

The Plurality Rule. We first translate the input instance into the non-combinatorial Plurality-SHIFT BRIBERY instance as described above. Then, we apply the known, exact, polynomial-time algorithm for Plurality-SHIFT BRIBERY [62] on this instance. Let s be the cost of the solution found for the non-combinatorial instance. If $s > B$, then it is impossible to ensure p's victory in the combinatorial instance (since the number of available shift actions is insufficient).

If $s \leq B$, then to obtain a solution F for the Plurality-COMBINATORIAL SHIFT BRIBERY instance we do as follows. For each voter v that in the (non-combinatorial) bribed election ranks p first, we select shift actions in the combinatorial instance so that v ranks p first. Note that $|F| \leq s$ and that F is indeed a (combinatorial) solution.

Towards a contradiction, assume that there is a successful set of shift actions F' with size smaller than $|F|/\Lambda$. It is easy to see, however, that such a set of shift actions would correspond to a bribery of cost smaller than s for the non-combinatorial instance. Since s is the cost of the optimal solution for the non-combinatorial instance, this is a contradiction.

The Borda Rule. The case of Borda-COMBINATORIAL SHIFT BRIBERY follows analogously, but instead of using the polynomial-time exact algorithm for the non-combinatorial instance, we use the 2-approximation algorithm for Borda-SHIFT BRIBERY [61, 62]. Let s be the cost of the solution found. If $s > 2B$, then it is impossible to ensure p's victory.

Otherwise, to obtain a solution F for the combinatorial instance, for each vote v where the non-combinatorial solution shifts p by some t positions, we include t

[2] Strictly speaking, there is no need to ensure that the price functions are convex, but this is the variant of SHIFT BRIBERY that we generalize in this chapter, so we consider this variant, for consistency.

shift actions that affect this voter. Clearly, $|F| \leq s$ and F is a correct solution for the combinatorial instance.

If there exists a solution F' for the combinatorial instance that used less than $|F|/(2 \cdot \Lambda)$ shift functions, then there would be a solution for the non-combinatorial instance with costs smaller than $|F|/2 \leq s/2$. Since we used a 2-approximate algorithm for the non-combinatorial instance, this is impossible. □

We mention that it might be possible to improve the approximation ratio given in Theorem 4.10, at least for the Borda rule. The idea might be to cast the problem as a variant of the SET MULTICOVER problem, which is a generalization of the SET COVER problem where each element has its own covering requirement. Then, one could use an approximation algorithm for the SET MULTICOVER problem (for example, the one suggested by Rajagopalan and Vazirani [132]) and plug it into the 2-approximation algorithm of Elkind and Faliszewski [61].

We can achieve better approximation guarantees for the Borda rule, when we further restrict the allowed shift actions. To obtain these results, we again use the framework of Elkind and Faliszewski [61], albeit in a different way. In essence, they have shown the following: if, for a given variant of SHIFT BRIBERY, either for the Plurality rule or for the Borda rule, one can provide a function that computes how to obtain the highest number of points for the preferred candidate given some budget B, then there is a 2-approximation algorithm for this variant of SHIFT BRIBERY.[3] Note that such an algorithm does not solve the SHIFT BRIBERY problem. While it maximizes the score of p, it does not ensure that no candidate receives higher score. Indeed, an optimal solution might increase the score of p to a smaller extent, but at the expense of more dangerous opponents.

Theorem 4.11. *Borda-*COMBINATORIAL SHIFT BRIBERY *is* 2*-approximable in polynomial time when we allow only* $(+1, +1)$*-shift actions.*

Proof. By the discussion preceding the theorem statement, it suffices to provide a function that, given an instance of COMBINATORIAL SHIFT BRIBERY with budget B, finds a set of shift actions that obtains the highest possible number of points for the preferred candidate p without exceeding the budget.

The general idea of achieving this is to compute a maximum b-matching in an auxiliary multigraph (multigraphs allow multiple edges between the vertices). A b-matching of a multigraph G with respect to the function $b : V(G) \to \mathbb{N}$ (called

[3] In fact, their result applies to all scoring rules, but in this chapter we only focus on the Plurality rule and on the Borda rule.

a *covering function*) is an edge-induced subgraph of G such that each vertex u has degree at most $b(u)$. It is known that a b-matching can be computed in polynomial time [83].

We construct the auxiliary multigraph G as follows. For each voter v_i we create a vertex u_i. For each shift action with effect 1 on voter v_i and effect 1 on voter v_j, we create an edge $\{u_i, u_j\}$. Then, we define a covering function b such that $b(u_i)$ is the number of positions that p can be shifted forward in the preference order of voter v_i (that is, the position of p in the preference order of voter v_i).

If G has a b-matching of size at least B, then it corresponds to a set of shift actions that increase the score of p by $2B$, which is the highest gain possible. If G has a b-matching of size $k < B$, then we take the shift actions corresponding to the edges of this b-matching (these shift actions maximize the number of points that p can gain from shift actions that move p within two votes) and greedily select more shift actions, such that each of them pushes p forward in one vote, to use up the budget (at this point, every shift action can affect p in a single vote only). Thus, for a given budget, our function computes the highest point gain possible for p. □

Next, we consider interval shift actions. That is, we fix some order of the voters and restrict each shift action to have effect only on voters which comprise intervals. Importantly, we also allow "holes" inside these intervals. Unfortunately, the algorithm requires XP-time for the parameterization by the length of the longest interval.

Theorem 4.12. *Both for the Plurality rule and for the Borda rule,* COMBINATORIAL SHIFT BRIBERY *can be* 2*-approximated in* XP*-time when we allow only interval shift actions, provided that we take Λ, the upper bound on the number of voters affected by each shift action, as the parameter.*

Proof. As per the discussion preceding Theorem 4.11, it suffices to describe how to find a set of shift actions which, under a given budget, maximizes the number of points that the preferred candidate p gains.

To this end, we use a dynamic programming algorithm. Consider an input for COMBINATORIAL SHIFT BRIBERY with election $E = (C, V)$, preferred candidate p, and budget B. Let $m := |C|$ and let $n := |V|$. We have $V = (v_1, \ldots, v_n)$. Our algorithm uses the following table for partial results. For numbers $x, y, s_0, \ldots, s_{\Lambda-1}$, the table entry:

$$T[x, y, s_0, s_1, \ldots, s_{\Lambda-1}]$$

denotes the maximum number of additional points that candidate p can gain from voters $v_1,\ldots,v_x$ under the condition that (1) exactly y shift actions are used, each of them affecting only voters from the set $\{v_1,\ldots,v_x\}$, and (2) for each $i \in \{0,...,\Lambda-1\}$, candidate p is shifted to position s_i in the preference order of voter v_{x-i}. In other words, we iterate over the voters and store the effect that the applied shift actions had on the last Λ voters. The size of the table is $n \cdot B \cdot m^{\Lambda+1}$.

Our algorithm is almost the same for the Plurality rule and for the Borda rule. The only difference is in computing the scores of the candidates. Let z, $0 \leq z \leq m-1$, denote the position of p in the preference order of some voter (position 0 means that p is ranked first). Then, by score(z) we mean the score that p gains from this voter. Clearly, for the Plurality rule we have score(z) = 1 for $z = 0$ and score(z) = 0 otherwise. For the Borda rule we have score(z_i) = $m - z_i - 1$. For a set of voters and a vector $z_1,\ldots,z_t$, for $t \in [n]$ and where each z_i in $\{0,\ldots,m-1\}$) denotes the positions of p in the preference orders of these voters, we write score($z_1,\ldots,z_t$) to mean the score that p gains from these voters. That is:

$$\text{score}(z_1,\ldots,z_t) = \sum_{i\in[t]} \text{score}(z_i).$$

Given this preparation, we are ready to describe our algorithm (jointly for the Plurality rule and for the Borda rule).

Initialization. We initialize the entries $T[\Lambda, y, s_0, s_1, \ldots, s_{\Lambda-1}]$ of the table as follows. We check whether there is a set of y shift actions that have effects only on voters from $(v_1,\ldots,v_\Lambda)$ and such that applying this set of y shift actions shifts candidate p to positions $s_0,\ldots,s_{\Lambda-1}$ in the preference orders of the voters $v_1,\ldots,v_\Lambda$, respectively. If such set exists, then we set $T[\Lambda, y, s_0, s_1, \ldots, s_{\Lambda-1}]$ to score($s_0, s_1, \ldots, s_{\Lambda-1}$). Otherwise, we set $T[\Lambda, y, s_0, s_1, \ldots, s_{\Lambda-1}]$ to $-\infty$. (We explain how to check if such a set of shift actions exists at the end of the proof.)

Recursion Step. To compute the table entries $T[x, y, s_0, s_1, \ldots, s_{\Lambda-1}]$ for $x > \Lambda$, one has to find those subsets of i shift actions (for $i \in [y]$) whose last affected voter is v_x, that ensure—together with $y - i$ shift actions whose last affected voter is from the set $\{v_1,\ldots,v_{x-1}\}$—that, for each j, $0 \leq j \leq \Lambda - 1$, it holds that p is shifted to position s_j in the preference order of v_{x-j}.

More specifically, in the update phase we compute, for each x, $\Lambda < x \leq n$, each y, $0 \leq y \leq B$, and each vector $(s_0,\ldots,s_{\Lambda-1}) \in \{0,\ldots,m-1\}^\Lambda$, the table entry $T[x, y, s_0, s_1, \ldots, s_{\Lambda-1}]$ as follows. We say that a vector $(\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_{\Lambda-1}) \in \{0,\ldots,m\}^\Lambda$

is (x,i)-*realizable* for some i $(0 \le i \le y)$, if there is a set of i shift actions whose last affected voter is v_x and such that for each j, $0 \le j \le \Lambda - 1$, it shifts candidate p by $\hat{s}_j$ positions in the preference order of voter v_{x-j}. We write $R(x,i)$ to denote the set of vectors from $\{0,\ldots,m-1\}^\Lambda$ that are (x,i)-realizable (we describe how to compute $R(x,i)$ later). Then, we compute $T[x,y,s_0,s_1,\ldots,s_{\Lambda-1}]$ as follows:

$$T[x,y,s_0,s_1,\ldots,s_{\Lambda-1}] = \max\{T[x-1,y-i,s^*,s_0-\hat{s}_1,\ldots,s_{\Lambda-1}-\hat{s}_{\Lambda-1},s^*]$$

$$+\,\mathrm{score}(s_0,s_1,\ldots,s_{\Lambda-1}) - \mathrm{score}(s_1-\hat{s}_1,\ldots,s_{\Lambda-1}-\hat{s}_{\Lambda-1}) \;:$$

$$0 \le i \le y, 0 \le s^* \le m-1, (s_0,\hat{s}_1,\ldots,\hat{s}_{\Lambda-1}) \in R(x,i)\}$$

Informally, for each "realizable total effect" of i shift actions whose last affected voter is v_x, the number of points that candidate p gains is the number of additional points that candidate p gains by shift actions for which the last affected voter is from $(v_1,\ldots,v_{x-1})$ plus the number of additional points that candidate p gains by shift actions for which the last affected voter is v_X (to avoid double counting, this is expressed as the difference in the middle line of the above formula).

We next show how to compute $R(x,i)$. We try every vector $(\hat{s}_0,\ldots,\hat{s}_{\Lambda-1}) \in \{0,\ldots,m-1\}^\Lambda$, and for each such vector, we check whether it is (x,i)-realizable. Perhaps the easiest way to do this is to formulate this problem as an integer linear program (ILP) with a constant number of variables, as we describe next.

Let $(\hat{s}_0,\ldots,\hat{s}_{\Lambda-1})$ be a vector for which we want to check whether it is (x,i)-realizable. For each subset $Q \subseteq \{0,\ldots,\Lambda-1\}$, we say that a shift action is of type Q if it affects exactly the voters v_{x-i} with $i \in Q$. For each such subset Q, we introduce an integer variable x_Q, denoting the number of shift actions of type Q used in the (x,i)-realization of our vector. We solve the following ILP:

$$\sum_{Q \subseteq \{0,\ldots,\Lambda-1\}} x_Q = i \tag{4.1}$$

$$\sum_{Q \subseteq \{1,\ldots,\Lambda-1\}} x_{Q\cup\{0\}} = i \tag{4.2}$$

$$\sum_{j \in Q} x_Q = \hat{s}_j \qquad \forall j : 0 \le j \le \Lambda - 1 \tag{4.3}$$

(Note that the middle constraint ensures that the last affected voter is v_x.) Since the number of variables in this ILP is 2^Λ, it follows, from a result of Lenstra

[104], that this ILP can be solved in XP-time with respect to the parameter Λ (indeed, even in FPT-time). Note that, using the same ILP but without the middle constraint, we can check which vectors $(s_0, \ldots, s_{\Lambda-1})$ we can use in the initialization step.

Coming back to our dynamic program, it is clear that finding how to obtain the maximum score for p while respecting our budget can be found by taking the maximum over the table entries $T[n, B', s_0, s_1, \ldots, s_{\Lambda-1}]$, for all possible values of B', $0 \leq B' \leq B$, and $(s_0, s_1, \ldots, s_{\Lambda-1}) \in \{0, \ldots, m-1\}^{\Lambda}$. □

While in this section we showed that it is indeed possible to achieve some approximation algorithms for some special cases of the COMBINATORIAL SHIFT BRIBERY problem, the settings for which our algorithms are efficient are quite restrictive. This means that in practice one might want to seek good heuristics and use our algorithms as a guidance for the initial search.

4.9. Outlook

We next state some of the research directions motivated by the results described in this chapter.

- One immediate question is whether COMBINATORIAL SHIFT BRIBERY, both for the Plurality rule and for the Borda rule, can be solved in polynomial-time for $(+1, +1)$-shift actions or interval actions, under the assumption that the number of candidates is a constant.

- We proved approximation guarantees for our approximation algorithms. It might be interesting to perform experiments in order to check whether, in practice, our approximation algorithms find better solutions than what we could theoretically prove.

- Our results suggest studying further restrictions of the COMBINATORIAL SHIFT BRIBERY problem. As an example, since parameterizing by the number of available shift actions immediately gives fixed-parameter tractability results, a natural question is whether other natural parameterizations exist which could also lead to positive results.

- Naturally, one might consider other voting rules as well. Most interesting are Condorcet-consistent rules (which always select the Condorcet winner, if it exists), such as the Copeland rule, since these rules tend to behave

rather differently than scoring rules (both the Plurality rule and the Borda rule are scoring rules): for example, different behavior between scoring rules and non-scoring rules is apparent in Chapter 3.

- It might also be interesting to consider domain restrictions regarding the preferences of the voters (for example, single-crossing seems particularly natural in the context of interval shift actions, since it means that each shift action affects voters with somewhat similar preferences), as it is well-demonstrated that restricting the domain of the voters can lead to tractability (see, for example, [33, Theorem 10]). Pursuing this direction, however, would require a careful discussion of which shift actions can be applied: for example, it is not clear whether we should require single-crossingness also from the bribed election.

Appendix for Chapter 4

We provide proofs missing from Chapter 4.

4.A. Proof of Theorem 4.5

Theorem 4.5. *Assuming* $\mathsf{P} \neq \mathsf{NP}$, COMBINATORIAL SHIFT BRIBERY *is inapproximable (in polynomial time) both for the Plurality rule and for the Borda rule, even for elections with only two candidates and when allowing only* $(+1,-1)$*-shift actions.*

Proof. We provide a reduction from the W[2]-complete SET COVER problem parameterized by the solution size [57].

> SET COVER
> **Input:** A universe of elements X, a collection $\mathscr{S}$ of sets of elements of X, and a budget h.
> **Question:** Is there a subcollection $\mathscr{S}' \subseteq \mathscr{S}$ of sets such that $|\mathscr{S}'| \leq h$ and $\bigcup_{S \in \mathscr{S}'} S = X$?

Let $(\mathscr{S}, X, h)$ be a SET COVER instance. We assume, without loss of generality, that every element belongs to at least one set. We construct an instance of Plurality-COMBINATORIAL SHIFT BRIBERY. We set the budget B to be $|X|$. The candidate set is $\{p, d\}$, where p is the preferred candidate. We have an *element voter* v_i for each element x_i, with preference order $d \succ p$. We have a *set voter* v_j^S for each set S_j, with preference order $p \succ d$. We also have $|X| + |\mathscr{S}| - 2h - 1$ *dummy voters*, each with preference order $d \succ p$. For each element x_i and each set S_j, if $x_i \in S_j$, then we construct a shift action f_j^i with effect of $+1$ on v_i and effect of -1 on v_j^S. This completes the construction. It is easy to see that the reduction can be computed in polynomial time.

Next, we show that there is a successful set of shift actions (note that the size of this set is not important, that is, we allow infinite budget) if and only if there is a set cover of size h.

For the "if" direction, assume that there is a set cover $\mathscr{S}'$ of size at most h. We show how to build a successful set of shift actions. We start with $F' = \emptyset$ and, for each element x_i, we choose an arbitrary set $S_j \in \mathscr{S}'$ which contains x_i, and add the corresponding function f_j^i to F'. After applying F', observe that p becomes a winner: all $|X|$ element voters and $|\mathscr{S}| - h$ set voters prefer p but only $|X| + |\mathscr{S}| - 2h - 1$ dummy voters and h set voters prefer d.

For the "only if" direction, assume that there is a successful set of shift actions $F' \subseteq F$. Let h' be the number such that, after applying the shift actions from F', p is preferred by exactly $|S| - h'$ set voters (that is, shift actions in F' correspond to h' sets from $\mathscr{S}$). For p to be a winner, a majority of the voters (that is, at least $|X| + |\mathscr{S}| - h$ voters) must prefer p. Thus, after applying F', at least $X - (h - h')$ element voters prefer p. This means that there is a collection of h' sets from $\mathscr{S}$ that jointly cover at least $|X| - (h - h')$ elements. Since every element belongs to some set, we can extend this collection to a set cover by adding at most $h - h'$ sets (in the worst case, one set for each uncovered element). This proves that there is a set cover for $(\mathscr{S}, X, h)$.

Note that in the above argumentation we made no assumptions regarding the size of F'. Hence, finding any solution for our Plurality-COMBINATORIAL SHIFT BRIBERY instance, including approximate solutions for any approximation factor, implies finding a set cover of size at most h. This means that, unless P = NP, Plurality-COMBINATORIAL SHIFT BRIBERY is inapproximable in polynomial time. □

4.B. Remaining Cases from the Proof of Theorem 4.6

Theorem 4.6. *Both for the Plurality rule and for the Borda rule,* COMBINATORIAL SHIFT BRIBERY *is* W[1]*-hard for the combined parameter* (m, B)*, even if we either only allow* $(+1, -1)$*-shift actions or only allow* $(+1, +1)$*-shift actions.*

The Borda Rule with $(+1, +1)$-Shift Actions. We can slightly modify the reduction used for the Plurality rule with $(+1, +1)$-shift actions. Specifically, we describe a parameterized reduction from the W[1]-hard CLIQUE problem, parameterized by the solution size, to Borda-COMBINATORIAL SHIFT BRIBERY with $(+1, +1)$-shift actions, parameterized by (m, B) .

Let (G,h) be an instance of CLIQUE with $V(G) = \{u_1,\dots,u_{n'}\}$ and $E(G) = \{e_1,\dots,e_{m'}\}$. We create an instance of Borda-COMBINATORIAL SHIFT BRIBERY as follows. The set of candidates is $\{p\} \cup D$, where $D = \{d_1,\dots,d_{h-1}\}$. We create the following voters.

1) For each vertex $u_i \in V(G)$, we create a corresponding *vertex voter* v_i with preference order:
$$d_1 \succ \cdots \succ d_{h-1} \succ p.$$

2) We create $n' - 2h$ *dummy voters*, each with preference order:
$$p \succ d_2 \succ \cdots \succ d_{h-1} \succ d_1.$$

3) We create h *dummy voters*, each with preference order:
$$d_{h-1} \succ p \succ d_2 \succ \cdots \succ d_{h-2} \succ d_1.$$

4) We create $n' - h$ *dummy voters*, each with preference order:
$$p \succ d_1 \succ \cdots \succ d_{h-1}.$$

5) We create $n' - h$ *dummy voters*, each with preference order:
$$d_1 \succ p \succ d_2 \succ \cdots \succ d_{h-1}.$$

For each edge $\{u_i,u_j\} \in E(G)$, we create a shift action $f_{\{u_i,u_j\}}$ with effect 1 on the vertex voters v_i and v_j and effect 0 on all other voters. Finally, we set the budget to $B := \binom{h}{2}$. This completes the construction, which is computable in polynomial time.

The proof of correctness follows the same lines as the proof for the corresponding proof for the Plurality rule with $(+1,+1)$-shift actions, but instead of counting the number of approvals, we compute the Borda scores of the candidates. Indeed, this is the reason for our additional dummy voters.

In particular, the construction ensures that d_1 is the original winner of the election and that the difference between the Borda score of p and the Borda score of d_1 is exactly h^2. Furthermore, each shift action can increase the score of p by at most two. Hence, to make p a co-winner one must increase the score of p by $h(h-1)$ and decrease the score of d_1 by h. This is possible if and only if the shift actions correspond to the edges of some clique of size h.

The Plurality Rule with $(+1,-1)$-Shift Actions. We still reduce from the W[1]-hard CLIQUE problem, parameterized by the solution size, but the reduction is a bit more involved.

Let (G,h) be a CLIQUE instance where the graph G has $n' := |V(G)|$ vertices and $m' := |E(G)|$ edges. We construct a Plurality-COMBINATORIAL SHIFT BRIBERY instance as follows. We let the set of candidates be $\{p,d\} \cup D$, where $D := \{d_1,\ldots,d_{h-1}\}$, and we create the following voters:

1) For each vertex v_i, we create $\binom{h}{3}$ *vertex voters* $v_i^1,\ldots,v_i^{\binom{h}{3}}$ corresponding to v_i, each with preference order:

$$d_1 \succ \cdots \succ d_{h-1} \succ p.$$

2) For each edge $e_j = \{v_{i_1}, v_{i_2}\}$, we create a corresponding *edge voter* w_j with preference order:

$$p \succ d_1 \succ \cdots \succ d_{h-1}.$$

3) We create $2\binom{h}{2} + (n' - 2h)\binom{h}{3} - m'$ *dummy voters*, each with preference order:

$$p \succ d_1 \succ \cdots \succ d_{h-1}.$$

For each edge $e_j = \{v_{i_1}, v_{i_2}\}$, we construct $2\binom{h}{3}$ shift actions, denoted by:

$$f^1_{e_j,v_{i_1}},\ldots,f^{\binom{h}{3}}_{e_j,v_{i_1}} \text{ and } f^1_{e_j,v_{i_2}},\ldots,f^{\binom{h}{3}}_{e_j,v_{i_2}},$$

where for each $z \in [\binom{h}{3}]$, we have that (a) $f^z_{e_j,v_{i_1}}$ has effect of $+1$ on $v^z_{i_1}$ and effect of -1 on w_j, and (b) $f^z_{e_j,v_{i_2}}$ has effect of $+1$ on $v^z_{i_2}$ and effect of -1 on w_j. Finally, we set the budget $B := 2\binom{h}{2}\binom{h}{3}$. This completes the construction. It is easy to see that the reduction can be computed in polynomial time.

Observe that, initially, the edge voters and the dummy voters prefer p, while the vertex voters prefer d_1. Therefore, the initial score of p is $2\binom{h}{2} + (n' - 2h)\binom{h}{3}$, while the initial score of d_1 is $n'\binom{h}{3}$. We can assume, without loss of generality, that it means that d_1 is the winner of the election (instances not satisfying this assumption can be solved in constant time).

It remains to show that our constructed instance contains a successful set of shift actions F' of size at most h if and only if (G,h) contains a clique of size at most B. The general idea is that, if we choose the shift actions corresponding

to the edges connecting the nodes of an h-size clique, then we will ensure that p becomes the preferred candidate for $h\binom{h}{3}$ additional vertex voters, while making d_1 the preferred candidate for only $\binom{h}{2}$ additional edge voters. A more detailed argumentation follows.

For the "if" direction, let $H \subseteq V(G)$ be a set of h vertices which form a clique in G and let $E' \subseteq E(G)$ be the set of edges connecting vertices from H. We choose the following set of shift actions:

$$F' = \{f^z_{e_j,v_{i_1}}, f^z_{e_j,v_{i_2}} : e_j = \{v_{i_1}, v_{i_2}\} \in E', z \in [\binom{h}{3}]\}.$$

We show that F' is a successful set of shift actions. To this end, observe that for each vertex voter v_i^z with $v_i \in V'$ and $z \in [\binom{h}{3}]$, candidate p is shifted $h-1$ positions forward, therefore p becomes the preferred candidate for these voters. This means that $h\binom{h}{3}$ additional vertex voters prefer p (and, thus, do not prefer d_1 anymore). Furthermore, p is shifted backwards only for the voters in $\{w_j : e_j \in E'\}$, that is, d_1 becomes the most preferred candidate for $\binom{h}{2}$ edge voters and p remains the most preferred candidate for $m' - \binom{h}{2}$ edge voters. Thus, p and d tie as winners.

For the "only if" direction, let F' be a successful set of shift actions. To make p a winner of the election, p must be shifted to the top position for at least $h\binom{h}{3} - \binom{h}{2}$ vertex voters (no other type of voters can be affected positively). By the pigeonhole principle, these vertex voters correspond to at least h different vertices (there are $\binom{h}{3}$ voters corresponding to each vertex). In effect, at least $\binom{h}{2}$ edge voters must be effected negatively so that d_1 becomes their most preferred candidate. Thus, to make p win the election, p must be shifted to the top position for at least $h\binom{h}{3}$ vertex voters. This implies that $|F'| \geq (h-1) \cdot h \cdot \binom{h}{3} = 2\binom{h}{2}\binom{h}{3} = B$ and, hence, $|F'| = B$. It follows that p is shifted backwards, making d_1 the most preferred candidate, for exactly $\binom{h}{2}$ edge voters and that p must be shifted to the top position for exactly $h\binom{h}{3}$ vertex voters corresponding to exactly h different vertices. By construction, this implies that these h vertices form a clique, and we are done.

The Borda Rule with $(+1,-1)$-Shift Actions. For the Borda rule, the reduction is, once again, a bit more involved, but the main idea is similar to that for the Plurality rule.

Let (G,h) be an instance of the CLIQUE problem where the graph G has $n' := |V(G)|$ vertices and $m' := |E(G)|$ edges. We construct a Borda-COMBINATORIAL

SHIFT BRIBERY instance as follows. The set of candidates is $\{p,d\} \cup D$, where $D := \{d_1,\dots,d_{h-1}\}$, and we create the following voters:

1) For each vertex v_i, we create $\binom{h}{3}$ *vertex voters*, $v_i^1,\dots,v_i^{\binom{h}{3}}$, corresponding to v_i, each with preference order:

$$D \succ p.$$

2) For each edge $e_j = \{v_{i_1}, v_{i_2}\}$, we create a corresponding *edge voter* w_j with preference order:

$$d_1 \succ \cdots \succ d_{h-2} \succ p \succ d_{h-1}.$$

3) We define a value $L := \dfrac{\binom{h}{2} + (n'\binom{h}{3} + m')(h-1) - (\binom{h}{3}h^2) - m'}{h-1}$. Without loss of generality, we can assume that L is an integer (this requires only simple modifications to the input clique instance). We create L *dummy voters*, each with preference order:

$$p \succ d_{h-1} \succ \cdots \succ d_1.$$

For each edge $e_j = \{v_{i_1}, v_{i_2}\}$, we construct $2\binom{h}{3}$ shift actions, denoted by

$$f^1_{e_j,v_{i_1}},\dots,f^{\binom{h}{3}}_{e_j,v_{i_1}} \text{ and } f^1_{e_j,v_{i_2}},\dots,f^{\binom{h}{3}}_{e_j,v_{i_2}},$$

where, for each $z \in [\binom{h}{3}]$, we have that (a) $f^z_{e_j,v_{i_1}}$ has effect of $+1$ on $v^z_{i_1}$ and effect of -1 on w_j, and (b) $f^z_{e_j,v_{i_2}}$ has effect of $+1$ on $v^z_{i_2}$ and effect of -1 on w_j. Finally, we set the budget $B := 2\binom{h}{2}\binom{h}{3}$. This completes the construction. It is easy to see that the reduction can be computed in polynomial time. The proof of correctness follows the same lines as the proof of correctness for the Plurality rule and, thus, is omitted.

4.C. Proof of Theorem 4.7

Theorem 4.7. *Borda*-COMBINATORIAL SHIFT BRIBERY *is* W[1]*-hard with respect to the number n of voters, even when there are no budget constraints and even if we only allow* $(+1,-1)$*-shift actions.*

Proof. We reduce from the the following W[1]-hard problem [114, Lemma 3.2].

STRONGLY REGULAR MULTICOLORED CLIQUE
Input: Two integers, d and h, and an undirected graph $G = (V, E)$, where each vertex has one of h colors in $[h]$, and where each vertex is adjacent to exactly d vertices of each color different from its own.
Question: Does there exist a clique of size h containing one vertex from each color class?

Given an instance of STRONGLY REGULAR MULTICOLORED CLIQUE, we construct an instance of COMBINATORIAL SHIFT BRIBERY, for the Borda rule. The general idea of the reduction is as follows. The set of "important" candidates consists of our preferred candidate p and the candidates that correspond to the edges. For technical reasons, for each edge $e = \{v, v'\}$, we introduce two candidates, e^1 and e^2; one of them is associated with "touching" the vertex v while the other is associated with "touching" the vertex v'. (In fact, we will introduce more edge candidates and some vertex candidates, but we will use them only to ensure correct structure of the election and appropriate bribery behavior.) We build two groups of voters, the vertex-selecting voters and the edge-electing voters. The first group implements picking vertices for the clique (one vertex from each color), while the second group implements picking edges (one edge for each pair of colors). We ensure that, for any successful set of shift actions has any chance of being successful, it must hold that h vertices and $\binom{h}{2}$ edges are picked. Importantly, this property holds even in the unbribed election.

We make sure that p wins the election if and only if the picked voters and the picked edges correspond to a clique (with vertices of each color). To this end, we define the voters so that there are two numbers, α and β, such that:

1) There are h vertices picked by the vertex-selecting voters, each of a different color. The vertex-selecting voters give α points to each edge candidate that is associated with touching one of the selected vertices, and $\alpha + 1$ points to all other edge candidates. This means that by picking a vertex we decrease the score of the edge candidates for the edges that touch this vertex.

2) There are $\binom{h}{2}$ edges picked by the edge-selecting voters, one edge for each pair of colors. The edge-selecting voters give $\beta + 1$ points to each edge candidate that corresponds to a picked edge, and β points to all the remaining edge candidates. This means that, by picking an edge, we increase the score of the candidates corresponding to it.

3) Candidate p gets $\alpha + \beta + 1$ points, irrespectively of which shift actions we apply.

Note that, in the unbribed election, every candidate gets at most $\alpha+\beta+2$ points and p always gets $\alpha+\beta+1$ points. Thus, the challenge is to ensure that every candidate gets exactly $\alpha+\beta+1$ points. By the above description, this is possible only if we pick the vertices and the edges that correspond to a clique of size h, consisting of vertices with different colors. Indeed, if we select an edge e that does not touch two selected vertices, then e^1 and e^2 would receive $\beta+1$ points from edge-selecting candidates and at least one of them would receive $\alpha+1$ points from vertex-selecting voters. In effect, p would not be a winner.

Without loss of generality, we assume that the edges and the vertices which are initially selected in the unbribed election do not form a clique (otherwise there would be a trivial solution for the input problem).

Construction. We now formally describe the reduction. Then, we give an example of applying it to a simple instance, and finally we prove the correctness of the reduction. We illustrate some aspects of the correctness proof using our example.

Candidates. Our set of candidates is somewhat involved. Our important candidates are the preferred candidate p and the sets of edge candidates, E_1 and E_2, defined below.

Let $E(G)=\{e_1,\ldots,e_\mu\}$ be the set of edges of graph G. We create two edge-candidate sets: $E^1=\{e_1^1,\ldots,e_\mu^1\}$ and $E^2=\{e_1^2,\ldots,e_\mu^2\}$. For each $i\in[h]$, let n_i be the number of vertices in G with color i and let $V^i=\{v_1^i,\ldots,v_{n_i}^i\}$ be the set of these vertices. For each color i and each vertex $v_j^i\in V^i$, we define the neighborhood of v_j^i as follows:

$$N(v_j^i) := \{e_\ell^1 : e_\ell=\{v_j^i,v_{j'}^{i'}\}\in E\wedge i<i'\}$$

$$\cup\,\{e_\ell^2 : e_\ell=\{v_j^i,v_{j'}^{i'}\}\in E\wedge i>i'\}.$$

(This, perhaps a bit strange way of using color numbers to pick edge candidates either from E^1 or E^2, is implementing the fact that for each edge $e\in E(G)$ we have two candidates, e^1 and e^2, associated with touching different endpoints of e.)

For technical reasons we need further candidates as follows. To adjust the scores of all other candidates, we introduce a single dummy candidate d. We create two further candidate sets, $E^0=\{e_1^0,\ldots,e_\mu^0\}$ and $E^3=\{e_1^3,\ldots,e_\mu^3\}$, which will act as "guards" for the edge-selecting voters. For each V^i we create two

candidate sets, $U^i := \{u^i_j : v^i_j \in V^i\}$ and $U'^i = \{u'^i_j : v^i_j \in V^i\}$, with $U := \bigcup_{1\le i\le h} U^i$ and $U' := \bigcup_{1\le i\le h} U'^i$, which will act as "guards" for the vertex-selecting voters.

Thus, our set of candidates is: $C := U \cup U' \cup E^0 \cup E^1 \cup E^2 \cup E^3 \cup \{p, d\}$.

Vertex-Selecting Voters. We now describe the vertex-selecting voters. For each color i and each vertex v^i_j, we define the following parts of preference orders (for $j = 1$, we assume that u^i_{j-1} and u'^i_{j-1} are $u^i_{n_i}$ and $u'^i_{n_i}$ respectively):

$$A(v^i_j) : u^i_j \succ \overrightarrow{N(v^i_j)} \succ u'^i_j,$$

$$B(v^i_j) : u^i_{j-1} \succ \overrightarrow{N(v^i_j)} \succ u'^i_{j-1}.$$

For each color i we create three pairs of voters. The voters in the first pair, w_i and w'_i, have the following preference orders:

$$w_i : p \succ A(v^i_1) \succ A(v^i_2) \succ A(v^i_3) \succ \cdots \succ A(v^i_{n_i}) \succ \overrightarrow{R^i},$$

$$w'_i : \overrightarrow{R^i} \succ B(v^i_1) \succ B(v^i_{n_i}) \succ B(v^i_{n_i-1}) \succ \cdots \succ B(v^i_2) \succ p,$$

where R^i is the set of the remaining candidates, that is, $R^i := C \setminus (\{p\} \cup U^i \cup U'^i \cup N(v^i_1) \cup \cdots \cup N(v^i_{n_i}))$. The voters in the second pair, q_i and q'_i, have preference orders that are the reverse of w_i and the reverse of w'_i, respectively. Finally, the voters in the last pair, $\bar{q}_i$ and $\bar{q}'_i$, have preference orders:

$$\bar{q}_i : \overrightarrow{C \setminus (\{d\} \cup N(v^i_1))} \succ d \succ \overrightarrow{N(v^i_1)},$$

$$\bar{q}'_i : \overleftarrow{N(v^i_1)} \succ \overleftarrow{C \setminus (\{d\} \cup N(v^i_1))} \succ d.$$

In effect, the first two pairs of voters jointly give $2(|C|-1)$ points to each of the candidates. The last pair gives $|C|-1$ points to the candidates in $N(v^i_1)$ and $|C|$ points to all other candidates (except d, who receives less than $|C|-1$ points).

Let $\alpha := h(2(|C|-1)+|C|)-1$. Altogether, the vertex-selecting voters give the following scores to the candidates: the candidates in $N(v^1_1) \cup N(v^2_1) \cup \cdots \cup N(v^h_1)$ receive α points, wile all other candidates, except d, receive $\alpha+1$ points (d receives less than α points). Thus, in the unbribed election, $v^1_1, \ldots, v^h_1$ are the selected vertices.

For each color i, we introduce $(n_i-1)\cdot((h-1)\cdot d+2)$ shift actions with effect -1 on voter w_i and effect $+1$ on voter w_i'. To understand where this number comes from, we note that: (1) for each vertex v_j^i, we have $|N(v_j^i)| = (h-1)\cdot d$ (each vertex is connected with d vertices of each color different than its own), (2) in $A(v_j^i)$ and in $B(v_j^i)$, the candidates from $N(v_j^i)$ are surrounded by two vertex candidates, and (3) for an integer t, $1 \le t \le n_i - 1$, applying $t((h-1)\cdot d+2)$ of these shift actions has the effect that the candidates in $N(v_1^i)$ gain one point (that is, v_1^i ceases to be selected), the candidates in $N(v_{t+1}^i)$ lose one point (that is, v_{t+1}^i becomes selected), and no other candidate changes his score (later we will argue that applying numbers of such shift actions which are not multiples of $((h-1)\cdot d+2)$ cannot ensure p's victory).

Edge-Selecting Voters. For the edge-selecting voters, we need the following additional notation. Let $E_{x,y}$ denote the set of candidates representing edges between vertices of color x and color y, that is,

$$E_{x,y} := \{e_\ell^{q\in\{0,1,2,3\}} : e_\ell = \{v_j^x, v_{j'}^y\} \in E\}.$$

We write $n_{x,y}$ to denote the number of edges between vertices of color x and color y. By $\mathrm{id}_z^{x,y}$ we refer to the index of the zth edge between vertices of color x and y. For example, if e_3, e_7 and e_{57} are the only three edges between vertices of colors 1 and 2, then $n_{1,2}=3$, $\mathrm{id}_1^{1,2}=3$, $\mathrm{id}_2^{1,2}=7$, and $\mathrm{id}_3^{1,2}=57$.

For each pair $\{x,y\}$ of distinct colors and each edge $e_{\mathrm{id}_j^{x,y}}$, we introduce the following parts of preference orders (for $j=n_{x,y}$, we assume that $\mathrm{id}_{j+1}^{x,y}=\mathrm{id}_1^{x,y}$):

$$R(e_{\mathrm{id}_j^{x,y}}): e^0_{\mathrm{id}_j^{x,y}} \succ e^1_{\mathrm{id}_j^{x,y}} \succ e^2_{\mathrm{id}_j^{x,y}} \succ e^3_{\mathrm{id}_j^{x,y}},$$

$$S(e_{\mathrm{id}_j^{x,y}}): e^0_{\mathrm{id}_{j+1}^{x,y}} \succ e^1_{\mathrm{id}_j^{x,y}} \succ e^2_{\mathrm{id}_j^{x,y}} \succ e^3_{\mathrm{id}_{j+1}^{x,y}}.$$

For each pair $\{x,y\}$ of distinct colors we introduce three pairs of voters. The voters in the first pair, $w_{x,y}$ and $w'_{x,y}$, have the following preference orders (where $R^{x,y}$ is the set of the remaining candidates, that is, $R^{x,y} := C \setminus (\{p\} \cup E_{x,y})$):

$$w_{x,y}: R(e_{\mathrm{id}_1^{x,y}}) \succ p \succ R(e_{\mathrm{id}_2^{x,y}}) \succ R(e_{\mathrm{id}_3^{x,y}}) \succ \cdots \succ R(e_{\mathrm{id}_{n_{x,y}}^{x,y}}) \succ \overrightarrow{R^{x,y}},$$

$$w'_{x,y}: \overrightarrow{R^{x,y}} \succ S(e_{\mathrm{id}_{n_{x,y}}^{x,y}}) \succ S(e_{\mathrm{id}_{n_{x,y}-1}^{x,y}}) \succ \cdots \succ S(e_{\mathrm{id}_2^{x,y}}) \succ S(e_{\mathrm{id}_1^{x,y}}) \succ p,$$

The voters in the second pair, $q_{x,y}$ and $q'_{x,y}$, have preference orders that are the reverse of $w_{x,y}$ and the reverse of $w'_{x,y}$, respectively. Finally, the voters in the last pair, $\bar{q}_{x,y}$ and $\bar{q}'_{x,y}$, have the following preference orders:

$$\bar{q}_{x,y} : e^1_{\mathrm{id}_1^{x,y}} \succ e^2_{\mathrm{id}_1^{x,y}} \succ d \succ \overrightarrow{C \setminus (\{d, e^1_{\mathrm{id}_1^{x,y}}, e^2_{\mathrm{id}_1^{x,y}}\})},$$

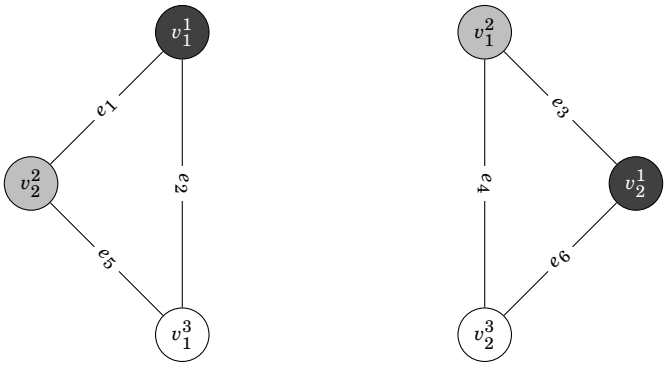
$$\bar{q}'_{x,y} : \overleftarrow{C \setminus (\{d, e^1_{\mathrm{id}_1^{x,y}}, e^2_{\mathrm{id}_1^{x,y}}\})} \succ e^2_{\mathrm{id}_1^{x,y}} \succ e^1_{\mathrm{id}_1^{x,y}} \succ d.$$

The first two pairs of voters jointly give $2(|C|-1)$ points to each of the candidates. The last pair gives $|C|$ points to both $e^1_{\mathrm{id}_1^{x,y}}$ and $e^2_{\mathrm{id}_1^{x,y}}$, and $|C|-1$ points to all other candidates (except d, who receives less than $|C|-1$ points).

Let $\beta := 3\binom{h}{2}(|C|-1)$. Altogether, for each pair of distinct colors $\{x,y\}$, the edge-selecting voters give $\beta+1$ points to candidates $e^1_{\mathrm{id}_1^{x,y}}$ and $e^2_{\mathrm{id}_1^{x,y}}$. All other candidates receive β points (except for d, who receives less than β points). Thus, in the unbribed election, the selected edges are exactly "the first edges between each pair of colors" (that is, edges of the form $e_{\mathrm{id}_1^{x,y}}$, for each pair of distinct colors $\{x,y\}$).

For each pair $\{x,y\}$ of distinct colors, we create $4(n_{x,y}-1)$ shift actions with effect -1 on voter $w_{x,y}$ and effect $+1$ on voter $w'_{x,y}$. The intuition behind these shift actions is similar to the case of vertex-selecting voters. We make the following observations: (1) for each edge $e_{\mathrm{id}_\ell^{x,y}}$, there are four candidates listed in $R(e_{\mathrm{id}_\ell^{x,y}})$ and four candidates listed in $S(e_{\mathrm{id}_\ell^{x,y}})$, and (2) for an integer t, $1 \le t \le n_{x,y}-1$, if we apply $4t$ such shift actions, then the candidates $e^1_{\mathrm{id}_1^{x,y}}$ and $e^2_{\mathrm{id}_1^{x,y}}$ lose one point (edge $e_{\mathrm{id}_1^{x,y}}$ ceases to be selected), the candidates $e^1_{\mathrm{id}_{t+1}^{x,y}}$ and $e^2_{\mathrm{id}_{t+1}^{x,y}}$ gain one point (edge $e_{\mathrm{id}_{t+1}^{x,y}}$ becomes selected), and the scores of all other candidates remain unchanged (we will later argue that, if we apply a number of shift actions that is not a multiple of 4, then p certainly is not a winner of the resulting election).

To conclude the construction, we set the budget $B := \infty$ (that is, we can use as many shift actions as we like). It is easy to verify that the reduction is computable in polynomial time and that it is a parameterized reduction, since the number of voters in the resulting election is upper-bounded by a function dependent only on h. Before proving the correctness of the construction, we consider the following example (we will refer to it during the correctness proof as well).



$V^1 = \{v_1^1, v_2^1\}$, $V^2 = \{v_1^2, v_2^2\}$, $V^3 = \{v_1^3, v_2^3\}$, $h = 3$, $d = 1$

Figure 4.7. A 3-colored graph with six vertices where each vertex is adjacent to one vertex from each of the color classes V^1, V^2, and V^3 (other than its own color class).

Example 10. Consider the STRONGLY REGULAR MULTICOLORED CLIQUE instance (d, h, G) with $d = 1$, $h = 3$, and the graph G from Figure 4.7. Our construction produces the following set of candidates:

$$C := U \cup U' \cup E^0 \cup E^1 \cup E^2 \cup E^3 \cup \{p, d\},$$

with

$$U = \{u_1^1, u_2^1, u_1^2, u_2^2, u_1^3, u_2^3\},\ U' = \{u_1'^1, u_2'^1, u_1'^2, u_2'^2, u_1'^3, u_2'^3\},$$

and

$$E^i = \{e_1^i, e_2^i, \ldots, e_6^i\}, 0 \leq i \leq 3.$$

Furthermore, we have:

$$N(v_1^1) := \{e_1^1, e_2^1\}, \qquad N(v_2^1) := \{e_3^1, e_6^1\},$$

$$N(v_1^2) := \{e_3^2, e_4^1\}, \qquad N(v_2^2) := \{e_1^2, e_5^1\},$$

$$N(v_1^3) := \{e_2^2, e_5^2\}, \qquad N(v_2^3) := \{e_4^2, e_6^2\}.$$

For the vertex-selecting group of voters, we create the following voters. For each color i, we create two voters w_i and w'_i:

$$w_1 : p \succ u_1^1 \succ e_1^1 \succ e_2^1 \succ u_1'^1 \succ u_2^1 \succ e_3^1 \succ e_6^1 \succ u_2'^1 \succ \overrightarrow{R^1},$$

$$w'_1 : \overrightarrow{R^1} \succ u_2^1 \succ e_1^1 \succ e_2^1 \succ u_2'^1 \succ u_1^1 \succ e_3^1 \succ e_6^1 \succ u_1'^1 \succ p,$$

$$w_2 : p \succ u_1^2 \succ e_3^2 \succ e_4^1 \succ u_1'^2 \succ u_2^2 \succ e_1^2 \succ e_5^1 \succ u_2'^2 \succ \overrightarrow{R^2},$$

$$w'_2 : \overrightarrow{R^2} \succ u_2^2 \succ e_3^2 \succ e_4^1 \succ u_2'^2 \succ u_1^2 \succ e_1^2 \succ e_5^1 \succ u_1'^2 \succ p,$$

$$w_3 : p \succ u_1^3 \succ e_2^2 \succ e_5^2 \succ u_1'^3 \succ u_2^3 \succ e_4^2 \succ e_6^2 \succ u_2'^3 \succ \overrightarrow{R^3},$$

$$w'_3 : \overrightarrow{R^3} \succ u_2^3 \succ e_2^2 \succ e_5^2 \succ u_2'^3 \succ u_1^3 \succ e_4^2 \succ e_6^2 \succ u_1'^3 \succ p,$$

with $R^i := C \setminus (\{p\} \cup U^i \cup U'^i \cup N(v_1^i) \cup \cdots \cup N(v_{n_i}^i))$, $1 \le i \le 3$. For each of these voters we add a voter with reversed preferences. (This means that, so far, all candidates get the same total score.) We finish this group of voters by creating, for each color i, two voters, $\bar{q}_i$ and $\bar{q}'_i$, with preference orders:

$$\bar{q}_i : \overrightarrow{C \setminus (\{d\} \cup N(v_1^i))} \succ d \succ \overrightarrow{N(v_1^i)},$$

$$\bar{q}'_i : \overleftarrow{N(v_1^i)} \succ \overleftarrow{C \setminus (\{d\} \cup N(v_1^i))} \succ d.$$

This ensures that, for each color i, all the candidates in $N(v_1^i)$ get α points, and all other candidates get $\alpha + 1$ points (except d who gets less points). We create 4 shift actions with effect -1 on voter w_i and effect $+1$ on voter w'_i.

For the second group of edge-selecting voters, recall that $E_{x,y}$ denotes the set of candidates representing edges between vertices of color x and color y. Specifically, we have:

$$E_{1,2} := \{e_1^0, e_1^1, e_1^2, e_1^3 e_3^0, e_3^1, e_3^2, e_3^3\},$$

$$E_{1,3} := \{e_2^0, e_2^1, e_2^2, e_2^3 e_6^0, e_6^1, e_6^2, e_6^3\},$$

$$E_{2,3} := \{e_4^0, e_4^1, e_4^2, e_4^3 e_5^0, e_5^1, e_5^2, e_5^3\}.$$

For each pair $\{x,y\}$ of distinct colors we create two voters, $w_{x,y}$ and $w'_{x,y}$, as follows:

$$w_{1,2} : e_1^0 \succ e_1^1 \succ e_1^2 \succ e_1^3 \succ p \succ e_3^0 \succ e_3^1 \succ e_3^2 \succ e_3^3 \succ \overrightarrow{R^{1,2}},$$

$$w'_{1,2} : \overrightarrow{R^{1,2}} \succ e_1^0 \succ e_3^1 \succ e_3^2 \succ e_3^3 \succ e_3^0 \succ e_1^1 \succ e_1^2 \succ e_1^3 \succ p,$$

$$w_{1,3} : e_2^0 \succ e_2^1 \succ e_2^2 \succ e_2^3 \succ p \succ e_6^0 \succ e_6^1 \succ e_6^2 \succ e_6^3 \succ \overrightarrow{R^{1,3}},$$

$$w'_{1,3} : \overrightarrow{R^{1,3}} \succ e_2^0 \succ e_6^1 \succ e_6^2 \succ e_2^3 \succ e_6^0 \succ e_2^1 \succ e_2^2 \succ e_6^3 \succ p,$$

$$w_{2,3} : e_4^0 \succ e_4^1 \succ e_4^2 \succ e_4^3 \succ p \succ e_5^0 \succ e_5^1 \succ e_5^2 \succ e_5^3 \succ \overrightarrow{R^{2,3}},$$

$$w'_{2,3} : \overrightarrow{R^{2,3}} \succ e_4^0 \succ e_5^1 \succ e_5^2 \succ e_4^3 \succ e_5^0 \succ e_4^1 \succ e_4^2 \succ e_5^3 \succ p,$$

where $R^{x,y} := C \setminus (\{p\} \cup E[x,y])$. For each of these voters we add a voter with reversed preferences. Further, for each pair $\{x,y\}$ of distinct colors, we add two voters, $\bar{q}_{x,y}$ and $\bar{q}'_{x,y}$, as follows:

$$\bar{q}_{x,y} : e^1_{\mathrm{id}_1^{x,y}} \succ e^2_{\mathrm{id}_1^{x,y}} \succ d \succ \overrightarrow{C \setminus (\{d, e^1_{\mathrm{id}_1^{x,y}}, e^2_{\mathrm{id}_1^{x,y}}\})},$$

$$\bar{q}'_{x,y} : \overleftarrow{C \setminus (\{d, e^1_{\mathrm{id}_1^{x,y}}, e^2_{\mathrm{id}_1^{x,y}}\})} \succ e^2_{\mathrm{id}_1^{x,y}} \succ e^1_{\mathrm{id}_1^{x,y}} \succ d.$$

Altogether, for each pair $\{x,y\}$ of distinct colors, candidates $e^1_{\mathrm{id}_1^{x,y}}$ and $e^2_{\mathrm{id}_1^{x,y}}$ get $\beta+1$ points while all other candidates get β points (except d, which gets less points). For each pair $\{x,y\}$ of distinct colors, we create 4 shift actions with effect -1 on voter $w_{x,y}$ and effect $+1$ on voter $w'_{x,y}$. △

Properties of the Construction. We now discuss several properties of our construction. These properties will play a significant rule in showing the correctness of the reduction. To illustrate our arguments, we will come back to our example from time to time. We begin by looking at the scores of the candidates.

Lemma 4.1. *The following claims hold:*

1) *In the unbribed election, every candidate receives at most $\alpha+\beta+2$ points and every candidate from $\{p\} \cup U \cup U' \cup E^0 \cup E^3$ receives exactly $\alpha+\beta+1$.*

2) *In every bribed election, the score of p is exactly $\alpha+\beta+1$.*

3) *After applying a successful set of shift actions, the score of p is $\alpha+\beta+1$ and the scores of all other candidates are at most $\alpha+\beta+1$.*

Proof. It is easy to see that the first claim holds, based on the discussion provided throughout the construction. The second claim holds since applying every shift action decreases the score of p by one point in one vote and increases it by one point in another vote. Note that the number of shift actions is set such that applying each shift action always moves p within the two votes on which the shift action acts. The last claim follows directly from the second one. (Lemma 4.1) □

Let us now consider the process of selecting vertices. In the description of vertex-selecting voters we mentioned that, initially, for each color i, vertex v_1^i is selected, and if for some integer t, $1 \le i \le n_i - 1$, we apply $t((h-1)\cdot d+2)$ shift actions that affect voters w_i and w'_i, then v_1^i ceases to be selected and v_{t+1}^i becomes selected. We now argue that, if we apply a number of these shift actions that is not divisible by $((h-1)\cdot d+2)$, then p is not a winner of the resulting election.

To see that this is the case, recall that in the preference orders of voter w_i and w'_i there are exactly $(h-1)\cdot d$ candidates from $E^1 \cup E^2$ between each pair of candidates $\{u_j^i, u_j'^i\}$. Furthermore, if p passes some candidate u_j^i in the preference order of voter w_i (increasing u_j^i's score by one), then it must also pass candidate u_j^i in the preference order of voter w'_i (decreasing u_j^i's score by one): otherwise, u_j^i would end up with score $\alpha+\beta+2$ and, by Lemma 4.1, p would not be a winner (there are no possibilities of influencing the score of u_j^i other than shifting p in the preference lists of w_i and w'_i). Hence, p also passes candidate $u_j'^i$ and all candidates between u_j^i and $u_j'^i$ in the preference lists of w_i and w'_i. This, however, means that if p is to be a winner of the election, then the number of applied shift actions with effects on voters w_i and w'_i is a multiple of $((h-1)\cdot d+2)$ (p passes candidate u_j^i, candidate $u_j'^i$, and $h\cdot d$ candidates in between). Figure 4.8 provides an illustration of the above reasoning.

We next discuss selecting edges. As for the case of vertex-selecting voters, in the description of our construction we have argued that (a) initially, for each pair $\{x,y\}$ of distinct colors, edge $e_{\mathrm{id}_1^{x,y}}$ is selected and that (b) after applying $4t$, $1 \le t \le n_{x,y}-1$, shift actions that affect voters $w_{x,y}$ and $w'_{x,y}$, $e_{\mathrm{id}_1^{x,y}}$ ceases to be selected and $e_{\mathrm{id}_{t+1}^{x,y}}$ becomes selected. We now argue that, if we used a number of

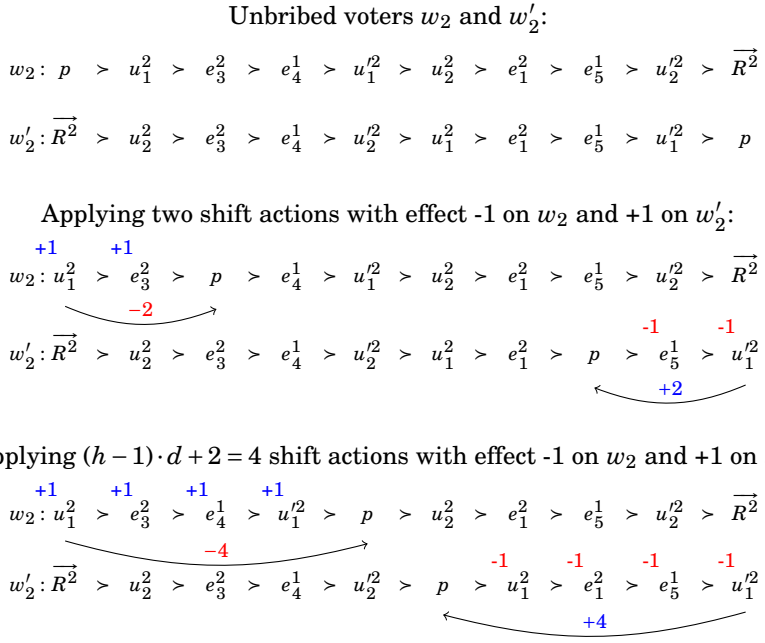


Figure 4.8. Illustration of bribery actions affecting the first voter group of our running example (Example 10). Note that, in the unbribed election, every candidate from $U \cup U'$ obtains $\alpha+\beta+1$ points in total. For each color i, there is only one type of shift actions which affects voter w_i and w_i': those shift actions with effect -1 on voter w_i and effect $+1$ on voter w_i'. No other shift action can affect any voter from the first group. Applying a multiple of $((h-1)\cdot d+2)$ shift actions with effect -1 on voter w_i and effect $+1$ on voter w_i' ensures that the candidates from $U^i \cup U'^i$ receive at most $\alpha+\beta+1$ points in total, whereas applying any other number of these shift actions implies that some candidate from U^i receives $\alpha+\beta+2$ points and, hence, p cannot win. We illustrate this with color 2 in our running example.

such shift actions that is not a multiple of four, then p certainly would not be a winner of the election.

To see that this is the case, note that we designed the preference orders of $w_{x,y}$ and $w'_{x,y}$ so that the candidates $e^0_{\mathrm{id}_j^{x,y}}$ and $e^3_{\mathrm{id}_j^{x,y}}$, for $j \in \{2,\dots,n_{x,y}\}$, follow p in vote $w_{x,y}$ in the same order in which they precede p in $w'_{x,y}$. In effect, if we apply a shift action that affects voters $w_{x,y}$ and $w'_{x,y}$ a number of times that is not a multiple of four, then one of these candidates obtains $\alpha+\beta+2$ points. Since there is no other way to affect the score of these candidates, by Lemma 4.1, it follows that, in this case, p cannot be a winner. We illustrate this effect in Figure 4.9.

Solution for Example 10. Before we complete the correctness proof, let us illustrate the solution for our example.

The unbribed election selects vertex v^1_1, v^2_1, and v^3_1 and the edges e_1, e_2 and e_4. Hence, for example, candidate e^2_4 receives $\alpha+\beta+2$ points and p (who receives only $\alpha+\beta+1$ points) is not a winner.

By applying four shift actions with effect -1 on w_2 and effect $+1$ on w'_2, we select v^2_2 instead of v^2_1 to be the vertex of color 2 in our clique (as depicted at the bottom of Figure 4.8). By applying four shift actions with effect -1 on $w_{2,3}$ and effect $+1$ on $w'_{2,3}$, we select e_5 instead of e_4 to be the edge between color 2 and color 3 in our clique (as depicted at the bottom of Figure 4.9). Now, each candidate from $\{e^1_1, e^1_2, e^2_1, e^2_2, e^1_5, e^2_5\}$ receives $\beta+1$ points from the edge-selecting voters, but only α points from the vertex-selecting voters. Every other candidate receives at most $\alpha+1$ points from the vertex-selecting voters and at most β points from the edge-selecting voters. Hence, p (with $\alpha+\beta+1$ points) is a winner. This solution corresponds to the left 3-colored triangle in Figure 4.7.

Correctness. It remains to show that there is a successful set of shift actions for the constructed Borda-COMBINATORIAL SHIFT BRIBERY instance if and only if there is an h-colored clique in graph G.

For the "if" direction, assume that there is an h-colored clique $H \subseteq V(G)$. Without loss of generality, let $H := \{v^1_{z_1}, \dots, v^h_{z_h}\}$ and let $E_H := \{\{v,v'\} : v,v' \in H\}$. Furthermore, let $z_{x,y}$ denote the index of the edge in $E_{x,y}$ representing the edge from E_H between the vertex of color x and the vertex of color y. That is, $z_{x,y} = j$ if and only if $e_{\mathrm{id}_j^{x,y}} \in E_H$. Then, it is easy to verify that the following set of shift actions is successful:

Unbribed voters $w_{2,3}$ and $w'_{2,3}$:

$$w_{2,3}\colon e_4^0 \succ e_4^1 \succ e_4^2 \succ e_4^3 \succ p \succ e_5^0 \succ e_5^1 \succ e_5^2 \succ e_5^3 \succ \overrightarrow{R^{2,3}}$$

$$w'_{2,3}\colon \overrightarrow{R^{2,3}} \succ e_4^0 \succ e_4^1 \succ e_4^2 \succ e_4^3 \succ e_5^0 \succ e_5^1 \succ e_5^2 \succ e_5^3 \succ p$$

Applying two shift actions with effect -1 on $w_{2,3}$ and +1 on $w'_{2,3}$:

$$w_{2,3}\colon e_4^0 \succ e_4^1 \succ e_4^2 \succ e_4^3 \succ \overset{+1}{e_5^0} \succ \overset{+1}{e_5^1} \succ p \succ e_5^2 \succ e_5^3 \succ \overrightarrow{R^{2,3}}$$

$$w'_{2,3}\colon \overrightarrow{R^{2,3}} \succ e_4^0 \succ e_4^1 \succ e_4^2 \succ e_4^3 \succ e_5^0 \succ e_5^1 \succ \overset{-1}{p} \succ \overset{-1}{e_4^2} \succ e_5^3$$

Applying four shift actions with effect -1 on $w_{2,3}$ and +1 on $w'_{2,3}$:

$$w_{2,3}\colon e_4^0 \succ e_4^1 \succ e_4^2 \succ e_4^3 \succ \overset{+1}{e_5^0} \succ \overset{+1}{e_5^1} \succ \overset{+1}{e_5^2} \succ \overset{+1}{e_5^3} \succ p \succ \overrightarrow{R^{2,3}}$$

$$w'_{2,3}\colon \overrightarrow{R^{2,3}} \succ e_4^0 \succ e_4^1 \succ e_4^2 \succ e_4^3 \succ p \succ e_5^0 \succ e_5^1 \succ e_5^2 \succ e_5^3$$

Figure 4.9. Illustration of bribery actions affecting the second voter group of our running example. Note that in the unbribed election, every candidate from $E^0 \cup E^4$ obtains $\alpha+\beta+1$ points in total. For each pair of colors x and y there is only one type of shift actions which affects voter $w_{x,y}$ and $w'_{x,y}$: those shift actions with effect -1 on voter $w_{x,y}$ and effect $+1$ on voter $w'_{x,y}$. No other shift action can affect any voter from the second group. Applying a multiple of 4 shift actions with effect -1 on voter $w_{x,y}$ and effect $+1$ on voter $w'_{x,y}$ ensures that the candidates from $E^0 \cup E^4$ receive at most $\alpha+\beta+1$ points from these voters, in total, whereas applying any other number of these shift actions implies that some candidate from E^0 receives $\alpha+\beta+2$ points and, hence, p cannot win. We illustrate this with color pair 2 and 3 in our running example.

1) For each color $i \in [h]$, include $(z_i - 1)((h-1) \cdot d + 2)$ shift actions with effects on voters w_i and w'_i.

2) For each pair $\{x, y\}$ of distinct colors, include $4(z_{x,y} - 1)$ shift actions with effects on voters $w_{x,y}$ and $w'_{x,y}$.

In other words, we select the vertices and the edges corresponding to the clique. In effect, the scores of all candidates is $\alpha + \beta + 1$ (except for d, who receives a lower score). Thus, p is among the tied winners.

For the "only if" direction, assume that there is a successful set of shift actions and consider the election after applying these shift actions. By construction, we know that edge-selecting voters pick exactly one edge for each pair of distinct colors. Hence, the graph induced by these edges contains vertices with h different colors. If this graph contains only h vertices, then this graph must be an h-colored clique (this graph cannot contain fewer than h vertices). Towards a contradiction, let us assume that this graph contains more than h vertices. Thus, there are two selected edges, e_j and $e_{j'}$, incident to two different vertices, $v_i \in e_j$ and $v_{i'} \in e_{j'}$, of the same color. By our construction (and the way vertex-selecting voters work), for at least one of the sets $N(v_i)$ and $N(v_{i'})$, each candidate in this set receives $\alpha + 1$ points from the vertex-selecting voters. Since both e_j and $e_{j'}$ are selected by the edge-selecting voters, however, these voters give $\beta + 1$ points to each of the candidates e^1_j, e^2_j, $e^1_{j'}$, and $e^2_{j'}$. Hence, at least one of these candidates receives $\alpha + \beta + 2$ points in total and, by Lemma 4.1, p is not a winner. This is a contradiction, and so the graph induced by the selected edges must be an h-colored clique. □

5. Anonymizing Elections

This chapter, similarly to Chapter 3 and Chapter 4, is about elections. The combinatorial problem discussed in this chapter is, however, not related to the possibility of manipulating elections, but to issues of privacy. We mention that Chapter 6 and Chapter 7 are also about privacy; not for data related to elections, however, but for data related to social networks.

Consider a scenario where the preference orders of some voters participating in an election are to be published. Specifically, we would like to publish all preference orders, together with some additional information regarding each voter (for example, name, occupation, etc.). It is clear that, in order to preserve the privacy of the voters, any identifying information (for example, the names of the voters) should be obfuscated before made public.

Obfuscating identifying information, however, is not enough for preserving the privacy of the voters. To see this, consider an adversary which knows the preference order of a specific voter whose vote is unique in the election. It is clear that such an adversary might de-anonymize that voter, thus breaching its privacy (and possibly learning some further private data which is associated to that voter).

We deal with the above privacy issue by *k-anonymizing* the election. That is, we are given an election, a voting rule, and a distance over elections (that is, a metric, modeling how different two elections are from each other). The task is to find an election which is not too far away from the original election (with respect to the given distance) while preserving the election winner (with respect to the given voting rule), and such that the resulting election will be k-anonymous; an election is said to be k-anonymous if, for every voter in it, there are at least $k-1$ other voters with the same preference order. Indeed, an adversary as described in the last paragraph cannot breach the privacy of the voters in a k-anonymous election.

Alice	:	Alice	$\succ$	Cinderella	$\succ$	David	$\succ$	Bob	$\succ$	Euclid
Bob	:	Euclid	$\succ$	Bob	$\succ$	David	$\succ$	Alice	$\succ$	Cinderella
Cinderella	:	Cinderella	$\succ$	Alice	$\succ$	David	$\succ$	Bob	$\succ$	Euclid
David	:	David	$\succ$	Cinderella	$\succ$	Euclid	$\succ$	Alice	$\succ$	Bob
Euclid	:	Cinderella	$\succ$	David	$\succ$	Euclid	$\succ$	Alice	$\succ$	Bob

Figure 5.1. The election used in the illustrating example.

We consider the problem of k-anonymizing elections for the Plurality rule and for the Condorcet rule, for the Discrete distance and for the Swap distance (formally defined in Section 5.3). We show that the (parameterized) complexity landscape of this problem is diverse, with cases ranging from being polynomial-time solvable to Para-NP-hard.

5.1. Illustrating Example

Consider the following example.

Example 11. Recall the group of people discussed in Section 3.1 and the election performed over their social network (Figure 3.2; for convenience, given also in Figure 5.1).

The corresponding examples in Section 3.1 and in Section 4.1 consider actions available for Bob in order to manipulate the election. Here, however, corresponding to the computational problem considered in this chapter, we assume that there is some identifying information attached to each voter participating in the election (for example, the age of each voter), and that we would like to make the election publicly available, while preserving the privacy of the voters participating in the election.

As a first step, we might want to obfuscate the names from the election, to arrive to the election-related data presented in Figure 5.2 (for example, Euclid's name has been obfuscated to be "E", while his real age (27) has been revealed).

Let us assume that an adversary wants to know the real age of Euclid. By simply looking at this published election-related data, it cannot distinguish Euclid

A (47) :	Alice	≻	Cinderella	≻	David	≻	Bob	≻ Euclid
B (14) :	Euclid	≻	Bob	≻	David	≻	Alice	≻ Cinderella
C (41) :	Cinderella	≻	Alice	≻	David	≻	Bob	≻ Euclid
D (70) :	David	≻	Cinderella	≻	Euclid	≻	Alice	≻ Bob
E (27) :	Cinderella	≻	David	≻	Euclid	≻	Alice	≻ Bob

Figure 5.2. The obfuscated election-related data.

from the other voters. However, if the adversary knows that the preference order of Euclid is

$$\text{Cinderella} \succ \text{David} \succ \text{Euclid} \succ \text{Alice} \succ \text{Bob},$$

then it can de-anonymize Euclid, since Euclid is the only voter with this preference order. Thus, we conclude that the above published election-related data does not preserve the privacy of the voters.

Hence, we require that each preference order would appear at least two times in the published election-related data. If this property, namely 2-anonymity, would hold, then the adversary discussed above will not be able to breach the privacy of the voters. In order to achieve the 2-anonymity property we allow swapping a few pairs of consecutive candidates.

In the current example, the minimum number of such swaps is eleven, and we reach the election-related data depicted in Figure 5.3 (specifically, we swapped Cinderella and Alice in the vote of C, David and Cinderella in the vote of D, and another nine swaps in the vote of B).

Indeed, in the resulting election-related data, each preference order appears at least twice. △

5.2. Introduction

In this chapter, we consider privacy issues when publishing preferences-related (or, election-related) data. Assume being given data consisting of a set of records, where each record (corresponding to a human or an agent) contains preferences-related information as well as some private (side) information. The task is to

A (47) : Alice ≻ Cinderella ≻ David ≻ Bob ≻ Euclid

B (14) : Alice ≻ Cinderella ≻ David ≻ Bob ≻ Euclid

C (41) : Alice ≻ Cinderella ≻ David ≻ Bob ≻ Euclid

D (70) : Cinderella ≻ David ≻ Euclid ≻ Alice ≻ Bob

E (27) : Cinderella ≻ David ≻ Euclid ≻ Alice ≻ Bob

Figure 5.3. The resulting 2-anonymized election-related data.

publish this data (for example, to let researchers analyze it) while preserving the privacy of the entities in it.

Two of the most well-studied approaches for achieving privacy when publishing information are *differential privacy* (see, for example, Dwork and Roth [59]) and *k-anonymity* (see, for example, Sweeney [138] and Machanavajjhala et al. [111]). Here, we follow the k-anonymity framework. Our main reason for following the k-anonymity framework is the fact that it allows for a deterministic combinatorial study, whereas differential privacy is a probabilistic method (see Clifton and Tassa [47] for a recent comparison between these two approaches). We mention that in Chapter 6 and in Chapter 7), as well, we follow the k-anonymity framework, but for anonymizing data related to social networks and not to elections.

We say that an election is *k-anonymous* if each preference order in it appears at least k times. Given an input election, the goal is to generate an election which is k-anonymous but still preserves some properties of the original election.

It is natural to consider the distance between the original election and the resulting anonymized election. To this end, we consider the following distances over elections. We study the Discrete distance (where each preference order can be transformed into any other preference order at unit cost) and the Swap distance (where each two consecutive candidates can be swapped at unit cost), as these are the most basic and well-studied distances defined on elections (see, for example, [64]). Indeed, we can view these distances also as defining the allowed operations (the Discrete distance allows to arbitrarily change a preference order while the Swap distance allows to swap two consecutive candidates). The idea is that, if the distance is small, then the anonymized election does not differ too

much from the original election; this is, arguably, more apparent in the Swap distance.[1]

Besides requiring that the original election and the resulting k-anonymized election will be close (with respect to the considered distance), we would like to preserve some specific properties of the original election.[2] Specifically, we require the winner of the election to be preserved. To this end, we need to consider voting rules. We study two voting rules, the Plurality rule and the Condorcet rule, as these are the most basic and well-studied voting rules, which are also good representative rules: the Plurality rule, albeit simple, can be seen as a representative rule for scoring rules, while the Condorcet rule can be seen as a representative rule for tournament-based rules (tournament-based rules are rules which depend on the tournament graph defined by pairwise comparisons between the candidates; see, for example, the book of Laslier [103] for more information).

In what follows, we study the parameterized complexity of k-anonymizing elections, under the Plurality rule and under the Condorcet rule, for the Discrete distance and for the Swap distance. We consider the two fundamental election-related parameters: the number of voters and the number of candidates; and an anonymity-related parameter: the anonymity level k. We show that the parameterized complexity landscape of our problem is diverse, with cases ranging from being polynomial-time solvable to Para-NP-hard.

In a way, this chapter can be seen as bringing the well-studied field of voting systems and social choice to the well-studied field of k-anonymity, with the hope of better understanding complexity issues of preserving privacy when publishing election-related data. Similarly, this chapter might serve as a bridge between Chapter 3 and Chapter 4, discussing election manipulation, to Chapter 6 and Chapter 7, discussing anonymization. We view our definition of k-anonymity for elections as being a natural adaptation of the concept of k-anonymity for tables and graphs to preferences-related (or, election-related) data.

5.2.1. Related Work

There is a big body of literature on security of elections and on preserving privacy of voters participating in (digital) elections. Chaum [40], Nurmi et al. [127],

[1] We mention that it is also possible to define the problems considered in Chapter 6 and in Chapter 7 through distances, albeit distances over graphs.

[2] Indeed, similar goals are also set in Chapter 6. There, specific properties of the original graph, such as the connectivity, the relative distances, and the diameter, are to be preserved.

and Cuvelier et al. [52], among others, considered cryptographic mechanisms to encrypt the votes, while Chen et al. [43], among others, considered differential privacy. Ashur and Dunkelman [4] showed how to breach the privacy of voters for the Israeli parliament when an adversary has access to the publicly-available nation-wide election statistics. This work is of some relevance to us as it considers privacy with respect to (publicly-available) published data.

Sweeney [138] introduced the concept of k-anonymity as a way to preserve privacy over published data, after demonstrating how to identify many individuals by mixing publicly-available medical data with publicly-available voter lists (interestingly, to some extent, Sweeney [138] already focuses on election-related data, specifically on the party affiliation; informally speaking, party affiliation corresponds to the Plurality rule, since only the first choice counts, while for the Condorcet rule we would need the complete preference orders publicly available). Much work has been done on k-anonymizing tables (for example, Meyerson and Williams [121], Bredereck et al. [25], and Bredereck et al. [29]; some of these concentrate on parameterized complexity) and on k-anonymizing graphs (for a literature review on k-anonymizing graphs we refer the reader to Section 6.2.1).

Here, we consider neither general tables nor graphs, but instead we consider elections. Indeed, elections can be described as tables, but here we require also to preserve the winner and allow different, election-specific, operations. Specifically, while the Discrete distance might be natural also for general tables, this is not the case for the Swap distance.

5.3. Specific Preliminaries

Considering distances over elections, we follow some of the notation used by Elkind et al. [64].

5.3.1. Elections and Distances

Given a set V' of preference orders, we say that a function $d : V' \times V' \to \mathbb{N}$ is a *distance function over preference orders* if it is a metric over preference orders. Given a distance function over preference orders d and two elections (over the same set of candidates), $E = (C,(v_1,\ldots,v_n))$ and $E' = (C,(v'_1,\ldots,v'_n))$, the above definition of a distance function over preference orders can be naturally extended to a distance function over elections by (1) fixing an arbitrary order for the voters of E, that is, $[v_1,\ldots,v_n]$, (2) considering all possible permutations for the voters

of E', that is, $[v'_{\pi(1)},\dots,v'_{\pi(n)}]$, and (3) defining the d-distance between E and E' to be $d(E,E') = \min_{\pi\in S_n}\sum_{i\in[n]} d(v_i, v'_{\pi(i)})$, where S_n is the set containing all possible permutations of $[n]$. We define the following distance functions.

- **Discrete distance.** $d_{\text{discr}}(v_1,v_2) = 0$ if and only if $v_1 = v_2$, while otherwise $d_{\text{discr}}(v_1,v_2) = 1$. Indeed, for two elections, $E = (C,(v_1,\dots,v_n))$ and $E' = (C,(v'_1,\dots,v'_n))$, it holds that $d_{\text{discr}}(E,E') = |\{i : v_i \neq v'_i\}|$.
- **Swap distance.** $d_{\text{swap}}(v_1,v_2) = |\{(c,c') \in C\times C : c \succ_{v_1} c' \ \wedge\ c' \succ_{v_2} c\}|$. Indeed, the swap distance $d_{\text{swap}}(v_1,v_2)$ (also called the Dodgson distance[3]) is the minimum number of swaps of consecutive candidates needed for transforming the preference order of v_1 to that of v_2.

Clearly, both the Discrete distance and the Swap distance are distance functions.

5.3.2. Anonymization

A group of voters with the same preference order is called a *block*. Using this notion, we have that an election is k-anonymous if and only if each block in it is of size at least k. We denote the number of voters in block B by $|B|$ and say that a block is *bad* if $0 < B < k$ (as it is not yet anonymized in this case). Since all voters in a block have the same preference order, it is valid to consider the preference order of the voters in the block. Specifically, a block of c-voters is called a *c-block*.

In order to make an election anonymous, we change the way some voters vote. When we say that we *move* a voter from one block to another, we mean that we transform the vote of the voter to be similar to the preference order of the other block.

5.3.3. Main Problem and Overview of Our Results

The main problem we consider in this chapter is defined as follows.

> $\mathscr{R}$-d-Election Anonymization ($\mathscr{R}$-d-EA)
> **Input:** An election $E = (C,V)$ where $C = \{c_1,\dots,c_m\}$ is the set of candidates and $V = (v_1,\dots,v_n)$ is the collection of voters, anonymity level k, and a budget s.

[3] Named after Charles Dodgson, better known as Lewis Carroll.

Table 5.1. Parameterized complexity of ELECTION ANONYMIZATION. The parameters considered are the number n of voters, the number m of candidates, and the anonymity level k.

	Discrete distance	Swap distance
Plurality rule	P [Theorem 5.1]	NP-h even when $n = k = 4$ [Theorem 5.3] FPT wrt. m [Theorem 5.5]
Condorcet rule	NP-h [Theorem 5.2] para. comp. wrt. k is open FPT wrt. n [Theorem 5.4] FPT wrt. m [Theorem 5.5]	NP-h even when $n = k = 4$ [Theorem 5.3] FPT wrt. m [Theorem 5.5]

> **Question:** Is there a k-anonymous election E' such that $\mathscr{R}(E) = \mathscr{R}(E')$ and $d(E,E') \leq s$ (where $\mathscr{R}$ is a voting rule, d is a distance function over elections, and an election is said to be k-anonymous if for each voter in it there are at least $k-1$ other voters with the same preference order)?

We study the (parameterized) complexity of $\mathscr{R}$-d-ELECTION ANONYMIZATION, where we consider both the Plurality rule and the Condorcet rule as the voting rule $\mathscr{R}$, and where we consider both the Discrete distance and the Swap distance as the distance d (that is, we consider the following four variants: PLURALITY-DISCRETE-EA, CONDORCET-DISCRETE-EA, PLURALITY-SWAP-EA, and CONDORCET-SWAP-EA). Our results are summarized in Table 5.1.

5.4. Results

Intuitively, from all variants considered in this chapter, PLURALITY-DISCRETE-EA should be the most tractable, as the Plurality rule is conceptually simpler than the Condorcet rule and the Discrete distance is conceptually simpler than the Swap distance. This intuition is correct: it turns out that PLURALITY-DISCRETE-EA is polynomial-time solvable, while all other variants are NP-hard. We begin by describing a polynomial-time algorithm, based on dynamic programming, for PLURALITY-DISCRETE-EA.

Theorem 5.1. PLURALITY-DISCRETE-EA *is polynomial-time solvable.*

Proof. We describe an algorithm based on applying dynamic programming twice, in a nested way. To understand the general idea, consider a candidate c with his corresponding c-voters. We have two cases to consider with respect to the solution election: either (1) some of the c-voters are transformed to be c'-voters (for some, possibly several, other candidates $c' \neq c$), or (2) some c'-voters (for some, possibly several, other candidates $c' \neq c$) are transformed to be c-voters. The crucial observation is that, with respect to anonymizing the c-voters, we do not need to remember the specific c'-voters discussed above, but only their number. Therefore, we define a first (*outer*) dynamic program, iterating over the candidates, and computing the most efficient way of anonymizing the c-voters, while considering all possible values for those numbers of c'-voters, and while making sure that the initial winners of the election stay winners. For each candidate c, in order to compute how to anonymize the c-voters, we define a second (*inner*) dynamic program, considering the c-blocks one at a time. For each c-block, the inner dynamic program decides whether to make the respective c-block empty (with zero voters) or full (with at least k voters), by considering the possible ways of transforming other c-voters or other c'-voters, similarly in spirit to the first (outer) dynamic programming. A full description of the algorithm follows.

We begin by guessing the end score of p, denoted by $\mathrm{end}(p)$, where p denotes the current winner (for simplicity, we assume that there is only one winner, but the algorithm works similarly for several initial winners). Since it is sufficient to consider only the number of voters from each block, the following notation will be useful.

For $i \in [m]$, let C_i denote the number of c_i-voters, let n_i denote the number of c_i-blocks, and for $j \in [n_i]$, let C_i^j denote the number of c_i-voters in the jth block of the c_i-voters. In particular, we have that $\sum_{j \in [n_i]} C_i^j = C_i$ holds for any $i \in [m]$. In

other words, we order the voters by their most preferred candidate, group them by their blocks, and consider only the size of each block, to arrive at the following sequence representation of the instance:

$$[C_1^1,\ldots,C_1^{n_1},C_2^1,\ldots,C_2^{n_2},\bullet\bullet\bullet,C_m^1,\ldots,C_m^{n_m}].$$

We iterate over the blocks, sorted as above, in a dynamic programming fashion. Specifically, we define a first (*outer*) dynamic programming table T such that

$$T(i,z_{1..i}),$$

for $i \in [m]$ and $-n \leq z_{1..i} \leq n$ (indeed, $z_{1..i}$ is just a symbol), represents the minimum number of operations needed in order to k-anonymize the voters corresponding to the first i candidates, that is, to anonymize the subsequence

$$[C_1^1,\ldots,C_1^{n_1},C_2^1,\ldots,C_2^{n_2},\bullet\bullet\bullet,C_i^1,\ldots,C_i^{n_i}],$$

while fulfilling the following requirements:

- **Respecting the guessed end score of** p. That is, we make sure that no candidate $c \neq p$ receives a greater score than end(p), which is the guessed end score of p.

- **Respecting the value of** $z_{1..i}$. Specifically, if $z_{1..i} < 0$, then this means that we should *import* exactly $-z_{1..i}$ voters. That is, we assume that we have a set of $z_{1..i}$ voters, all of which are originally c_{i_2}-voters, for some $i_2 > i$, and all these voters should end-up being c_{i_1}-voters, for some, possibly different, $i_1 \leq i$. In a similar way, if $z_{1..i} > 0$, then this means that we should *export* exactly $z_{1..i}$ voters. That is, we assume that we have a set of $z_{1..i}$ voters, all of which are originally c_{i_1}-voters, for $i_1 \leq i$, and such that these voters will end-up being c_{i_2}-voters, for some, possibly different $i_2 > i$.

As we want to k-anonymize all voters, we return True if and only if we have that $T(m,0) \leq s$, which means that it is possible to anonymize the blocks corresponding to all candidates while performing at most s operations.

Before we describe how to compute each $T(i,z_{1..i})$, we define, for each $i \in [m]$, a second (*inner*) dynamic programming table Q_i such that

$$Q_i(j,z_i^{1..j}),$$

for $j \in [n_i]$ and $-n \leq z_i^{1..j} \leq n$ (indeed, $z_i^{1..j}$ is just a symbol), represents the minimum number of operations needed in order to k-anonymize the voters corresponding to the c_i-voters, that is, to anonymize the subsequence

$$[C_i^1, \ldots, C_i^{n_i}],$$

while fulfilling the following requirement:

- **Respecting the value of** $z_i^{1..j}$. Specifically, if $z_i^{1..j} < 0$, then this means that we should *import* exactly $-z_i^{1..j}$ voters. That is, we assume that we have a set of $z_i^{1..j}$ voters, all of which are originally not voters from the first j blocks of the c_i-voters, and all these voters should end-up being in the first j blocks of the c_i-voters. In a similar way, if $z_i^{1..j} > 0$, then this means that we should *export* exactly $z_i^{1..j}$ voters. That is, we assume that we have a set of $z_i^{1..j}$ voters, all of which are originally voters from the first j blocks of the c_i-voters, and such that these voters should end-up being not in the first j blocks of the c_i-voters.

We describe now how to compute each $Q_i(j, z_i^{1..j})$ using our dynamic programming recursion. We consider all possible ways of anonymizing the jth block of the c_i-voters. Recall that the number of (c_i-)voters in the jth block of the c_i-voters is denoted by C_i^j, and let us denote the corresponding value in the solution, that is, the number of (c_i-)voters in the (same) jth block of the c_i-voters by C'^j_i. Since the number of voters is n, and due to the anonymization constraint, it follows that the possible values for each C'^j_i are either zero or some value between k and n (including); that is, we have that $C'^j_i \in \{0\} \cup [k, n]$. We consider all possible ways of anonymizing the jth block of the c_i-voters, that is, we consider the cost of achieving each possible value for C'^j_i. Specifically, we have the following recurrence relation:

$$Q_i(j, z_i^{1..j}) = \min_{C'^j_i \in \{0\} \cup [k,n]} Q_i(j-1, z_i^{1..j} - (C'^j_i - C_i^j)) + |C'^j_i - C_i^j|.$$

Note that, given any value for $z_i^{1..n_i}$, the minimum number of operations needed in order to anonymize all c_i-voters, is given by $Q_i(n_i, z_i^{1..n_i})$.

We describe now how to compute each $T(i, z_{1..i})$ using our dynamic programming recursion. We consider all possible ways of distributing the $z_{1..i}$ *import*/*export* voters to c_i-blocks and to c_{i_1}-blocks, for $i_1 < i$. Specifically, if $z_{1..i} > 0$,

then we consider all possible ways of exporting some voters from the c_i-blocks while the others are exported from c_{i_1}-blocks, for, possibly different, $i_1 < i$, and if $z_{1..i} < 0$, then we consider all possible ways of importing some voters to c_i-blocks while the others are imported to c_{i_1}-blocks, for, possibly different, $i_1 < i$. We have the following recurrence relation:

$$T(i, z_{1..i}) = \min_{-n \leq z_i^{1..n_i} \leq n} T(i-1, z_{1..i} - z_i^{1..n_i}) + Q_i(n_i, z_i^{1..n_i}).$$

For the proof of correctness, consider a yes-instance. It holds that, for each i, there is a correct value $z_{1..i}$ such that in the solution, the number $z_{1..i}$ represents the number of import/export voters with respect to the c_i-voters. This number $z_{1..i}$ will be identified by the outer dynamic programming, since it tries all possibilities for this number. Further, for each i and for each j, there is a correct value C'^j_i such that in the solution, the number C'^j_i represents the number of voters in the jth block of the c_i-voters. This number C'^j_i will be identified by the inner dynamic programming, since it tries all possibilities for this number. Finally, if the instance is a no-instance, then such numbers will not be identified by the algorithm.

The running time is clearly polynomial. Specifically, there are $O(mn)$ entries in the outer dynamic programming table T, since there are m candidates and $2n+1$ possible values for $z_{1..i}$. For each entry of T we consider $2n+1$ possibilities for $z_i^{1..n_i}$. There are m inner dynamic programming tables Q_i. Each inner dynamic programming table has $O(n^2)$ entries, since there are n voters and $2n+1$ possible values for $z_i^{1..n_i}$. For each entry of each Q_i we consider at most n values for C'^j_i. In total, we have running time of $O(m \cdot n^3)$. □

For the Condorcet rule, still considering the Discrete distance, we show that ELECTION ANONYMIZATION is NP-hard, by a reduction from a restricted variant of the EXACT COVER BY 3-SETS problem.

Theorem 5.2. CONDORCET-DISCRETE-EA *is* NP*-hard.*

Proof. We reduce from the NP-hard problem RESTRICTED EXACT COVER BY 3-SETS [86], defined as follows.

> RESTRICTED EXACT COVER BY 3-SETS
> **Input:** Collection $\mathcal{S} = \{S_1, \ldots, S_n\}$ of sets of size 3 over a universe $X = \{x_1, \ldots, x_n\}$ such that each element appears in exactly three sets.

Question: Is there a subset $\mathscr{S}' \subseteq \mathscr{S}$ such that each element x_i occurs in exactly one member of $\mathscr{S}'$?

We assume, without loss of generality, that

$$\{\{x_1,x_2,x_3\},\ldots,\{x_{n-2},x_{n-1},x_n\}\} = \{\{x_{3l-2},x_{3l-1},x_{3l}\} : l \in [n/3]\} \not\subseteq \mathscr{S},$$

and that $n = 0 \pmod 3$. Given an instance of RESTRICTED EXACT COVER BY 3-SETS, we create an instance for CONDORCET-DISCRETE-EA, as follows.

We create two candidates, p and d. For each element $x_i \in X$, we create a candidate x_i, such that the set of candidates is $\{X,p,d\}$. We create a set of $n/3$ voters, called *jokers*, such that the ith joker (for $i \in [n/3]$) has preference order $x_{3i-2} \succ x_{3i-1} \succ x_{3i} \succ p \succ d \succ X \setminus \{x_{3i-2},x_{3i-1},x_{3i}\}$. For each set S_j, we create k voters, each with preference order $S_j \succ p \succ d \succ \overline{S_j}$. We refer to these voters as the *set voters*. We create another set of $(n-6)k + ((n/3)-2)$ voters, each with preference order $d \succ X \succ p$, called the *init voters*. Finally, we set s to $n/3$ and k to $(n/3)+2$. This finishes the construction. Consider the following example of the reduction.

Example 12. Consider the following instance of RESTRICTED EXACT COVER BY 3-SETS. We have a universe $X = \{1,\ldots,9\}$ and a collection of sets $\mathscr{S} = \{S_1,\ldots,S_9\}$, where:

$$S_1 = \{1,2,5\}, \quad S_2 = \{1,3,6\}, \quad S_3 = \{1,4,5\},$$

$$S_4 = \{2,6,7\}, \quad S_5 = \{2,8,9\}, \quad S_6 = \{3,5,9\},$$

$$S_7 = \{3,6,7\}, \quad S_8 = \{4,7,8\}, \quad S_9 = \{4,8,9\}.$$

Note that $n = 9$ and that the collection $\mathscr{S}' = \{S_1,S_7,S_9\}$ forms an exact cover. The election $E = (C,V)$ created by the reduction described above is such that the set of candidates is $C = \{x_1,\ldots,x_9,p,d\}$, the collection of voters is $V = \{\text{jokers}, \text{set voters}, \text{init voters}\}$, and we have that the jokers are:

$$\text{(joker 1) 1 voter with: } 1 \succ 2 \succ 3 \succ p \succ d \succ 4 \succ 5 \succ 6 \succ 7 \succ 8 \succ 9,$$

$$\text{(joker 2) 1 voter with: } 4 \succ 5 \succ 6 \succ p \succ d \succ 1 \succ 2 \succ 3 \succ 7 \succ 8 \succ 9,$$

$$\text{(joker 3) 1 voter with: } 7 \succ 8 \succ 9 \succ p \succ d \succ 1 \succ 2 \succ 3 \succ 4 \succ 5 \succ 6,$$

the set voters are:

(set voter block 1) 5 voters with: $1 \succ 2 \succ 5 \succ p \succ d \succ 3 \succ 4 \succ 6 \succ 7 \succ 8 \succ 9$,

(set voter block 2) 5 voters with: $1 \succ 3 \succ 6 \succ p \succ d \succ 2 \succ 4 \succ 5 \succ 7 \succ 8 \succ 9$,

(set voter block 3) 5 voters with: $1 \succ 4 \succ 5 \succ p \succ d \succ 2 \succ 3 \succ 6 \succ 7 \succ 8 \succ 9$,

(set voter block 4) 5 voters with: $2 \succ 6 \succ 7 \succ p \succ d \succ 1 \succ 3 \succ 4 \succ 5 \succ 8 \succ 9$,

(set voter block 5) 5 voters with: $2 \succ 8 \succ 9 \succ p \succ d \succ 1 \succ 3 \succ 4 \succ 5 \succ 6 \succ 7$,

(set voter block 6) 5 voters with: $3 \succ 5 \succ 9 \succ p \succ d \succ 1 \succ 2 \succ 4 \succ 6 \succ 7 \succ 8$,

(set voter block 7) 5 voters with: $3 \succ 6 \succ 7 \succ p \succ d \succ 1 \succ 2 \succ 4 \succ 5 \succ 8 \succ 9$,

(set voter block 8) 5 voters with: $4 \succ 7 \succ 8 \succ p \succ d \succ 1 \succ 2 \succ 3 \succ 5 \succ 6 \succ 9$,

(set voter block 9) 5 voters with: $4 \succ 8 \succ 9 \succ p \succ d \succ 1 \succ 2 \succ 3 \succ 5 \succ 6 \succ 7$,

and the init voters are:

(init voters) 16 voters with: $d \succ 1 \succ 2 \succ 3 \succ 4 \succ 5 \succ 6 \succ 7 \succ 8 \succ 9 \succ p$.

Finally, the budget s is set to 3, and the anonymity level k is set to 5.

Corresponding to the exact cover $\mathscr{S}' = \{S_1, S_7, S_9\}$, we move joker 1 to set voter block 1, joker 2 to set voter block 7, and joker 3 to set voter block 9. As a result, the election becomes k-anonymized. △

Let us first compute the winner of the input election. Note that the set voters and the jokers prefer p to d, while the init voters prefer d to p. As there are kn set voters and $n/3$ jokers, but only $(n-6)k + ((n/3)-2)$ init voters, p defeats d. Consider an element x_i. Exactly one joker which prefers x_i to p and exactly three set voters prefer x_i to p (as x_i appears in exactly three sets). Besides these voters, all the init voters prefer x_i to p, while all other voters prefer p to x_i. In total, there are $(n/3) - 1 + k(n-3)$ voters which prefer p to x_i and $1 + 3k + (n-6)k + ((n/3)-2) = (n/3) - 1 + k(n-3)$ voters which prefer x_i to p, so p and x_i are tied (we use the weak-Condorcet criterion, but the reduction can be modified to apply for the strong-Condorcet criterion as well). Moreover, it is not hard to see that d defeats x_i. Summarizing the above computations, we see that

p is the winner of the input election. Thus, in an anonymized solution election, p shall be the winner as well.

Given an exact cover $\mathscr{S}'$, we move all jokers to the set voters corresponding to the exact cover, that is, we move all jokers such that, for each set $S \in \mathscr{S}'$, we will have one joker in the block of set voters corresponding to S. Note that initially the jokers form an exact cover. Note further that they still form an exact cover in the solution, as we moved them to blocks of set voters corresponding to the sets of $\mathscr{S}'$. Importantly, as a result, the relative score of p and the x_i's has not changed; thus, p is still the winner. Finally, it is not hard to see that the election is anonymized.

For the other direction, note that the jokers are not anonymized (that is, their blocks are bad; specifically, each joker forms its own block) while all other blocks are anonymized. It is not possible to anonymize the jokers by moving other voters (or other jokers) to their blocks, as the budget is too small for that. Therefore, in a solution, all jokers should move to other blocks. There are two possibilities for each joker, either to move to the init voters or to move to some set voters. It is not a good idea to move jokers to the init voters, as the init voters prefer X to p. More formally, if in a solution, some jokers move to the init voters, then we can instead move them to an arbitrarily-chosen set voter, while keeping p the winner and the election anonymized. Therefore, we can assume that, in a solution, all jokers move to set voters. If the jokers move to set voters in a way that does not correspond to an exact cover, then at least one x_i would win over p (as at least one x_i would be covered twice, that is, at least two jokers would move to set voters preferring x_i to p; therefore, the relative score between x_i and p would change in such a way that x_i would win over p in a head-to-head contest). Therefore, a solution must correspond to an exact cover, and we are done. □

Moving further to the Swap distance, we show that ELECTION ANONYMIZATION is NP-hard for both voting rules considered, even for elections with only four voters (indeed, also for elections with anonymity level four; note that any input with $k > n$ is a trivial no-instance). Technically, in the corresponding reduction, from KEMENY DISTANCE, we set both n and k to four. We mention that Theorem 5.3 actually holds for all *unanimous* voting rules (a voting rule is unanimous if for any election where all voters prefer the same candidate c, it selects this preferred candidate c; see, for example, [64]).

Theorem 5.3. *For $\mathscr{R} \in$ {Plurality,Condorcet}, $\mathscr{R}$-SWAP-EA is NP-hard even if the number n of voters is four and the anonymity level k is four.*

Proof. We reduce from the KEMENY DISTANCE problem.

KEMENY DISTANCE
Input: An election $E' = (C', V')$ and a positive integer h.
Question: Is the Kemeny distance of $E' = (C', V')$ at most h (where the Kemeny distance is the minimum total number of swaps of neighboring candidates needed to have all voters vote the same; such a vote is called a *Kemeny vote*)?

KEMENY DISTANCE is NP-hard already for four voters [14, 60]. Given an input election for KEMENY DISTANCE, we create an instance for ELECTION ANONYMIZATION, as follows.

We initialize our election $E = (C, V)$ with the election given for KEMENY DISTANCE and create another candidate c (that is, we set $C = C' \cup \{c\}$). For each voter in the election, we place c as the first choice of the voter, that is, for each voter $v' \in V'$, we create a voter $v \in V$ with the same preference order as v' while preferring c to all other candidates. We set the anonymity level k to n (such that, in particular, all voters shall vote the same) and the budget s to h. This finishes the construction. Consider the following example of the reduction.

Example 13. Consider the following instance of KEMENY DISTANCE. We have an election $E' = (C', V')$ with $C' = \{a', b', c', d'\}$ and $V' = \{v'_1, v'_2, v'_3, v'_4\}$, where

$$v'_1 : a' \succ b' \succ c' \succ d',$$

$$v'_2 : b' \succ a' \succ c' \succ d',$$

$$v'_3 : a' \succ c' \succ b' \succ d',$$

$$v'_4 : a' \succ b' \succ d' \succ c'.$$

It is easy to see that a Kemeny vote of E' is $a' \succ b' \succ c' \succ d'$ and that the Kemeny distance of E' is 3.

The election $E = (C, V)$ created by the reduction described above is such that the set of candidates is $C = \{a', b', c', d', c\}$, the collection of voters is $V = \{v_1, v_2, v_3, v_4\}$, and we have that:

$$v_1 : c \succ a' \succ b' \succ c' \succ d',$$

$$v_2 : c \succ b' \succ a' \succ c' \succ d',$$

$$v_3 : c \succ a' \succ c' \succ b' \succ d',$$

$$v_4 : c \succ a' \succ b' \succ d' \succ c'.$$

Finally, the anonymity level k is set to 4, therefore in a solution all voters should vote the same. The budget s is set to 3, and it is easy to see that this can be achieved most efficiently (that is, with the minimum number of swaps of consecutive candidates), when all voters vote as $c \succ a' \succ b' \succ c' \succ d'$. △

It is clear that originally c is the winner of the election (both under the Plurality rule and under the Condorcet rule; indeed, for Theorem 5.3, we only require the rule to be *unanimous*; see, for example, [64]).

The crucial observation is that there is no need to swap the new candidate c; this follows since all voters already agree on him, as they place him first in their preference orders. More formally, as k is set to n, it follows that if in a solution c is swapped in some voters, then he must be swapped in all voters. Therefore, we can simply "unswap" these swaps to get a cheaper solution.

Finally, since k is set to n, it follows that all voters should vote the same in the resulting k-anonymous solution election. Thus, the best way to anonymize the election is by finding a Kemeny vote, and transforming all voters to vote as this Kemeny vote. Therefore, the election can be k-anonymized by at most s swaps if and only if the Kemeny distance of the input election is at most h. □

With respect to the parameter number n of voters, the situation for the Discrete distance is different than the situation for the Swap distance. Specifically, it turns out that CONDORCET-DISCRETE-EA is FPT with respect to n.

Theorem 5.4. CONDORCET-DISCRETE-EA *is fixed-parameter tractable with respect to the number n of voters.*

Proof. The crucial observation is that there is no need to create new blocks, besides, perhaps, one arbitrarily-chosen p-block (recall that a p-block is a block of p-voters). To see this, consider a solution that adds a new block B which is not a p-block. Change the solution by moving the voters in B to a new arbitrarily-chosen p-block (instead of the block B). While using the same budget, the election is still anonymized and p is still a Condorcet winner. In effect, by applying the above modification extensively, we have shown that for each solution, there exist a solution which do not use any new blocks, besides, perhaps, one arbitrarily-chosen p-block. It follows that no new blocks, besides, perhaps, one additional arbitrarily-chosen p-block, are needed. We mention that an additional p-block might be needed when there are no p-voters (and, therefore, no p-blocks) in the input election.

The above observation suggests the following simple algorithm. We begin by guessing whether we need a new p-block. Then, for each voter, we guess whether (1) it will not change, (2) it will move to some other original block (out of the possible $n-1$ other original blocks), or (3) it will move to the new p-block (if we guessed that such a new block p-block exists). This completes the algorithm.

Correctness follows by the observation above and by the brute-force nature of the algorithm. Fixed-parameter tractability follows since, for each voter (out of the n original voters), we guess where it will end up (out of n or $n+1$ possibilities); thus, the running time is $O(m \cdot n^n)$. □

Still considering the parameter number n of voters, we mention that it seems possible to also obtain a problem kernel for CONDORCET-DISCRETE-EA. The idea would be to reduce the number of candidates to be upper-bounded by the parameter n. This could be done by performing the following modification to each block B of c-voters: introduce a new candidate c_B, and replace the votes in the block B to prefer c and then c_B. This is plausible since it is enough to keep the initial winners and to preserve the initial blocks.

We move on to consider the number m of candidates. Similarly to numerous other problems in computational social choice (for example, see Theorem 4.9), all variants of ELECTION ANONYMIZATION considered in this chapter are FPT with respect to this parameter. This follows by applying a famous result of Lenstra [104] after formulating the problem as an integer linear program where the number of variables is upper-bounded by a function dependent only on the number m of candidates.

Theorem 5.5. *For $\mathscr{R} \in$ {Plurality, Condorcet} and $d \in$ {Discrete, Swap}, $\mathscr{R}$-d-EA is fixed-parameter tractable with respect to the number m of candidates.*

Proof. The crucial observation is that the number of different preference orders is $m!$, thus, in particular, upper-bounded by a function depending only on the parameter. We enumerate the set of $m!$ different preference orders and, for each $i, j \in [m!]$, we create a variable $x_{i,j}$, with the intended meaning that $x_{i,j}$ will represent the number of voters with preference order i in the input and preference order j in the solution.

We add the following budget constraint:

$$\sum_{i,j \in [m!]} x_{i,j} \cdot d(i,j) \leq s.$$

(Note that we can precompute all the distances, in polynomial time.)

For preference order i, we denote the number of voters with preference order i in the input by start_i and the number of voters with preference order i in the solution by end_i. For each i, it holds that:

$$\text{end}_i = \text{start}_i + \sum_{j \in [m!]} x_{j,i} - \sum_{j \in [m!]} x_{i,j}.$$

We guess a set $Z \subseteq [m!]$ of preference orders with the intent that these will be the preference orders that will be present in the solution. For each preference order $i \in Z$ we add a k-anonymity constraint, and, similarly, for each preference order $i \notin Z$ we require that its end_i will be 0, as follows:

$$\forall i \in Z : \text{end}_i \geq k,$$

$$\forall i \notin Z : \text{end}_i = 0.$$

Differently for the Plurality rule and the Condorcet rule, we add more constraints to make sure that the winner of the input will not change in the solution election, as follows.

For the Plurality rule, we guess the highest score z in the solution (that is, the winning score), and we check that the initial winner p gets exactly z points, by adding the following constraint:

$$\sum_{i \in [m!] \text{ and } p \text{ is at the first position of } i} \text{end}_i = z.$$

Similarly, for each non-winning candidate $c \neq p$, we add a non-winning constraint:

$$\sum_{i \in [m!] \text{ and } c \text{ is at the first position of } i} \text{end}_i \leq z.$$

For the Condorcet rule, for the initial Condorcet winner p (if it exists), we check that he indeed beats all other candidates in the solution:

$$\forall c \neq p : \sum_{i \in [m!] \text{ and } p \succ_i c} \text{end}_i > \sum_{i \in [m!] \text{ and } c \succ_i p} \text{end}_i.$$

Since the number of variables and the number of constraints is upper-bounded by the parameter, fixed-parameter tractability follows by applying a famous result by Lenstra [104]. □

5.5. Outlook

There are numerous opportunities for future research regarding the work presented in this chapter, some of which we briefly discuss next.

- As Table 5.1 suggests, there is one open question left, considering the parameterized complexity of ELECTION ANONYMIZATION for the Discrete distance, under the Condorcet rule, when parameterizing by the anonymity level k. One might also consider other parameterizations for ELECTION ANONYMIZATION, the most natural yet-unstudied parameter being the solution size (that is, the budget) s.

- Theorem 5.3 shows hardness of $Plurality$-SWAP-EA and $Condorcet$-SWAP-EA for elections with only four voters. It would be interesting to know whether these problems are tractable for elections with only three voters. We mention that the related problem of winner-determination for Kemeny voting is open (that is, it is not known whether finding the Kemeny voter for elections with only three voters is NP-hard).

- While the algorithm presented in the proof of Theorem 5.4 shows fixed-parameter tractability of $Condorcet$-DISCRETE-EA, it's running time is $O(m \cdot n^n)$. It might be possible to devise a different algorithm, with better running time, which might be better in practice.

- It is natural to extend this line of research to some other voting rules. Indeed, some of the results described in this chapter can be extended to other

rules. For example, it was already stated before that the reduction showing Para-NP-hardness with respect to the combined parameter number n of voters and anonymity level k actually works for any unanimous voting rule. As another example, we mention that the algorithm showing fixed-parameter tractability with respect to the number of candidates m can be extended to any rule that can be expressed via integer linear program. It is not clear, however, whether the other results presented in this chapter can be extended to other rules. For example, one may ask whether the polynomial-time algorithm presented in Theorem 5.1 can be extended to the Borda rule.

Even further, one might consider Approval voting, where each voter chooses a subset of candidates which she approves. Stretching a bit further, one might also consider multi-winner rules, where the goal might be to anonymize the election while preserving the same winning committee.

- Another natural extension of this work is to consider other distances besides the Discrete distance and the Swap distance (see, for example, the distances considered by Elkind et al. [63]). On a similar note, we mention that it seems natural to also consider anonymization by making total orders into partial ones.

- While we consider a *minsum* approach, as we compute the distance between two elections as the sum of distances between their voters (and allow to permute the voters), it might also be interesting to study a *minmax* approach; in a *minmax* approach, intuitively, we would not allow any individual voter to change her vote by too much, but we would not care about the total changes summed over the voters. This could be of particular interest as it might preserve the original election more closely.

- More generally, it is worth studying how to preserve more properties of the original election; while we only require to preserve the winner, one might require to preserve the full relative ranking of the candidates (that is, fixing some scoring rule, and considering two candidates c' and c'', such that c' is achieving a higher score than c'' in the original election, we might require c' to achieve a higher score than c'' in the resulting election as well).

  Another type of properties that one might want to preserve relates to domain restrictions. For example, assuming that the input election is single-peaked, one might require the anonymized solution election to be

single-peaked as well (for an introduction to domain restrictions in general, and to single-peaked in particular, see, for example, the book by Arrow et al. [2]).

- One might consider stronger notions of k-anonymity. A natural notion to consider would be the notion of l-diversity [111], which might be interesting to explore in the context of elections.
- The NP-hardness, and even Para-NP-hardness, of some of the variants of ELECTION ANONYMIZATION considered here suggest trying to approximate the corresponding variants. To this end, one might either sacrifice the anonymity level or the solution size.

6. Degree Anonymization by Vertex Addition

In this chapter, similarly to Chapter 5, we consider privacy issues when publishing data. Here (as well as in Chapter 7), however, we do not consider publishing data related to elections, but we consider publishing data related to social networks.

Specifically, assuming an adversary which knows the degrees of some of the vertices of an input graph (which represents the social network that is to be published), we would like to make the graph k-anonymous; a graph is said to be k-anonymous if, for each vertex in it, there are at least $k-1$ other vertices with the same degree. Indeed, such an adversary as described above cannot breach the privacy of the vertices in a k-anonymous graph.

In contrast to Chapter 7, where the way to achieve anonymity is by performing few graph contractions, the way to achieve anonymity in this chapter is by adding few new vertices, together with some incident edges.

Similarly to Chapter 5, where we are interested in preserving some properties of the given election (specifically, the winner of the election), here we are interested in preserving some properties of the given graph. We explore three variants of vertex addition, differentiated by the restrictions imposed on the allowed new edges, making sure that some properties of the input graph will hold also in the anonymized solution graph.

We derive some intractability results, even for very restricted cases (including trees and bounded-degree graphs) and obtain some encouraging fixed-parameter tractability results.

6.1. Illustrating Example

Consider the following example.

Example 14. Recall the group of people discussed in Section 3.1 and the social network representing their friendship relationships (Figure 3.1, and given also in Figure 6.1 for convenience).

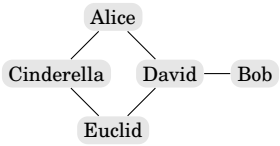


Figure 6.1. The social network used in the illustrating example.

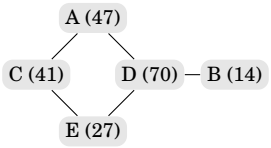


Figure 6.2. The obfuscated social network.

The corresponding example in Section 5.1 considers privacy issues when making some election-related data publicly available. Here, however, corresponding to the computational problem considered in this chapter, we assume that some identifying information is attached to the social network (for example, the age of each person), and that we would like to publish the social network, while preserving the privacy of the people comprising it.

We might obfuscate the names from the social network, to arrive at the social network presented in Figure 6.2 (for example, Euclid's name has been obfuscated to be "E", and his real age "27" is revealed). Let us assume that an adversary wants to know the real age of David. Simply by looking at this obfuscated social network, it cannot distinguish David from the others. However, if the adversary knows how many friends David has (that is, the degree of David), then it can de-anonymize David, since David is the only person in this social network with exactly three friends. In this sense, the above obfuscated social network is not preserving the privacy of the persons comprising the social network.

Hence, we would like to have that, in the published social network, each vertex degree would appear at least twice. If this property, namely 2-anonymity, would hold, then the adversary discussed above would not be able to breach the privacy of the persons comprising the social network. In order to achieve the above property, we would like to find the minimum number of new vertices (that is, new "dummy" people) that we can insert to the social network.

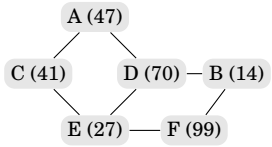


Figure 6.3. The resulting 2-anonymized social network.

In the current example, we can 2-anonymize the social network by adding only one new vertex, to reach the social network depicted in Figure 6.3 (specifically, we added one person, F, and connected him to both E and B).

Indeed, in the resulting social network, each vertex degree appears at least twice. △

6.2. Introduction

This chapter is concerned with making an undirected graph *k-anonymous*, that is, transforming it (at "low cost") into a graph where every vertex degree occurs either zero or at least k times. This graph modification scenario is motivated by data privacy requests in social networks; it focuses on degree-based attacks on identity disclosure of network nodes.

There is good reason why vertex addition may be preferred to other graph modification operations when aiming at k-anonymity. The central point here is the "utility" of the anonymized graph. For instance, in the edge addition scenario, inserting a new edge always destroys distance properties between vertices and indeed may introduce undesirable and misleading "fake relations". Adding new vertices and connecting them to some of the vertices of the original graph could avoid this problem and gives at least a better chance to preserve essential graph properties such as connectivity, shortest paths, or diameter. For example, adding a new vertex and connecting it to only one existing vertex does not change distances between any existing vertices. Chester et al. [44] further discussed the benefits of vertex addition.

6.2.1. Related Work

The general problem of degree anonymization has been studied, both theoretically and practically, for several graph modification operations. Liu and Terzi [107] (see also Clarkson et al. [46] for an extended version) pioneered degree-based identity anonymization in graphs, and gave a heuristic, based on dynamic programming, for k-anonymizing an input graph by adding as few edges as possible. Lu et al. [109] proposed another heuristic for the same problem, based on a greedy approach. Hartung et al. [91] studied the parameterized complexity of degree anonymization by edge additions. The main result there is an algorithm which runs in FPT-time when parameterizing by the maximum degree in the input graph. The problem of k-anonymizing an input graph by performing as few edge modifications as possible, that is, edge switchings, edge deletions, and edge additions, was studied by Casas-Roma et al. [39], which gave a heuristic solution for it. Bredereck et al. [24] considered degree anonymization by vertex deletion, again from a parameterized complexity point-of-view. Anonymization by adding new vertices was studied by Chester et al. [44], which gave a non-optimal algorithm with some approximation guarantees for the problem. We follow this last line of research, and provide a more thorough computational complexity analysis of degree anonymization by vertex addition.

6.2.2. Model Discussion

The basic decision version of the vertex addition problem we study is as follows (see Figure 6.4 for a simple example).

> DEGREE ANONYMIZATION (VA)
> **Input:** A simple undirected graph $G = (V, E)$ and $k, t \in \mathbb{N}$.
> **Question:** Is there a k-anonymous graph $G' = (V \cup V', E \cup E')$ such that $|V'| \leq t$ and $E' \subseteq \{\{u, v\} \subseteq V \cup V' : u \in V' \vee v \in V'\}$?

It is important to note that Chester et al. [44] studied a slightly different model, with decisive consequences for computational complexity: Their model gets as input a simple undirected graph $G = (V, E)$, integers t and k, and also a vertex subset $X \subseteq V$, and the task is to k-anonymize the degree sequence (that is, the vertex degrees sorted in ascending order) of $X \cup V'$ *and* the degree sequence of X. In contrast, we consider the simpler model where $X = V$, and we require to k-anonymize only the degree sequence of $X \cup V'$ $(= V \cup V')$.

To better understand the difference in the models, consider the example depicted in Figure 6.5. In this example, the minimum solution size for the model of Chester et al. [44] is four, while the minimum solution size for our model is two. The crucial difference is that in the solution for our model, the new vertex and the old vertex of degree five *together* will form a 2-anonymized "block". Nevertheless, we conjecture that our results (both positive and negative) extend to the model of Chester et al. [44].

6.2.3. Overview of Our Results

Partially answering an open question of Chester et al. [44], we show that DEGREE ANONYMIZATION (VA) is weakly NP-hard for a compact encoding of the input.

We provide several (fixed-parameter) tractability results, exploiting parameterizations by the maximum vertex degree of the input graph, the number of added vertices, and the maximum number of (implicitly) added new edges. The tractability result regarding the parameter maximum number of (implicitly) added new edges is given by developing a *bikernelization* [1, 100] to a closely related number problem. This is one of the most technical results in this chapter.

We consider several variants of degree anonymization by vertex addition, differentiated by the which edges we allow to introduce to the input graph.

DEGREE ANONYMIZATION (VA): Here we allow to insert edges which are incident to the new vertices.

DEGREE ANONYMIZATION (VC): Here we only allow "cloned" vertices to be added[1] (that is, identical copies of existing vertices with exactly the same neighborhood; see Figure 6.6 for an example).

Π-PRESERVING DEGREE ANONYMIZATION (VA): Here we explicitly demand the preservation of some desirable features of the input graph (expressed by Π) such as distance properties. For these practically interesting variants, we prove computational hardness already for very restricted cases (for instance even on trees).

Table 6.1 surveys most of our results, and some open questions.

[1]The cloning operation is frequently studied in the context of privacy, see, for example, the work by Bilge et al. [15]. It is also studied in the context of social choice: for example, the work by Elkind et al. [66] studied manipulating elections by cloning candidates.

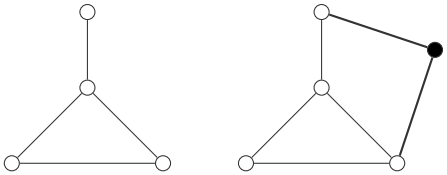


Figure 6.4. DEGREE ANONYMIZATION (VA): The input graph on the left is not 2-anonymous. The graph on the right is a possible solution for anonymizing by vertex addition. The new vertex (black) is arbitrarily connected to some other vertices.

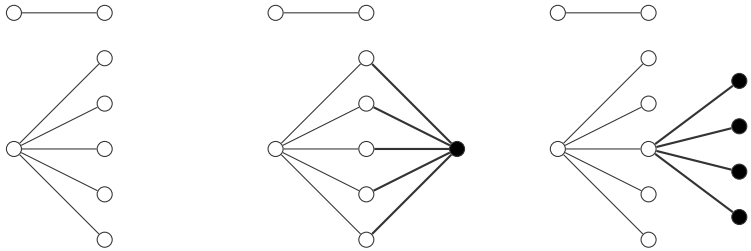


Figure 6.5. An example showing the difference between our model and the model of Chester et al. [44]. In this example, $k = 2$ and $X = V$. The input graph on the left is not 2-anonymous. The graph in the middle is a minimum solution for our model, using only one additional vertex, while the graph on the right is a minimum solution for the model of Chester et al. [44], using four addition vertices.

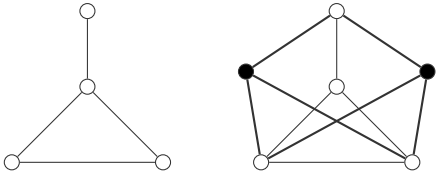


Figure 6.6. DEGREE ANONYMIZATION (VC): The input graph on the left is not 2-anonymous. The graph on the right is a possible solution for anonymizing by vertex cloning. The two added vertices (black) are clones of the middle vertex. Note that it is not possible to 2-anonymize the graph by adding only one clone.

Table 6.1. Overview of our results. Each column represents a different problem variant, where VC (respectively Π, VA) stands for DEGREE ANONYMIZATION (VC) (respectively Π-PRESERVING DEGREE ANONYMIZATION (VA), DEGREE ANONYMIZATION (VA)). The first row refers to standard complexity analysis, while the remaining rows show parameterized complexity results with respect to several parameters. Here, Δ denotes the maximum degree of the input graph, k is the degree of anonymity, s is the maximum number of added edges, and t is the maximum number of added vertices.

Parameter	VC	Π	VA
-	NP-h [Th. 6.1]	NP-h [Th. 6.3]	weakly NP-h [Th. 6.6]
Δ	Para-NP-h [Th. 6.1]	open	open
k	Para-NP-h [a] [Th. 6.2]	NP-h[a] [Th. 6.3]	open
s	open	W-h[b] [Th. 6.4]	FPT [Th. 6.10]
t	W-h [Th. 6.2]	W-h [Th. 6.3]	XP[c] [Th. 6.7]
(Δ, k)	open	open	FPT [Th. 6.9]
(Δ, t)	open	open	FPT [Th. 6.8]
(k, t)	W-h [Th. 6.2]	W-h [Th. 6.3]	XP[c] [Th. 6.7]

[a] Even on trees.
[b] Only for Π = Distances.
[c] Open whether FPT.

6.2.4. Organization of this Chapter

Problem definitions are given in Section 6.3. In Section 6.4, we consider degree anonymization by vertex addition, where there are some specific constraints on the allowed edges connecting the newly added vertices (including degree anonymization by vertex cloning). Here, we give strong hardness results (sometimes, even on trees). Then, in Section 6.5, we move on to consider degree anonymization by vertex addition, without constraints on the allowed edges connecting the newly added vertices. Here, we solve some easy cases in Section 6.5.1, then we give a hardness result in Section 6.5.2, accompanied by some tractable cases in Section 6.5.3. We conclude in Section 6.6.

6.3. Specific Preliminaries

Degree Anonymization (VA) allows to add vertices and edges incident to the new vertices. For a given solution of a yes-instance, we denote the actual number of new vertices by t' and the total number of newly inserted edges by s' (indeed, $0 \leq t' \leq t$ and $0 \leq s' \leq s$).

Π-Preserving Degree Anonymization (VA) defines some constraints on the new edges. The idea is to preserve some desirable properties of the input graph. A general definition reads as follows.

> Π-Preserving Degree Anonymization (VA)
> **Input:** An undirected graph $G = (V, E)$ and $k, t \in \mathbb{N}$.
> **Question:** Is there a k-anonymous graph $G' = (V \cup V', E \cup E')$ such that $|V'| \leq t$, $E' \subseteq \big\{\{u, v\} \subseteq V \cup V' : u \in V' \vee v \in V'\big\}$, and Π is preserved?

We now discuss what "Π is preserved" means for three properties we consider here. First, we say that the *connectedness* remains preserved if any pair of disconnected vertices in G remains disconnected in G'. As introducing vertices and edges cannot disconnect vertices, this property can be formalized as:

$$\forall u, v \in V\colon \operatorname{dist}_G(u, v) = \infty \iff \operatorname{dist}_{G'}(u, v) = \infty.$$

Second, we say that the *distances* remain preserved if, for any pair of vertices in G, their distance is the same in G and G', formally:

$$\forall u, v \in V\colon \operatorname{dist}_G(u, v) = \operatorname{dist}_{G'}(u, v).$$

Third, we say that the *diameter* remains unchanged if the diameter of G and G' is the same, formally:

$$\max_{u,v \in V} \mathrm{dist}_G(u,v) = \max_{u,v \in V \cup V'} \mathrm{dist}_{G'}(u,v).$$

Note that the diameter property also considers paths between newly added vertices, whereas this is not the case for the first two properties. The reason for this is that the diameter is naturally defined as a single number, whereas the other properties store information for each pair of vertices.

A further restricted variant of DEGREE ANONYMIZATION (VA) is to use vertex cloning for modifying the graph. Here, cloning a vertex v means to introduce a new vertex v' and make v' adjacent to all neighbors of v. Formally, we arrive at the following problem:

> DEGREE ANONYMIZATION (VC)
> **Input:** An undirected graph $G = (V,E)$ and $k,t \in \mathbb{N}$.
> **Question:** Can G be transformed into a k-anonymous graph by at most t vertex cloning operations?

We remark that there are two different cloning variants: consider two adjacent vertices u and v. If both u and v are cloned, then although the clone u' is adjacent to v and the clone v' is adjacent to u, the clones u' and v' may or may not be adjacent depending on the variant. If the clones are inserted simultaneously at the same time, then u' and v' are not adjacent. If the clones are inserted one after the other, then u' and v' are adjacent (no matter in which order they are inserted). Our results for DEGREE ANONYMIZATION (VC) (Theorems 6.1 and 6.2) hold for both variants.

6.4. Constrained Degree Anonymization

Cloning is a natural and well-motivated modification operation for social networks. Unfortunately, we face computational intractability even on very restricted input graphs with maximum degree three. The corresponding parameterized reduction is from CUBIC INDEPENDENT SET.

Theorem 6.1. DEGREE ANONYMIZATION (VC) *is* NP*-hard, even on graphs with maximum degree three.*

Proof. We provide a reduction from the CUBIC INDEPENDENT SET problem which remains NP-hard even on cubic graphs (that is, 3-regular graphs) [84].

> CUBIC INDEPENDENT SET
> **Input:** An undirected 3-regular graph G and an integer h.
> **Question:** Is there a set of h pairwise non-adjacent vertices?

Let $G = (V, E)$ be a cubic graph. First, we construct the incidence graph $G' = (V', E')$ of G, where G' is a bipartite graph with the two vertex sets V and E and for each edge $e = \{u, v\} \in E$, we have $\{u, e\}, \{v, e\} \in E'$. Then, we add $n + 2(h+3)^2$ triangles (K_3) and $(h+3)^2$ cliques of order four (K_4). Finally, we set $t := h$ and $k := n + 4(h+3)^2 + 4h$.

It remains to show that G has an independent set of size at least h if and only if G' can be k-anonymized by cloning at most t vertices.

For the "if" direction, let $I \subseteq V$ denote an independent set of size h. Denote by G'' the graph that results from cloning all vertices of I in G'. We show that G'' is k-anonymous. First, observe that G' contains exactly $n + 4(h+3)^2$ vertices of degree three and $1.5n + 3(n + 2(h+3)^2)$ vertices of degree two. Cloning a vertex of I in G' increases the degree of three degree-two vertices by one and introduces one degree-three vertex. Furthermore, as I is an independent set, the degree of no vertex is increased to four. Hence, G'' contains $n + 4(h+3)^2 + 4t = k$ degree-three vertices and more than $3(n + 2(h+3)^2) > k$ degree-two vertices and thus is k-anonymous.

For the "only if" direction, let G'' be the k-anonymous graph obtained from cloning t vertices in G'. First, observe that cloning t vertices in G' can increase the maximum degree by at most t. Thus, $\Delta_{G''} \leq 3 + t$. Next, we show that G'' contains no vertex of degree four or more. Assume towards a contradiction that there exists a vertex v of degree four or more. As cloning one vertex u can introduce at most $\Delta_{G''} + 1$ vertices of degree $\deg_{G''}(v)$ (namely u and its neighbors) and since there are no degree-$\deg_{G''}(v)$ vertices in G', there are at most $t(\Delta_{G''} + 1) \leq h(h+4) < k$ vertices of degree $\deg_{G''}(v)$. This is a contradiction to the assumption that G'' is k-anonymous. Hence, all vertices in G'' have degree at most three.

Observe that cloning a vertex $e \in E \subseteq V'$ or any vertex in one of the K_4's creates degree-four vertices and, hence, no such vertex is cloned to obtain G''. Cloning two vertices (or one vertex twice) of a triangle creates a vertex of degree four. Hence, at most one vertex of each triangle is cloned, creating exactly two degree-three vertices. Since cloning a vertex $v \in V \subseteq V'$ introduces at most four degree-three

vertices and G'' has to have at least $4t$ degree-three vertices more than G', it follows that only vertices in V are cloned to obtain G''. Furthermore, as cloning two vertices that are adjacent in G or one vertex twice introduces a degree-four vertex, it follows that the cloned vertices form an independent set of size $t = h$ in G. □

Also from the viewpoint of fixed-parameter algorithms, we have no good news with respect to the standard parameter "solution size" t, even on trees. The corresponding parameterized reduction is from SET COVER.

Theorem 6.2. DEGREE ANONYMIZATION (VC) *is* NP*-hard and* W[2]*-hard with respect to the maximum number t of clones, even if the anonymity level k is two and the graph is a tree.*

Proof. We provide a reduction from the W[2]-complete SET COVER problem parameterized by the solution size [57].

> SET COVER
> **Input:** A universe of elements X, a collection $\mathcal{S}$ of sets of elements of X, and a budget h.
> **Question:** Is there a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ of sets such that $|\mathcal{S}'| \leq h$ and $\bigcup_{S \in \mathcal{S}'} S = X$?

Given a SET COVER instance $I = (\mathcal{S}, X, h)$, we construct a graph $G = (V, E)$ as follows. For each $x_i \in X$, we define $f(x_i) := (h+3)i + m$. First, for each element $x \in X$, we add a star with $f(x)$ leaves (a $K_{1,f(x)}$) whose center-vertex is denoted by v_x. Next, for each set $S \in \mathcal{S}$, we add a star $K_{1,2|S|}$ and a *set gadget* consisting of a tree of depth three. The root of the tree is v_S with $2|S|$ child vertices, partitioned into $|S|$ parts with two vertices each. The two vertices of each part correspond to one element $x \in S$ and have $f(x) - 2$ degree-one child vertices each and, thus, a degree of $f(x) - 1$. Finally, we set $t = h$ and $k = 2$.

It remains to show that I has a set cover of size h if and only if G can be k-anonymized by cloning at most t vertices. To this end, first observe, that the vertices v_x, for $x \in X$, are the only vertices violating 2-anonymity. We denote the set containing all these vertices by V_X.

For the "if" direction, let $\mathcal{S}' \subseteq \mathcal{S}$ be a set cover of size h. We show that cloning all vertices in the set $V_C := \{v_S : S \in \mathcal{S}'\}$ (containing the vertices corresponding to the sets in $\mathcal{S}'$) results in a 2-anonymous graph. Since $\mathcal{S}'$ is a set cover, for every vertex $v_x \in V_X$, there exists a corresponding pair of vertices in a set-gadget whose

degree is increased from $f(x)-1$ to $f(x)$ due to the cloning. Furthermore, no new degrees are introduced, as each neighbor v of a cloned vertex corresponds to an element x and $\deg_G(v) = f(x)-1$.

For the "only if" direction, let V_C, $|V_C| \leq t$, denote the set of cloned vertices and let G' be the resulting 2-anonymous graph. For each set $S \in \mathcal{S}$, C_S denotes the set of all child vertices of v_S. Define $C_{\mathcal{S}} := \bigcup_{S \in \mathcal{S}} C_S$. Furthermore, set $V_{\mathcal{S}} := \{v_S : S \in \mathcal{S}\}$. Observe that, for each $x \in X$, the only vertices in G whose degrees differ by at most t from v_x are vertices in $C_{\mathcal{S}}$. Furthermore, note that, by construction, cloning a vertex from $V \setminus V_{\mathcal{S}}$ introduces at most one vertex having the same degree as a vertex from V_X. As cloning a vertex in $V_{\mathcal{S}}$ neither decreases the size of any block to one nor introduces a block of size one, and since every vertex in $C_{\mathcal{S}}$ has a neighbor in $V_{\mathcal{S}}$, we can assume that $V_C \subseteq V_{\mathcal{S}}$ (otherwise, we can replace vertices in $V_C \setminus V_{\mathcal{S}}$ with vertices in $V_{\mathcal{S}}$). Next, observe that, for each vertex $v_x \in V_X$, there has to be another vertex in G' having the same degree as v_x. Thus, the vertices in $V_C \subseteq V_{\mathcal{S}}$ correspond to a set cover of size at most $t = h$. □

We can adjust the reduction from Theorem 6.2 to also work for Π-PRESERVING DEGREE ANONYMIZATION (VA), by constructing the graph in such a way that there are only few possibilities to connect the newly added vertices to the remaining graph.

Theorem 6.3. *For* $\Pi \in$ {*Distances, Diameter, Connectivity*}, Π-PRESERVING DEGREE ANONYMIZATION (VA) *is* NP*-hard and also* W[2]*-hard with respect to the maximum number* t *of added vertices, even if the anonymity level* k *is* 2*. For* $\Pi \in$ {*Distances, Connectivity*}*, this is also true on trees.*

The proof of Theorem 6.3 is deferred to the appendix to this chapter.

We strengthen parts of Theorem 6.3 (using a reduction from the W[1]-hard problem CLIQUE) by also showing that the problem remains intractable with respect to the typically larger parameter number s of added edges. For simplicity, we consider s as part of the input.

Theorem 6.4. *For* $\Pi =$ *Distances,* Π-PRESERVING DEGREE ANONYMIZATION (VA) *is* W[1]*-hard with respect to the maximum number* s *of new edges.*

Proof. We give a reduction from the W[1]-complete CLIQUE problem parameterized by the clique size [57].

CLIQUE
Input: An undirected graph G and an integer h.
Question: Is there a set of h pairwise adjacent vertices?

Given a CLIQUE instance $I := (G,h)$, we construct a graph $G' = (V',E')$ such that G has a clique of size h if and only if G' can be k-anonymized by adding at most t new vertices and at most s new edges, that is, an instance $I' := (G',k,t,s)$. For each $z \in [t+1]$, we define $f(z) := (t+1)(h-1)+z(t+3)$, and add a star $K_{1,f(z)}$ with center vertex v_z to G'. For each $z \in [t+1]$ and for each edge $e_i \in E = \{e_1,\ldots,e_m\}$, we add a vertex (denoted by $v_{z,i}$), and we say that e_i is the corresponding edge of $v_{z,i}$.

For each pair $z_1,z_2 \in [t+1]$ and for each pair of edges e_{i_1} and e_{i_2}, we connect v_{z_1,i_1} and v_{z_2,i_2} by a path of length two if the corresponding edges e_{i_1} and e_{i_2} share a common vertex as an endpoint. We add some new leaves to each $v_{z,i}$ such that the degree of each $v_{z,i}$ is changed to $f(z)-2$. We add some *safety gadgets* for the new vertices to safely fall into their block: Specifically, we create k stars $K_{1,(t+1)(h-1)}$. Finally, we set $k = \binom{h}{2}+1$, $t = h$, and $s = 2(t+1)\binom{h}{2}$.

It remains to show that I is a yes-instance if and only if (G,k,t) is a yes-instance.

For the "if" direction, given a clique of size h in G, we add one new vertex for each clique vertex v connecting it to all $v_{z,i}$ where the corresponding edge e_i is incident to v and inside the clique. We show that this operation is permitted, as we do not violate Π. Indeed, it holds that each pair v_{z,i_1},v_{z,i_2} whose corresponding edges e_{i_1},e_{i_2} share a common vertex already had distance two.

We now show that the graph is k-anonymous. First, note that the only vertices that need to be anonymized are the v_z's. The degree of each $v_{z,i}$ whose corresponding edge is inside the clique was incremented by exactly two, and since there are exactly $\binom{h}{2}$ such edges, the v_z's are now anonymized. We can assume that $m \geq k+\binom{h}{2}$ (as this case can be easily dealt with), and, therefore, no new blocks were introduced. Finally, the new vertices fall into the block anonymized by the safety gadgets.

For the "only if" direction, consider a new vertex w, a pair $z_1,z_2 \in [t+1]$, and a pair of edges e_{i_1},e_{i_2}. The vertices v_{z_1,i_1} and v_{z_2,i_2} cannot both be connected to w, since the original distance between them was at least four and would be changed to two if both were connected to w.

Now, assume that (G,k,t,s) is a yes-instance. As only t new vertices are added, there exists a $z' \in [t+1]$ such that no new vertex has the same degree as $v_{z'}$. Since there is a gap of $t+1$ above $v_{z'}$, it must be anonymized from below. The only possible vertices to use are the $v_{z',i}$. At least $\binom{h}{2}$ of the $v_{z',i}$ must reach the degree of $v_{z'}$ and since we can associate an original vertex to each new vertex, it follows that they correspond to a clique. □

6.5. Plain Degree Anonymization

In this main section of this chapter, we study the unrestricted problem DEGREE ANONYMIZATION (VA), without any restrictions on how to connect the new vertices to the input graph. This freedom might raise hope to find solutions more efficiently. Indeed, settling the computational complexity of DEGREE ANONYMIZATION (VA) turns out to be tricky in that, on the one hand, we observe that several cases are fairly easy to solve, but we are not aware of any polynomial-time algorithm solving the problem in general. On the other hand, we can only prove weak NP-hardness for a number version of the problem.

In terms of fixed-parameter tractability, however, it turns out that the DEGREE ANONYMIZATION (VA) problem is more accessible. We obtain some fixed-parameter tractability results regarding, amongst others, certain (combined) parameters (for example, s, (Δ, k), and (Δ, t)), for some of which we proved the cloning and property-preserving problem variants to be W-hard.

6.5.1. Easy Cases

We start by analyzing the complexity of DEGREE ANONYMIZATION (VA) with respect to the two input values degree k of anonymity and number t of added vertices. Figure 6.7 provides a two-dimensional map indicating those combinations of k and t for which the problem is polynomial-time solvable or even trivial. In the following, we briefly state the corresponding results, starting with the following easy observation.

Observation 1. *Let $I = (G, k, t)$ be an instance of* DEGREE ANONYMIZATION (VA) *with G being an n-vertex graph. If $k > n + t$, then I is a no-instance.*

This holds as there are not enough vertices to make the graph anonymous, even if all vertices are in the same block, that is, the resulting graph is regular. For the other two solvable cases in Figure 6.7, we use the following result by Erdős and Kelly [71].

Theorem 6.5 (Erdős and Kelly [71])**.** *Let $G = (V, E)$ be a graph with n vertices, maximum degree Δ, and minimum degree δ. Let $d \geq \Delta$ be some integer and let $\xi = \sum_{v \in V}(d - \deg(v))$. Then, there exists a d-regular graph H with $n + t$ vertices containing G as an induced subgraph if and only if:*

1) $td \geq \xi$,

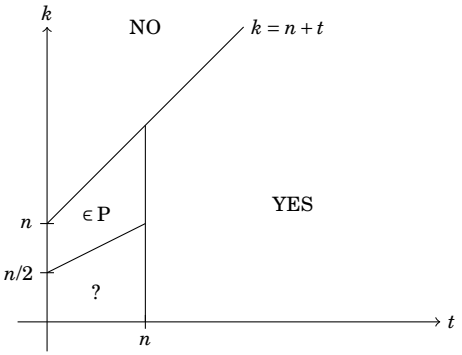


Figure 6.7. Visualization of our knowledge about the complexity of DEGREE ANONYMIZATION (VA) depending on the degree k of anonymity, the number n of vertices in the input graph, and the maximum number t of added vertices. The NO-cases follow from Observation 1, the YES-cases are due to Proposition 6.1, and the polynomial-time solvable cases follow from Proposition 6.2. For values inside the "?-area", the complexity is open for the graph problem (the number version is shown to be weakly NP-hard, see Theorem 6.6).

2) $t^2 - (d+1)t + \xi \geq 0$,

3) $t \geq d - \delta$, *and*

4) $(t+n)d$ *is even.*

We remark that the proof given by Erdős and Kelly [71] is stated for the case $d = \Delta$ but actually proves this more general result. We also remark that the proof is constructive (indeed, it uses a result of Erdős and Gallai [72], which has a corresponding constructive version due to Hakimi [89] and Havel [92]).

First, it follows from Theorem 6.5 that if we are allowed to add enough new vertices (that is, at least n), then it is always possible to construct a regular graph (which is also clearly anonymous), as shown next.

Proposition 6.1. *Let* $I = (G, k, t)$ *be an instance of* DEGREE ANONYMIZATION (VA) *with* G *being an* n*-vertex graph. If* $k \leq n + t$ *and* $t \geq n$*, then* I *is a yes-instance.*

Proof. We use Theorem 6.5 to show that for $t \geq n$ there always exists a Δ- or $(\Delta+1)$-regular supergraph H with $n+t$ vertices containing the input graph G as induced subgraph. To this end, we check Conditions (i) to (iv) of Theorem 6.5. First, observe that $\xi \leq n \cdot d \leq t \cdot d$ and, thus, Condition (i) is satisfied for $d = \Delta$ and $d = \Delta + 1$. Second, as $\Delta \leq n - 1$ in each graph, it follows that, for $d = \Delta$ we have that

$$t^2 - (d+1)t + \xi \geq t(t - (n - 1 + 1)) + \xi = \xi \geq 0,$$

and for $d = \Delta + 1$ we have that

$$t^2 - (d+1)t + \xi \geq t(t - (n+1)) + \xi \geq \xi - n \geq 0,$$

as in the case of $d = \Delta + 1$ it follows that $\xi \geq n$. Third, we have that

$$t \geq n \geq \Delta + 1 \geq d \geq d - \delta.$$

Fourth, observe that either $(n+t)\Delta$ or $(n+t)(\Delta+1)$ is even. Hence, if we are given a DEGREE ANONYMIZATION (VA) instance with $t \geq n$, then it is always possible to create a regular graph with $n+t$ vertices. Thus, if $t \geq n$, then the DEGREE ANONYMIZATION (VA) is a yes-instance if and only if $k \leq t + n$. □

We can also use Theorem 6.5 algorithmically, as follows.

Proposition 6.2. DEGREE ANONYMIZATION (VA) *is polynomial-time solvable for* $2k > (n+t)$.

Proof. Let $I = (G = (V,E), k, t)$ be an instance of DEGREE ANONYMIZATION (VA) with $2k > (n+t)$. By Observation 1 and Proposition 6.1, we can assume that $k \leq n+t < 2k$ and $t < n$. Observe that in this case any solution (if it exists) transforms G into a regular graph. Hence, the question is whether there is a regular graph H with at most $n+t$ vertices containing G as induced subgraph.

Our algorithm is as follows. First, we guess in $O(n^2)$ time the number $t' \leq t$ of vertices that we will add and the degree d of the final regular graph H. Then, we reduce to Theorem 6.5, that is, we compute $\xi = \sum_{v \in V} d - \deg(v)$ and check the four inequalities. □

6.5.2. (Weak) NP-Hardness

In Figure 6.7, we left open the computational complexity of DEGREE ANONYMIZATION (VA) for instances with $2k \leq n+t$. We now partially settle this question by proving that a closely related number version of the problem is weakly NP-hard. To this end, note that since we are not allowed to add any edges between old vertices, the actual structure of the input graph G becomes negligible and we only need to store the information of how many vertices of which degree it contains (that is, its block sequence $B(G)$):

Observation 2. *Let G and G' be two graphs with identical block sequences, that is, $B(G) = B(G')$. Then, for the* DEGREE ANONYMIZATION (VA) *instances $I := (G,k,t)$ and $I' := (G',k,t)$, it holds that I is a yes-instance if and only if I' is a yes-instance.*

Based on Observation 2, we can now define a closely related number version of DEGREE ANONYMIZATION (VA).

> BLOCK SEQUENCE ANONYMIZATION (VA)
> **Input:** A realizable block sequence B and $k, t \in \mathbb{N}$.
> **Question:** Is there an undirected graph G with block sequence B such that (G,k,t) is a yes-instance of DEGREE ANONYMIZATION (VA)?

Note that BLOCK SEQUENCE ANONYMIZATION (VA) is a pure number problem (that is, its input consists only of numbers). This helps us to develop a polynomial-time reduction from a weakly NP-hard version of the SUBSET SUM problem. An NP-hard problem is *weakly* NP-hard if it can be solved in polynomial-time provided that the input is encoded in unary. We obtain the following theorem.

Theorem 6.6. BLOCK SEQUENCE ANONYMIZATION (VA) *is weakly* NP*-hard.*

Proof. We reduce from the weakly NP-hard CHANGE MAKING problem [110].

> CHANGE MAKING
> **Input:** Integers $a_1,\dots,a_n$ and integers m and b.
> **Question:** Are there non-negative integers $x_1,\dots,x_n$ such that $\Sigma_{i\in[n]}x_i \leq m$, and $\Sigma_{i\in[n]}x_i a_i = b$?

We can assume, without loss of generality, that $\forall i \neq j : |a_i - a_j| \geq m^3$. If this property does not hold, then we simply multiply all numbers by m^3, that is, we set a_i to be $m^3 \cdot a_i$ and set b to be $m^3 \cdot b$. It is easy to verify that this new instance is a yes-instance if and only if the original instance is a yes-instance.

We now create an equivalent BLOCK SEQUENCE ANONYMIZATION (VA) instance (B,k,t), with $t := m$ and $k := t(b+n+5t+1)$. The realizable block sequence B is the block sequence of a graph G, which is defined as follows. We introduce several gadgets, that is, subgraphs of G with distinguished vertices of specific degrees which play an important role in the correctness proof. In the following, we only specify the degrees of these *proper* vertices. To build these gadgets, we add an appropriate number of degree-one neighbors. Our construction ensures that, when k-anonymizing G by adding t vertices, the degree-one vertices will always keep their degree. The construction works as follows.

Add a *b-gadget* consisting of $5t$ *base vertices* of degree $n+t$, add b *count vertices* of degree $n+2t-1$, and add $k-b-5t$ *b-catch vertices* of degree $n+2t$. For each $i \in [n]$, add one *a_i-gadget* consisting of one *a_i-vertex* of degree $a_i+n+4t+1$ and $k-1$ *a_i-catch vertices* of degree $a_i+n+5t+1$. Finally, add a *dummy gadget* consisting of one *dummy vertex* of degree $n+4t+1$ and $k-1$ *dummy catch vertices* of degree $n+5t+1$. This completes the construction. See Figure 6.8 for an illustration.

We show that $(a_1,\dots,a_n,m,b)$ is a yes-instance of CHANGE MAKING if and only if (B,k,t) is a yes-instance of BLOCK SEQUENCE ANONYMIZATION (VA).

For the "if" direction, assume that there are integers $x_1,\dots,x_n$ such that $\Sigma_{i\in[n]}x_i \leq m$ and $\Sigma_{i\in[n]}x_i a_i = b$. We construct a k-anonymous graph G' by adding t new vertices to G as follows. For each $i \in [n]$, add x_i new *solution vertices* and connect them with a_i count vertices such that the degree of each count vertex is increased by one. (Note that this is possible since $\Sigma_{i\in[n]}x_i a_i = b$.) If $d := t - \Sigma_{i\in[n]}x_i > 0$, then add d further new *auxiliary vertices*. Connect each of the new vertices (solution and auxiliary vertices) to each *a_i-vertex*, $i \in [n]$, to the base vertices, and to the dummy vertex.

We now claim that G' is k-anonymous: First, observe that all new vertices are k-anonymous: The auxiliary vertices are of degree $n+5t+1$ together with k further

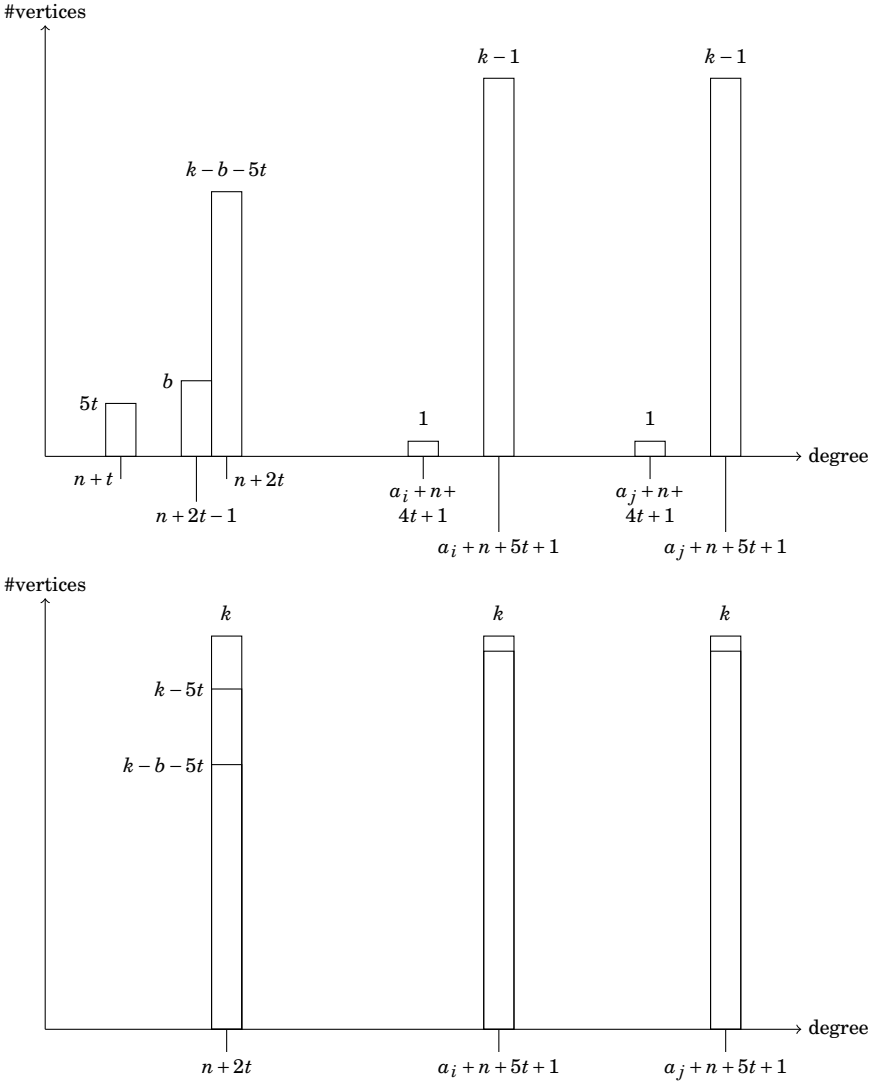


Figure 6.8. The reduction used in Theorem 6.6. The construction (corresponding to a CHANGE MAKING input) is depicted on top, while an anonymized solution (corresponding to a CHANGE MAKING solution) is depicted at the bottom. The plots show the number of vertices in each block (that is, of each degree).

vertices from the dummy gadget. Each solution vertex is of some degree $a_i + n + 5t + 1$, $i \in [n]$, together with $k - 1$ *a_i-catch vertices* and one *a_i-vertex*. Second, the original vertices are also k-anonymous: All k vertices from the dummy gadget get degree $n + 5t + 1$ since we connected the dummy vertex to all t new vertices and the degree of each dummy catch vertex remains unchanged. All k vertices from the b-gadget get degree $n + 2t$ since we connected each base vertex to all t new vertices, each count vertex with exactly one new vertex, and the degrees of the b-catch vertices remain unchanged. For each $i \in [n]$, all k vertices of the a_i-gadget get degree $a_i + n + 5t + 1$ since the degree of each *a_i-catch vertex* remains $a_i + n + 5t + 1$ and each a_i-vertex gets degree $a_i + n + 5t + 1$ since we connected it to all t new vertices. Thus, G' is k-anonymous.

For the "only if" direction, assume that there is a vertex set $V' := \{v_1, \ldots, v_{t'}\}$, $t' \leq t$, and a set of edges E' such that the graph $G' := (V \cup V', E \cup E')$ is k-anonymous. We show that $y_1, \ldots, y_n$ with $y_i = |\{v \in V' : \deg_{G'}(v) = a_i + n + 5t + 1\}|$ is a solution for $(a_1, \ldots, a_n, m, b)$. First, we show that, for each proper original vertex from V, the degree in G' is already determined by the construction of G. To this end, recall that we can only increase the degree of each original vertex by at most t. We say two original vertices are *close enough* if their degrees in G differ by at most t. Consider some proper vertex from the b-gadget, from an a_i-gadget, or from the dummy gadget, and observe that only proper vertices from the same gadget are close enough. Moreover, since, even together with t potential new vertices from V', each gadget contains less than $2k$ proper vertices, it follows that all vertices from the same gadget must end up with the same degree in G'. More precisely, all vertices from the b-gadget must have degree $n + 2t$ in G': adding V' must increase the degree of each base vertex by t to $n + 2t$, which is already the original degree of the b-catch vertices. All vertices from the dummy gadget must have degree $n + 5t + 1$ in G': adding V' must increase the degree of the dummy vertex by t to $n + 5t + 1$, which is already the original degree of the dummy catch vertices. For each $i \in [n]$, all vertices from the a_i-gadget must have degree $a_i + n + 5t + 1$ in G': adding V' must increase the degree of each a_i-vertex by t to $a_i + n + 5t + 1$, which is already the original degree of the a_i-catch vertices.

Second, observe that each new vertex from V' must have degree $a_i + n + 5t + 1$ for some $i \in [n]$ or degree $n + 5t + 1$: Clearly, as we have already seen, adding V' increases the degree of all $5t$ base vertices, of all n a_i-vertices, and of the dummy vertex by t. Thus, each new vertex has degree at least $n + 5t + 1$. Since we have at most $t < k$ new vertices, each new vertex must have the same degree in G' as some proper original vertex.

Third, observe that each (non-proper) degree-one vertex from V will keep degree one in G': only other degree-one vertices are close enough and the degree-one vertices also cannot reach the degree of any new vertex. Thus, if some original degree-one vertex has degree at least two in G', then at least k original degree-one vertices must have degree at least two in G'. Then, since $k = t(b+n+5t+1)$, at least one new vertex would have degree at least $b+n+5t+1$ in G'. This is not possible since no proper original vertex can reach this degree.

Finally, recall that $y_i, i \in [n]$, denotes the number of new vertices from V' that have degree $a_i + n + 5t + 1$ in G'. Since adding V' must increase the degree of each of the b count vertices by exactly one, there are exactly b edges between new vertices and count vertices. Furthermore, let ℓ denote the number of edges between new vertices. It holds that $b + \ell/2 = \Sigma_{i\in[n]} y_i a_i$. More precisely,

$$\frac{\ell}{2m^3} = \frac{\Sigma_{i\in[n]} y_i a_i}{m^3} - \frac{b}{m^3}$$

must be an integer since each $a_i, i \in [n]$, and b are divisible by m^3. Since $0 \leq \ell < \binom{t}{2} < m^2$, this is possible only if $\ell = 0$. Thus, there are no edges between new vertices in E' but there exist some $y_1, \ldots, y_n$ such that $\Sigma_{i\in[n]} y_i a_i = b$. Hence, $y_1, \ldots, y_n$ is indeed a solution for $(a_1, \ldots, a_n, m, b)$. □

6.5.3. Tractability Results

While it remains open whether DEGREE ANONYMIZATION (VA) is NP-hard, the weak NP-hardness result for BLOCK SEQUENCE ANONYMIZATION (VA) (Theorem 6.6) lets us conjecture that also the graph problem may be hard to solve. Hence, a parameterized approach solving DEGREE ANONYMIZATION (VA) is reasonable. Notably, we provide several (fixed-parameter) tractability results contrasting the hardness results for the constrained problem versions considered in Section 6.4.

A natural parameter to consider is the solution size t. Unfortunately, we do not know whether DEGREE ANONYMIZATION (VA) is fixed-parameter tractable with respect to t; we can only show that DEGREE ANONYMIZATION (VA) is polynomial-time solvable when t is a constant.

Theorem 6.7. DEGREE ANONYMIZATION (VA) *parameterized by the maximum number t of added vertices is* XP.

Proof. Let us denote the number of edges that a new vertex t_i has to old vertices by $d_{\text{old}}(t_i)$. We first guess t' (in $O(t)$ time), and $d_{\text{old}}(t_i)$, for each new vertex t_i

(in $O(n^t)$ time). We also guess the subgraph induced by the new vertices (in $O(2^{t^2})$ time). Then, we use a modified version of a dynamic program used by Liu and Terzi [107].

The first modification is needed since we have to satisfy the guessed degrees of the new vertices when we k-anonymize the old vertices. Specifically, we k-anonymize the combined degree sequence containing the original degrees of the old vertices, and also the guessed degrees of the new vertices, with the exception that the new vertices' degrees are fixed (that is, cannot be changed, as they are already guessed). The second modification is needed to make sure that we can realize the overall increases of degrees by using exactly the guessed degrees of the new vertices. Specifically, we maintain a vector of size t' with entries upper-bounded by the guessed $d_{\text{old}}(t_i)$, where each entry in the vector contains the number of edges already used for the corresponding new vertex (the vector is initialized to contain only zeros). These modifications give a multiplicative factor of $O(n^t)$ to the running time of the dynamic programming, resulting in an overall time complexity of $O(2^{t^2} \cdot k \cdot n^t)$. □

We can "improve" containment in XP (as shown in Theorem 6.7) to fixed-parameter tractability with respect to the combined parameter (t, Δ). Before proving the corresponding theorem, we introduce some notation and a helpful lemma.

For a set A of vertices whose addition (together with the addition of edges) transforms a graph $G = (V, E)$ into a k-anonymous graph, we call A an *addition set* and we write $G + A$ for the k-anonymous graph. Furthermore, the edges in $G + A$ having at least one endpoint in A (the "added" edges) are denoted by $E(A)$. Hence, $G + A = (V \cup A, E \cup E(A))$.

Clearly, for an addition set A of size t, all vertices in $G + A$, except those in A, have degree at most $\Delta + t$ where Δ is the largest degree in G. It may happen that the degree of some (potentially all) vertices from A in $G + A$ is larger than $\Delta + t$. In this case, there are full blocks in $G + A$ of degree larger than $\Delta + t$ consisting only of vertices from A, implying that $t \geq k$. We call blocks corresponding to degrees greater than $\Delta + t$ *large-degree blocks*. Lemma 6.1 shows that we may assume that there are at most two large-degree blocks which are, in terms of their degree values, not "too far away" from each other.

Lemma 6.1. *Let* (G, k, t) *be a yes-instance of* DEGREE ANONYMIZATION (VA). *There is an addition set* A *of size at most* t *such that in* $G + A$ *there are at most*

two large-degree blocks. Furthermore, if there are two large-degree blocks, then their degrees differ by exactly one.

Proof. Let (G,k,t) be a yes-instance of DEGREE ANONYMIZATION (VA) and let A be a corresponding addition set. Assume that there are at least two large-degree blocks B_i and B_j in $G+A$ with $|j-i|>1$. We restructure $E(A)$ to obtain a solution as desired. To this end, we introduce some notation. Let $A^L = \{a_1,\ldots,a_\ell\}$ denote the set of vertices in the large-degree blocks. Observe that $\ell \geq 2k$. For any vertex $v \in (V \cup A) \setminus A^L$, denote by $\deg^L(v)$ the number of neighbors of v in A^L, formally: $\deg^L(v) := |N_{G+A}(v) \cap A^L|$.

The idea is to restructure the graph such that each old vertex will be adjacent to the same number of new vertices, but the new vertex degrees will be closer to each other. We achieve this by a round-robin method. Specifically, the restructuring is done as follows: first, remove all edges that have at least one endpoint in A^L. Let $v_1,\ldots,v_{n+|A|-\ell}$ be an arbitrary ordering of vertices in $(V \cup A) \setminus A^L$. Then, we add edges in several steps such that in step i, vertex $v_i \in (V \cup A) \setminus A^L$ gets $\deg^L(v_i)$ incident edges, that is, overall the degree of v_i will remain unchanged. For every $a \in A^L$, denote by $\deg^L_i(a)$ the degree of a before step i. Then, let X_i denote the $\deg^L(v_i)$ vertices in A^L having the smallest value $\deg^L_i(a)$. In step i, we make v_i adjacent to all vertices in X_i. Observe that, since $\deg^L(v_i) \leq \ell$, in each step it holds for all $j,j' \in \{1,\ldots,\ell\}$ that $|\deg^L_i(a_j) - \deg^L_i(a_{j'})| \leq 1$. Hence, after the last step, there are at most two large-degree blocks B_d and B_{d+1} whose degrees differ by at most one.

It remains to ensure that $|B_d| \geq k$ and $|B_{d+1}| \geq k$. If $|B_d|$ is even, then add a perfect matching on the vertices in B_d resulting in one large block of size $\ell \geq 2k$. Otherwise, if $|B_d| < |B_{d+1}|$, then set $d' := d+1$ and add edges to introduce a Hamilton cycle in $G[B_d]$, increasing the degree of every vertex in B_d by two. If $|B_d| \geq |B_{d+1}|$, then set $d' := d$. Now, add a matching between vertices in $B_{d'}$ such that after adding these edges it holds that $0 \leq |B_{d'}| - |B_{d'+1}| \leq 2$. Observe that this is always possible as $|B_{d'}| \geq |B_{d'+1}|$. We claim that $|B_{d'}| \geq k$ and $|B_{d'+1}| \geq k$. Suppose towards a contradiction that $|B_{d'+1}| < k$. Since $|B_{d'+1}| \leq |B_{d'}| \leq |B_{d'+1}|+2$ and $|B_{d'+1}| + |B_{d'}| = \ell \geq 2k$, it follows that $\ell = 2k$, $|B_{d'+1}| = k-1$, and $|B_{d'}| = k+1$. Furthermore, as we already handled the case where $|B_d|$ was even, it follows that $|B_{d'}|$ and $|B_{d'+1}|$ are odd and k is even. Also, observe that we started with the assumption that there is a solution with at least two large blocks B_i and B_j. Since $\ell = 2k$, there are exactly two large blocks in this solution. Thus, the sum of the degrees of all vertices in A^L is $|B_i| \cdot i + |B_j| \cdot j$ which is an even number as $|B_i| = |B_j| = k$ is even. However, as $|B_{d'}|$ and $|B_{d'+1}|$ are odd, the sum of

all degrees of vertices in A^L after the restructuring is $|B_{d'}| \cdot d' + |B_{d'+1}| \cdot (d'+1)$ which is odd as either d' or $d'+1$ is odd. This is a contradiction to the fact that our restructuring did not change the degrees of vertices in $(V \cup A) \setminus A^L$ and our operations (edge deletions and insertions) do not change the parity of the sum of the degrees of the vertices in A^L. □

Using Lemma 6.1, we can now prove fixed-parameter tractability for the parameter combination (t, Δ); however, the result uses Lenstra's result about the fixed-parameter tractability of integer linear programs [104].

Theorem 6.8. DEGREE ANONYMIZATION (VA) *is fixed-parameter tractable with respect to the combined parameter* (t, Δ)*, where* t *is the number of new vertices and* Δ *is the maximum degree.*

Proof. Our algorithm consists of three phases. First (Phase I), we guess what the solution looks like, specifically guessing the degrees of the good blocks, and the degrees of the new vertices, while respecting the guessed degrees of the good blocks. Then (Phase II), we use a bottom-up "lazy" method to solve the instance for the old vertices, respecting the guessed degrees of the new vertices. Finally (Phase III), we use an integer linear program to solve the instance for the new vertices. A detailed description follows.

Phase I. We guess the subgraph induced by the new vertices (in $O(2^{t^2})$ time). We know, from Lemma 6.1, that the number of possible blocks in the solution is upper-bounded by $\Delta + t + 2 = O(\Delta + t)$. We guess the degrees of the two large-degree blocks (in $O(n+t)$ time). Then, we guess, for each block, whether it is empty or full (in $O(2^{\Delta+t})$). Finally, we guess the degree of each new vertex (in $O(\Delta+t)^t$ time). Phase I runs in $O\left((n+t) \cdot 2^{t^2} \cdot 2^{\Delta+t} \cdot (\Delta+t)^t\right) =: O((n+t) \cdot f_1(t,\Delta))$ time.

Phase II. We perform the following bottom-up lazy method. For ease of presentation, in what follows, we say that we *move* a vertex up, meaning that we connect it to some new vertices, thus changing its degree and moving it to a different block of some desired degree. Further, we can choose which new vertices to use in a round-robin way, subject to their guessed degrees; specifically, each new vertex participates in this round-robin procedure until it reaches its guessed degree.

We start with the lowest degree block, and work all the way up to the highest degree block. If the current block B_i is guessed to be empty, then we move its vertices up, to the first block above it which is guessed to become full (if there is a

gap greater than t to such a block, then we move to the next iteration). Otherwise, if it is guessed to be full, then we distinguish between the following two cases: if the number of old vertices in the block plus the number of new vertices guessed to be in this block is at least k, then we do nothing since it means that this block is already anonymized, and continue with the next block. Otherwise, B_i has a shortage of some z_i many vertices to become full, so we find the maximum $j < i$ such that the number of old vertices in B_j plus the number of new vertices guessed to be in B_j is greater than k (specifically, equals $k + z_j$ for some z_j spare vertices in B_j; if the gap $i - j$ is greater than t, then we continue with the next iteration since B_i cannot be k-anonymized). We move $\min(z_i, z_j)$ spare vertices from B_j to B_i. If, after moving these spare vertices, B_i still needs some more vertices (that is, if $z_i > z_j$), then we repeat this step once more, looking for the maximum $j' < j$ such that the number of old vertices in $B_{j'}$ plus the number of new vertices guessed to be in $B_{j'}$ is greater than k, until we have enough vertices in the current block. If at the end of this phase all of the blocks are anonymized, then we continue with the next phase. The overall time cost of Phase II is in $O(\Delta + t)^3 =: O(f_2(t, \Delta))$.

Our approach is lazy since we are performing the minimum amount of changes to make the old vertices anonymous. First, we use the spare vertices from the *closest* full block below the current one. Second, we move the minimum number of vertices to make the blocks anonymized for the old vertices, that is, we only make the bad blocks full, but never overfull (in other words, we do not introduce blocks with strictly more than k vertices in them).

Phase III. We check whether each new vertex reached its guessed degree. If so, then we return True. If there are new vertices which did not yet reach their guessed degrees, then we still have some hope of reaching these degrees (due to the laziness of Phase II) so we try to move up some more old vertices, until we reach the guessed degrees, while not destroying the anonymity of the blocks. To this end, denote the number $|B_i| - k$ of spare vertices in each full block B_i by z_i. Note that we can move any number of up to z_i vertices from this block, to any full block above it, and no other moves are possible. Now, our problem reduces to the following integer linear program.

> **Input:** Integers $\{z'_1, \ldots, z'_{n'}\}$, an $n' \times m'$ matrix $A = a_{ij}$, and an integer Z.

Task: Maximize $\sum_{i\in[n']}\sum_{j\in[m']} a_{ij}x_{ij}$ such that $\sum_{i\in[n']}\sum_{j\in[m']} a_{ij}x_{ij} \leq Z$ and $\forall j : \sum_{i\in[n']} a_{ij} \leq z_j$.

Specifically, we set n' and m' to be the number of full blocks (that is, $n' = m' \in O(\Delta + t)$). For each full block, we set z'_i to be z_i and a_{ij} to be the gap between the jth full block and the ith full block. We set Z to be equal to the overall sum of differences of guessed degrees of the new vertices and their degrees after Phase II. Note that any solution to the integer linear program is realizable as each a_{ij} is upper-bounded by the number of spare vertices z_j. Moreover, the number of variables is upper-bounded by the number of full blocks squared, that is, by $O((\Delta + t)^2)$. By a famous result of Lenstra [104], it follows that the time cost of this phase is $\text{poly}(n,t) \cdot f_3(t,\Delta)$, for some computable function f_3.

We now prove the correctness of the algorithm. As the algorithm only performs permitted operations (that is, adds up to t new vertices and connects them such that each edge is incident to at least one new vertex), it follows that if the input is a no-instance, then the algorithm returns False. Otherwise, if the input is a yes-instance, then at least one set of guesses from Phase I will be correct. Any solution must at least move the vertices that are moved in Phase II, and then the problem reduces to the integer linear program presented in Phase III. □

The question whether fixed-parameter tractability also holds for the parameter t along or for the parameter Δ alone remains open. Nevertheless, we find that fixed-parameter tractability also holds for the combined parameter (Δ, k).

Theorem 6.9. DEGREE ANONYMIZATION (VA) *is fixed-parameter tractable with respect to the combined parameter* (Δ, k)*, where* Δ *is the maximum degree and* k *is the anonymity level.*

Proof. It follows, from Theorem 6.8 that if $t \leq (\Delta k + k) \cdot (\Delta + 1)^2$, then we are done; therefore, we assume that $t > (\Delta k + k) \cdot (\Delta + 1)^2$. If $\Delta = 0$, then the input graph is already anonymized; therefore, we assume that $\Delta > 0$. If $k \geq n$, then we can solve the input instance in polynomial time by Observation 1, Proposition 6.1, and Proposition 6.2. Hence, we assume $k < n$.

Consider the following method for k-anonymizing the graph, showing that we can return True for any remaining instance. For each block B_i, if $|B_i| < \Delta k + k$, then we connect each vertex in B_i to $\Delta + 1 - i$ new vertices such that the degree of each old vertex in B_i becomes $\Delta + 1$ and the degree of each new vertex becomes one. If $|B_i| \geq \Delta k + k$, we do the same, but only for Δk arbitrarily chosen vertices in B_i.

The resulting graph has exactly two non-empty blocks: B_1 containing all of the new vertices (and only them) and $B_{\Delta+1}$ filled with all of the old vertices (and only them). Since $k < n = |B_{\Delta+1}|$, the block $B_{\Delta+1}$ is good.

Let $t' := |B_1|$ denote the number of new vertices added by our method. There are at most $\Delta+1$ blocks in the original graph and we move at most $\Delta k + k$ vertices from each block by at most $\Delta+1$. Therefore, $t' \leq (\Delta k + k) \cdot (\Delta+1)^2 \leq t$. If there are no blocks of size greater than $\Delta k + k$ in the original graph, then $t' \geq n > k$ since we moved all of the old vertices. Otherwise, if there is at least one block of size greater than $\Delta k + k$, then $t' \geq \Delta k \geq k$ since we moved at least Δk vertices from this block by at least one block. □

Contrasting the W[1]-hardness, established in Theorem 6.4, for Π-PRESERVING DEGREE ANONYMIZATION (VA) parameterized by the maximum number s of new edges, we conclude this chapter with showing fixed-parameter tractability for DEGREE ANONYMIZATION (VA) with respect to s. We again assume that s is given as part of the input.

Theorem 6.10. DEGREE ANONYMIZATION (VA) *is fixed-parameter tractable with respect to the maximum number s of newly inserted edges.*

To prove Theorem 6.10, we define the problem of anonymizing a general (not necessarily realizable) block sequence by vertex addition. Indeed, the fixed-parameter tractability of DEGREE ANONYMIZATION (VA) directly follows by providing a so-called bikernelization to this new problem (informally, a bikernelization is a kernelization to a different problem).

A *general block sequence* is a tuple $\mathscr{B} = (b_0, \ldots, b_d)$ of non-negative integers $b_i \geq 0$. We say that b_i denotes the *size* of the *block* of *degree* $i \in \{0, \ldots, d\}$ and we denote the length of $\mathscr{B}$ by $|\mathscr{B}|$. We consider tuples of integers $x = (x_0, \ldots, x_d)$, where $0 \leq x_i \leq b_i$ for all $i \in \{0, \ldots, d\}$, and as we usually think about these tuples as corresponding to the new vertices that we add in the anonymization process, we call such a tuple a *vertex*. For a tuple of integers $x = (x_0, \ldots, x_d)$, we write $x \leq \mathscr{B}$ to mean that $0 \leq x_i \leq b_i$ for all $i \in \{0, \ldots, d\}$. We define now what it means to *add* the vertex x to the general block sequence $\mathscr{B}$. Intuitively, for each $x = (x_0, \ldots, x_d)$, x_i denotes the number of degree-i vertices to which the "newly added vertex" x is connected. We denote the resulting general block sequence by $\mathscr{B} \oplus x$. Before giving the definition, we make a technical disclaimer: whenever we use an index i that is not contained in $\mathscr{B}$ (that is, $i > d$), then we implicitly assume that $\mathscr{B}$ is extended to length i by appending the corresponding number of zero entries to $\mathscr{B}$.

The general block sequence $\mathscr{B} \oplus x$ is generated out of $\mathscr{B}$ by iteratively performing the following operations, for each $i \in \{0, \ldots, d\}$: decrease b_i by x_i and increase b_{i+1} by x_i. These replacements correspond to increasing the degrees of the specified number of original vertices that are connected to the new vertex by one. Moreover, in order to insert the new vertex, let $\delta(x) := \sum_{i=0}^{|x|} x_i$ be its *degree* and increase $b_{\delta(x)}$ by one. Note that the number of *added edges* equals $\delta(x)$. For a sequence of vertices $(x^1, \ldots, x^t)$ such that $x^j \leq ((\mathscr{B} \oplus x^1)\ldots) \oplus x^{j-1}$ holds for all $1 \leq j \leq t$, we define $\mathscr{B} \oplus (x^1, \ldots, x^t) := ((\mathscr{B} \oplus x^1)\ldots) \oplus x^t$. Note that the order of adding the vertices does make a difference, as it might be that some orderings are possible while some other orderings are not. The total number of added edges then equals $\sum_{j=1}^{t} \delta(x^j)$.

We define the problem of anonymizing a general block sequence as follows.

> General Block Sequence Anonymization (VA)
> **Input:** A general block sequence $\mathscr{B}$ and $k, t, s \in \mathbb{N}$.
> **Question:** Are there vertices $x^1, \ldots, x^{t'}$, $t' \leq t$, with $\sum_{j=1}^{t'} \delta(x_j) \leq s$ such that $\mathscr{B}' := \mathscr{B} \oplus (x^1, \ldots, x^{t'})$ is k-anonymous, that is, either $b'_i \geq k$ or $b'_i = 0$ holds for each b'_i in $\mathscr{B}'$?

Note that this definition ensures that any Degree Anonymization (VA) instance (G, k, t, s) is a yes-instance if and only if $((|B_0|, \ldots, |B_{n-1}|), k, t, s)$ is a yes-instance of General Block Sequence Anonymization (VA). Therefore, in order to prove Theorem 6.10, it is sufficient to show that General Block Sequence Anonymization (VA) is fixed-parameter tractable with respect to s. To this end, we give a kernelization algorithm.

Lemma 6.2. General Block Sequence Anonymization (VA) *admits a problem kernel with respect to the maximum number s of newly inserted edges. The kernel is of size $s^{O(s)}$ and can be computed in linear time.*

To prove Lemma 6.2, we will introduce several polynomial-time data reduction rules. We give an overview of these rules now. First, we upper-bound the anonymity level k (Reduction Rule 6.1). Then, we upper-bound the maximum block size, that is $\max_i(b_i)$ (Reduction Rule 6.2). After that, we upper-bound the number t of added edges (Reduction Rule 6.3), followed by upper-bounding the number of *bad* blocks, that is, the number of blocks that have strictly less than k vertices, but also strictly more than 0 vertices (Reduction Rule 6.4). We then upper-bound the number of non-empty blocks (Reduction Rule 6.5), and finally, upper-bound the overall number of blocks (Reduction Rule 6.6). We mention here

that all our reduction rules can be easily seen to be polynomial-time computable, therefore we only prove their correctness.

We are ready to delve into the details. In order to prove the correctness of some of these reduction rules, we need the following simple lemma, which states that a sequence $(x^1,\dots,x^t)$ of vertices that can be added to a general block sequence $\mathscr{B}$ can also be added to another general block sequence $\mathscr{B}'$ of equal length if each entry in $\mathscr{B}'$ has value of at least $\sum_{j=1}^{t}\delta(x^j)$.

Lemma 6.3. *Let $\mathscr{B}=(b_0,\dots,b_d)$ be a general block sequence and let $x^1,\dots,x^t$ be a sequence of vertices such that $x^j \le ((\mathscr{B}\oplus x^1)\dots)\oplus x^{j-1}$ holds for all $1\le j\le t$. Further, let $s:=\sum_{j=1}^{t}\delta(x^j)$ and let $\mathscr{B}'=(b'_0,\dots,b'_d)$ be a general block sequence with $b'_i\ge\min\{b_i,s\}$ for all $i\in\{0,\dots,d\}$.*

Then, also $x^j \le ((\mathscr{B}'\oplus x^1)\dots)\oplus x^{j-1}$ holds for all $1\le j\le t$.

Proof. Let $I:=\{i : s\le b'_i<b_i\}$ be the set of indices where $\mathscr{B}'$ is strictly less than $\mathscr{B}$ but at least s. Note that for all other indices i, we have $b'_i\ge b_i$. Moreover, note that, for each $i\in\{0,\dots,d\}$, it holds that $\sum_{j=1}^{t}x_i^j\le s$, and thus, clearly, also $x_i^j\le s-\sum_{l=1}^{j-1}x_i^l$ holds for each $j\in\{1,\dots,t\}$.

We prove the lemma by induction on j. For $j=1$, by assumption, we have $x_i^1\le\min\{b_i,s\}\le b'_i$ for all $i\in\{0,\dots,d\}$, and thus $x^1\le\mathscr{B}'$. Now, for $j\ge 2$, let $\mathscr{B}'(j-1):=\mathscr{B}'\oplus(x^1,\dots,x^{j-1})=(b'_0(j-1),\dots,b'_q(j-1))$ for some $q\in\mathbb{N}$. This is well-defined by the inductive hypothesis. Note that for all $i\in\{0,\dots,q\}\setminus I$, we have $b'_i(j-1)\ge b_i(j-1)\ge x_i^j$. For each $i\in I$, it holds $b'_i(j-1)\ge s-\sum_{l=1}^{j-1}x_i^l\ge x_i^j$. Hence, $x^j\le\mathscr{B}'(j-1)$. □

Let us now specify the reduction rules. The first reduction rule upper-bounds the degree k of anonymity linearly in s.

Reduction Rule 6.1. *Transform an instance $((b_0,\dots,b_d),k,t,s)$ of* General Block Sequence Anonymization (VA) *to the instance $((b'_0,\dots,b'_d),k',t,s)$, where $k':=\min\{k,2s+1\}$ and, for $i\in\{0,\dots,d\}$, set:*

$$b'_i:=\begin{cases}b_i-(k-k'), & b_i\ge k-s,\\ b_i, & \textit{else}.\end{cases}$$

Lemma 6.4. *Reduction Rule 6.1 is correct.*

The proof of Lemma 6.4 is deferred to the appendix to this chapter.

The next reduction rule upper-bounds the maximum block size by $k+s$.

Reduction Rule 6.2. *Transform an instance* $((b_0,\ldots,b_d),k,t,s)$ *of* General Block Sequence Anonymization (VA) *to the instance* $((b'_0,\ldots,b'_d),k,t,s)$*, where, for* $i \in \{0,\ldots,d\}$*, set* $b'_i = \min\{k+s, b_i\}$.

Lemma 6.5. *Reduction Rule 6.2 is correct.*

The proof of Reduction Rule 6.2 is deferred to the appendix to this chapter.

Next, we upper-bound the number t of new vertices in s and k.

Reduction Rule 6.3. *Transform an instance* $(\mathscr{B},k,t,s)$ *of* General Block Sequence Anonymization (VA) *to the instance* $(\mathscr{B},k,\min\{t,k+2s\},s)$.

Lemma 6.6. *Reduction Rule 6.3 is correct.*

The proof of Reduction Rule 6.3 is deferred to the appendix to this chapter.

The next two reduction rules upper-bound the number of non-zero entries in the block sequence. First, we identify those blocks whose sizes need to be changed. Recall that a block is said to be *bad* if it contains less than k but more than zero vertices.

The general idea for these two reduction rules is to keep all blocks of degree at most $2s$ (since new vertices may end up in the first s blocks and an already existing vertex with degree at most s can reach degree at most $2s$), the bad blocks (since we have to fix them), some good blocks close to the bad blocks (to allow movement of vertices to or from bad blocks), and also some further good blocks (to set the correct degrees of the newly added vertices).

Our next reduction rule transforms the instance into a trivial no-instance if there are more bad blocks than one could fix by adding at most s edges. Each new edge introduced by a new vertex x with $x_i > 1$ can fix at most three bad blocks, namely b_i, b_{i+1}, and $b_{\delta(x)}$. Note that other block sizes are not affected by the edges corresponding to x_i.

Reduction Rule 6.4. *Let* $(\mathscr{B},k,t,s)$ *be an instance of* General Block Sequence Anonymization (VA) *with* $\mathscr{B} = (b_0,\ldots,b_d)$*. If* $\mathscr{B}$ *contains more than* $3s$ *entries* b_i *with* $0 < b_i < k$*, then return a trivial no-instance.*

The correctness proof of Reduction Rule 6.4 can be easily seen, and thus, omitted.

In order to upper-bound the number of good blocks, the decisive observation is that adding two edges to good blocks cannot be considered to be independent. We first explain the intuition behind.

For example, adding an edge to a vertex from a degree-i block of size k may only be possible if one also adds an edge to a vertex from a degree-$(i-1)$ block—one vertex moves from the degree-i block to the degree degree-$(i+1)$ block (causing it to be momentarily bad) and one moves from the degree-$(i-1)$ block to the degree-i block and block i ends up with size k.

The idea now is to consider consecutive blocks where operations on one block have influence on operations on the next block. Fortunately, the number of operations influencing each other is upper-bounded by the total number s of added edges. More formally, we use the concepts of "scope" and "chain". Let $\mathscr{B} = (b_0, \ldots, b_d)$ be a general block sequence. The *scope* of a position $z > s$ in $\mathscr{B}$ is the sequence of positions $(z-s, \ldots, z)$. The *scope fingerprint* F_z of z is the subsequence $F_z := (b_{z-s}, \ldots, b_z)$. Let $(x^1, \ldots, x^t)$ be a sequence of vertices. A *chain* with respect to $(x^1, \ldots, x^t)$ is a pair (y,z) of positive integers with $y < z$ such that $\forall i \in \{y, \ldots, z-1\} : \exists x^j : x^j_i > 0$ (that is, every degree in the chain is moved). A chain (y,z) is maximal if $(y-1,z)$ and $(y,z+1)$ are no chains.

Note that a chain has length at most s, that is, $z - y \leq s$, since the total number of added edges is at most s. That is, every (maximal) chain (y,z) is "fully contained" in the scope of z.

We are now ready to formulate our most technical reduction rule which upper-bounds the number of good blocks which are not empty. It will iteratively mark positions in $\mathscr{B}$ corresponding to blocks that have to be kept and finally set the entries in $\mathscr{B}$ at all non-marked positions to zero.

Reduction Rule 6.5. *Let* $(\mathscr{B}, k, t, s)$ *be an instance of* General Block Sequence Anonymization (VA) *with* $\mathscr{B} = (b_0, \ldots, b_d)$.

1) *Mark the positions* $0, \ldots, 2s$.
2) *Mark all positions* i *with* $\exists z : |z - i| \leq s \wedge 0 < b_z < k$.
3) *Iteratively do the following starting with* $j := 0$.
   a) *Find the next non-marked position* $z > j$.
   b) *Compute the scope fingerprint* F_z *and set* $j := z$.
   c) *If* F_z *has been computed before less than* $2s^2 + s$ *times, then mark the positions* $z-s, \ldots, z$.
4) *For each* $0 \leq i \leq d$, *set* $b'_i := b_i$ *if position* i *is marked, and set* $b'_i := 0$ *otherwise.*

Return $(\mathscr{B}' = (b'_0, \ldots, b'_d), k, t, s)$.

Lemma 6.7. *Reduction Rule 6.5 is correct.*

The proof of Lemma 6.7 is deferred to the appendix to this chapter.

It remains to upper-bound the largest degree by some function in s. To this end, observe the following with respect to high-degree blocks. First, by adding at most s edges, no new vertex can end up in a block of vertices with degree larger than s. Second, by adding some vertices and at most s edges, we cannot decrease the degree of any original vertex and we can only increase the degree of an original vertex by at most s.

Based on the observations above, we introduce the concept of "high-degree large gaps" as follows. Let $(\mathscr{B} = (b_0, \ldots, b_d), k, t, s)$ be an instance of GENERAL BLOCK SEQUENCE ANONYMIZATION (VA). We say that a pair of positive integers (ℓ, r) describes a *high-degree large gap* of $\mathscr{B}$ if

1) $s < \ell < r$,
2) $r - \ell > s$, and
3) $\forall i \in \{\ell, \ldots, r\} : b_i = 0$.

Our final reduction rule shrinks the high-degree large gaps in a general block sequence.

Reduction Rule 6.6. *Transform an instance* $(\mathscr{B}, k, t, s)$ *of* GENERAL BLOCK SEQUENCE ANONYMIZATION (VA) *with some high-degree large gap* (ℓ, r) *in* $\mathscr{B}$*, to the instance* $(\mathscr{B}', k, t, s)$*, where* $\mathscr{B}'$ *is constructed from* $\mathscr{B}$ *by removing the entries* $b_{\ell+s}, \ldots, b_r$.

Lemma 6.8. *Reduction Rule 6.6 is correct.*

The proof of Lemma 6.8 is deferred to the appendix to this chapter.

For proving Lemma 6.2, it remains to show that the above reduction rules indeed yield a problem kernel with respect to s.

Proof (of Lemma 6.2). Let $I := (\mathscr{B}, k, t, s)$ be an arbitrary GENERAL BLOCK SEQUENCE ANONYMIZATION (VA) instance. Our kernelization algorithm first applies to I Reduction Rules 6.1, 6.2, and 6.3, in that order. Let $I' := (\mathscr{B}', k', t', s)$ be

the instance achieved after that. Clearly, we have $k' \in O(s)$, $\max_i b'_i \leq k' + s \in O(s)$, and also $t' \leq k' + 2s \in O(s)$. Thus, all numbers in I' are upper-bounded by s. In order to upper-bound the maximum degree in s, we apply Reduction Rules 6.4 and 6.5 once, and then apply Reduction Rule 6.6 exhaustively to I'. We denote the resulting instance by $I'' := (\mathscr{B}'', k', t', s)$. After application of Reduction Rule 6.4, I'' either is a constant-size no-instance, or $\mathscr{B}''$ contains at most $3s$ bad blocks. Now, consider the number of marked positions in each step of Reduction Rule 6.5. In Step 1, we mark $2s+1$ positions. Step 2 marks at most $3s(2s+1)$ positions, whereas in Step 3 we mark at most $(2s^2+s)s$ positions for each possible scope fingerprint. The number of possible scope fingerprints is $(\max_i b'_i + 1)^s \in s^{O(s)}$. Thus, the total number of non-zero blocks in $\mathscr{B}''$ is bounded by a function in s.

Finally, after exhaustive application of Reduction Rule 6.6 there are no more large-degree gaps in $\mathscr{B}''$, hence, the number of degree-zero blocks in $\mathscr{B}''$ is at most s times the number of non-zero blocks, which is again bounded in s. The correctness is guaranteed by Lemmas 6.4 to 6.8. Clearly, this process can be done in linear time since each reduction rule is applied at most a linear number of times and runs in linear time. The size of the kernel is governed by the number of possible chains, therefore upper-bounded by $s^{O(s)}$. Finally, since each reduction rule can be carried-out in polynomial time, it follows that the kernelization can be performed in polynomial time. □

6.6. Outlook

We state some ideas for future research.

- This chapter provides a step towards properties-preserving degree anonymization, that is, anonymizing a graph while preserving some of its properties. The set of properties considered in this chapter (that is, connectivity, pairwise distances, and diameter) is certainly not an exhaustive set of properties one might try to preserve.

  Specifically, there are other properties which could be more natural to consider in the context of social networks. As two immediate examples we mention the density and the clustering coefficient; the computational complexity of Π-PRESERVING DEGREE ANONYMIZATION (VA) with respect to these properties is unexplored so far.

- It is not clear what are the practical consequences of the algorithmic results presented in this chapter. We believe that, by carefully implementing the algorithms presented in the proofs of Theorem 6.8, Theorem 6.9, and Theorem 6.10, these algorithms might prove to be quite efficient for real-world instances.
- It turns out that most of the cases considered in this chapter are intractable, and even some of the positive algorithmic results obtained in this chapter are of high complexity. Thus, a deeper investigation of approximation algorithms may be beneficial, possibly following the route taken by Chester et al. [44] and by Chester et al. [45]. We refer the reader to Section 7.7 for a more elaborate discussion of approximation algorithms for degree anonymization problems.
- Finally, and naturally, studying more ways of anonymizing graphs by performing different graph modification operations (as done in Chapter 7, for example; there, for graph contractions) is of interest.

Appendix for Chapter 6

We provide proofs missing from Chapter 6.

6.A. Proof of Theorem 6.3

Theorem 6.3. *For* $\Pi \in \{$*Distances, Diameter, Connectivity*$\}$, Π-PRESERVING DEGREE ANONYMIZATION (VA) *is* NP*-hard and also* W[2]*-hard with respect to the maximum number* t *of added vertices, even if the anonymity level* k *is* 2*. For* $\Pi \in \{$*Distances, Connectivity*$\}$*, this is also true on trees.*

Proof. The proof is based on extending ideas from the proof of Theorem 6.2. We provide a reduction from the W[2]-complete SET COVER problem parameterized by the solution size [57].

> SET COVER
> **Input:** A universe of elements X, a collection $\mathscr{S}$ of sets of elements of X, and a budget h.
> **Question:** Is there a subcollection $\mathscr{S}' \subseteq \mathscr{S}$ of sets such that $|\mathscr{S}'| \leq h$ and $\bigcup_{S \in \mathscr{S}'} S = X$?

Given a SET COVER instance $I = (\mathscr{S}, X, h)$, we construct a Π-PRESERVING DEGREE ANONYMIZATION (VA) instance as follows. First, we set $k = 2$ and $t = h$. Next, for notational convenience, we define a helper function $f(i,z) := (t+1)n + (z-1)n(t+2) + i(t+2)$. Now we construct a graph $G = (V,E)$ as follows. For each element x_i, we define an *element gadget* containing, for each $z \in [t+1]$, a star $K_{1,f(i,z)}$ with center vertex $v_{i,z}$. For technical reasons, we further add an *element safety gadget* for each element x_i, containing two stars $K_{1,f(i,z)-1}$ for each $z \in [t+1]$. We also define, for each set S_j, a *set gadget* containing, for each element $x_i \in S_j$ and for each $z \in [t+1]$, a star $K_{1,f(i,z)-1}$ (its center denoted $v_{j,i,z}$). The general idea is that the element gadgets can be anonymized by connecting the set gadgets to some new vertices such that each new vertex corresponds to a set which contains the element corresponding to the element gadget. For the

new vertices, in order to be k-anonymous, we add a *safety gadget* consisting of k stars $K_{1,d}$ for each $d \in [(t+1)n]$.

The various gadgets are connected differently, depending on the property Π. The idea is to ensure that no new vertex can be connected to element gadgets corresponding to elements of different sets, by forcing such a connection to violate the property. For $\Pi \in$ {Distances, Diameter, Connectivity} and a fixed S_j, we introduce a new vertex v_j and connect it to $v_{j,i,z}$ for each $x_i \in S_j$ and each $z \in [t+1]$. We further add degree-one vertices connected to v_j to fill up its degree to $f(n, t+2)$.

For $\Pi \in$ {Distances, Diameter}, we introduce another new vertex u, and connect u to all of the v_j. Again, we fill up its degree to $f(n, t+2)$ by connecting new degree-one vertices to u.

For Π=Diameter, we add two new vertices v'_1 and v'_m. We connect v'_1 to v_1 and v'_m to v_m by a path of length 5. For each $j \in [m-1]$, we connect v_j to v_{j+1} by a path of length $t+8$. We connect u to each v_j by a path of length $\lceil ((t+8)m+1)/2 \rceil$.

It remains to show that I is a yes-instance if and only if (G, k, t) is a yes-instance.

For the "if" direction, given a set cover $\mathscr{S}'$, we add one new vertex w_j for each $S_j \in \mathscr{S}'$ and connect it to all of the $v_{j,i,z}$ for each $x_i \in S_j$ and each $z \in [t+1]$. We next show that this operation is permitted, as we do not violate Π. Indeed, consider any S_j together with any pair of elements $x_{i_1}, x_{i_2} \in S_j$ and any pair $z_1, z_2 \in [t+1]$: for Π=Connectivity, it holds that v_{j,i_1,z_1} and v_{j,i_2,z_2} were already connected; for $\Pi \in$ {Distances, Diameter}, it holds that v_{j,i_1,z_1} and v_{j,i_2,z_2} already had distance two, therefore the shortest path of no pair of vertices has been changed. Furthermore, the maximum distance in the new graph is still attained by the distance between v'_1 to v'_m, therefore the diameter did not change as well.

We now show that the graph is k-anonymous. First, note that the only vertices that need to be anonymized are the $v_{i,z}$. Since $\mathscr{S}'$ is a set cover, for each i and for each z, the degree of at least one $v_{j,i,z}$ is incremented by one, thus anonymizing the corresponding $v_{i,z}$. Also, the element safety gadgets ensure that we introduced no new bad blocks, and the safety gadget ensures that the new vertices are anonymized.

For the "only if" direction, consider a new vertex w, two different sets $S_{j_1}, S_{j_2} \in \mathscr{S}$ and $z_1, z_2 \in [t+1]$. We show that v_{j_1,i_1,z_1} and v_{j_2,i_2,z_2} cannot both be connected to w: for Π=Connectivity, this is true since they are not connected; for $\Pi = Distances$, this is true since the distance between them is four, and would change to two; for Π=Diameter, this is true since this would decrease the diameter (specifically, the distance between v'_1 and v'_m) by more than t, and also creating a shortcut between the paths of length $t+8$ does not help.

Now, assume that (G,k,t) is a yes-instance. As only t new vertices are added, there exists an integer $z' \in [t+1]$ such that no new vertex has the same degree as $v_{i,z'}$ for any $x_i \in X$. Note that there is a gap of $t+1$ above each $v_{i,z'}$. Hence, it must be anonymized from below. The only possible vertices to use are the $v_{j,i,z'}$ and the centers of the stars in the element safety gadgets. We can assume, without loss of generality, that every element x_i is contained in at least one set S_j (as we can easily check if this is not the case, and return True). Therefore, if some vertex from an element safety gadget is used, we can always use another $v_{j,i,z'}$ to anonymize $v_{i,z'}$. We can then associate a unique set S_j with every newly added vertex, and since all of the $v_{i,z'}$ are anonymized, we get a set cover. □

6.B. Proofs of Reduction Rules for Lemma 6.2

We provide proofs of correctness for the reduction rules used in Lemma 6.2.

6.B.1. Proof of Lemma 6.4

Lemma 6.4. *Reduction Rule 6.1 is correct.*

Proof. The general intuition behind this rule is that if k is very large (with respect to s), then any small block must be somehow "fixed" (either by moving some vertices into it, or by moving all of the vertices in it to some other blocks). Therefore, for these blocks, the actual k could be just slightly larger than their sizes. Moreover, for any large block, not all of the vertices of this block can be moved to another block. Therefore, k and their sizes can be decreased.

More formally, we have to show that $I := (\mathscr{B} = (b_0,\dots,b_d),k,t,s)$ is a yes-instance if and only if $I' := (\mathscr{B}' = (b'_0,\dots,b'_d),k',t,s)$ is a yes-instance. Clearly, this is true for $k' = k$ since this implies $I = I'$. Thus, we can assume that $k' = 2s+1 < k$. Therefore, $b'_i \geq \min\{b_i, k-s-(k-k')\} = \min\{b_i,k'-s\} \geq \min\{b_i,s\}$ holds for each $i \in \{0,\dots,d\}$. Let $(x^1,\dots,x^{t'})$, for $t' \leq t$, be a sequence of vertices with $\sum_{j=1}^{t'} \delta(x^j) \leq s$ such that

$$\mathscr{B}(t') := \mathscr{B} \oplus (x^1,\dots,x^{t'}) = (b_0(t'),\dots,b_q(t'))$$

is well-defined. Then, by Lemma 6.3, also

$$\mathscr{B}'(t') := \mathscr{B}' \oplus (x^1,\dots,x^{t'}) = (b'_0(t'),\dots,b'_q(t'))$$

is well-defined. Now, for $i \leq q$ such that $i > d$, we have $b'_i(t') = b_i(t')$ since these values only depend on $(x^1, \ldots, x^{t'})$. Also, for all $i \leq d$ with $b_i = b'_i$, it clearly holds that $b'_i(t') = b_i(t')$. Finally, for the remaining indices i, we have $b_i(t') > 0$ since $b_i \geq k - s > s + 1$, and also $b'_i(t') > 0$ since $b'_i \geq k - s - (k - k') = k' - s = s + 1$. Moreover, it holds that:

$$b'_i(t') - b'_i = b_i(t') - b_i \iff$$

$$b'_i(t') - (b_i - (k - k')) = b_i(t') - b_i \iff$$

$$b'_i(t') - k' = b_i(t') - k.$$

Hence, $b'_i(t') \geq k'$ if and only if $b_i(t') \geq k$. Consequently, $\mathscr{B}(t')$ is k-anonymous if and only if $\mathscr{B}'(t')$ is k'-anonymous showing that I is a yes-instance if and only if I' is a yes-instance. □

6.B.2. Proof of Lemma 6.5

Lemma 6.5. *Reduction Rule 6.2 is correct.*

Proof. The general idea of this rule is that the size of any very large block can be decreased. The basic reason is that we cannot use more than s vertices of any block, including these large blocks.

More formally, let $I := (\mathscr{B}, k, t, s)$ be an input instance and let $I' := (\mathscr{B}', k, t, s)$ denote the transformed instance. Note that $b'_i \geq \min\{b_i, s\}$ for each $i \in \{0, \ldots, d\}$. Let $(x^1, \ldots, x^{t'})$, for $t' \leq t$, be a sequence of vertices with $\sum_{j=1}^{t'} \delta(x^j) \leq s$ such that

$$\mathscr{B}(t') := \mathscr{B} \oplus (x^1, \ldots, x^{t'}) = (b_0(t'), \ldots, b_q(t'))$$

is well-defined. Then, again, by Lemma 6.3, also

$$\mathscr{B}'(t') := \mathscr{B}' \oplus (x^1, \ldots, x^{t'}) = (b'_0(t'), \ldots, b'_q(t'))$$

is well-defined. Clearly, for any i with $b_i > k + s$, we have $b_i(t') > k$ and also $b'_i(t') \geq k$ since $b'_i = k + s$. For all other i, it holds $b'_i(t') = b_i(t')$. Hence, I is a yes-instance if and only if I' is a yes-instance. □

6.B.3. Proof of Lemma 6.6

Lemma 6.6. *Reduction Rule 6.3 is correct.*

Proof. The general idea of this rule is that in any solution which uses a lot of new vertices, most of these new vertices must be isolated (that is, of degree 0). However, a solution does not need a lot of new isolated vertices, therefore, we can reduce t.

More formally, let $t^* := \min\{t, k+2s\}$. For $t^* = t$, there is nothing to show. Thus, assume that $t^* = k+2s < t$. If $I := (\mathscr{B}, k, t, s)$ is a no-instance, then, clearly, also $I' := (\mathscr{B}, k, t^*, s)$ is a no-instance since $t^* < t$. Thus, let I be a yes-instance and let $t' = t_0 + t_1 \leq t$ be the number of newly added vertices, where t_0 denotes the number of added degree-zero vertices and t_1 denotes the number of added vertices of degree at least one (by saying the degree of the vertex we mean, naturally, the total sum of the elements in the vertex vector). Let $(x^1, \ldots, x^{t'})$ be the added vertices with $\sum_{j=1}^{t'} \delta(x^j) \leq s$ and note that we can assume that the t_0 first vertices $x^1 = \ldots = x^{t_0} = (0, \ldots, 0)$ are the degree-zero vertices. Moreover, note that $t_1 \leq s$ holds. Let $\mu := \min\{t_0, k+s\}$. We show that adding the $\mu + t_1 \leq k+s+s = t^*$ vertices $(x^1, \ldots, x^\mu, x^{t_0+1}, \ldots, x^{t'})$ is a solution for I'. This is clearly true for $\mu = t_0$, hence we can assume that $\mu = k+s < t_0$. First, we have to show that $\mathscr{B} \oplus (x^1, \ldots, x^\mu, x^{t_0+1}, \ldots, x^{t'})$ is well-defined. Trivially, $\mathscr{B} \oplus (x^1, \ldots, x^\mu)$ is well-defined. Let $\mathscr{B}(\mu) := \mathscr{B} \oplus (x^1, \ldots, x^\mu)$ and let $\mathscr{B}(t_0) := \mathscr{B} \oplus (x^1, \ldots, x^{t_0})$. Note that we have $b_0(\mu) = b_0 + \mu \geq s$ and, for all $i > 0$, we have $b_i(\mu) = b_i(t_0) = b_i$. Thus, since by assumption $\mathscr{B}(t') := \mathscr{B}(t_0) \oplus (x^{t_0+1}, \ldots, x^{t'})$ is well-defined, it follows from Lemma 6.3 that also $\mathscr{B}^* := \mathscr{B}(\mu) \oplus (x^{t_0+1}, \ldots, x^{t'})$ is well-defined. Note also that the total number of added edges does not change.

It remains to show that $\mathscr{B}^*$ is k-anonymous. This is true since $b_i^* = b_i(t')$ holds for each $i > 0$, and $b_0^* \geq k$ holds since $b_0(\mu) \geq k+s$. □

6.B.4. Proof of Lemma 6.7

Lemma 6.7. *Reduction Rule 6.5 is correct.*

Proof. The general idea of this rule is that, besides the first $2s$ blocks (which are important because the old vertices and the new vertices can reside in them at the end), and besides the chains corresponding to bad blocks, all other chains can be used only in order to let the new vertices achieve some desired degree. As we have only a limited number of such possible chains, and as they are used only by the new vertices, we can upper-bound the number of such useful chains.

More formally, first, assume that there is a solution $(\bar{x}^1, \ldots, \bar{x}^{t'})$, $t' \leq t$, such that $\mathscr{B}' \oplus (\bar{x}^1, \ldots, \bar{x}^{t'})$ is k-anonymous. It is easy to verify that $\mathscr{B} \oplus (\bar{x}^1, \ldots, \bar{x}^{t'})$ is also k-anonymous.

Second, assume that there is a solution $(x^1,\dots,x^{t'})$, $t' \le t$, such that $\mathscr{B} \oplus (x^1,\dots,x^{t'})$ is k-anonymous. We show how to adjust $(x^1,\dots,x^{t'})$ to obtain a solution $(\bar{x}^1,\dots,\bar{x}^{t'})$ such that $\mathscr{B}' \oplus (\bar{x}^1,\dots,\bar{x}^{t'})$ is k-anonymous. Intuitively, we show that every maximal chain (y,z) with respect to $(x^1,\dots,x^{t'})$ can either be realized equivalently with respect to $(\bar{x}^1,\dots,\bar{x}^{t'})$ or else we show a replacement ensuring k-anonymity.

We construct $(\bar{x}^1,\dots,\bar{x}^{t'})$ as follows. First, we initialize $\bar{x}^j = (0,\dots,0)$ of the same length as x^j for each $1 \le j \le t'$. For each maximal chain (y,z) with respect to $(x^1,\dots,x^{t'})$, we distinguish two cases:

Case 1. $\forall i \in \{y,\dots,z\} : b_i = b'_i$. In this case, we set $\bar{x}^j_i := x^j_i$ for all $j \in \{1,\dots,t'\}$, $i \in \{y,\dots,z\}$, that is, this chain can be realized equivalently.

Case 2. $\exists i \in \{y,\dots,z\} : b_i \ge k$ but $b'_i = 0$. This case is only possible since not all positions between y and z have been marked by the reduction rule. Hence, the scope fingerprint F_z of z must have been computed more than $2s^2+s$ times before.

In the following, we say that the scope of z is *touched* by a sequence of vertices $(x^1,\dots,x^{t'})$ if $\exists j \in \{1,\dots,t'\} : \exists i \in \{z-s,\dots,z\} : x^j_i > 0$.

We show that there is at least one position z' with fingerprint $F_{z'} = F_z$ whose scope is neither touched by $(x^1,\dots,x^{t'})$ nor by $(\bar{x}^1,\dots,\bar{x}^{t'})$ as constructed so far. Note that each edge touches at most $s+1$ scopes (as each scope is basically an interval of size s). Moreover, there are at most s edges in $(x^1,\dots,x^t)$ and, until now, at most $s-1$ (possibly different) edges in $(\bar{x}^1,\dots,\bar{x}^t)$ have been introduced. Altogether, at most $(2s-1)\cdot(s+1) = 2s^2+s-1$ scopes (with any fingerprint) have been touched. In particular, there is one scope $(z'-s,\dots,z')$ with scope fingerprint F_z that is not touched so far by $(\bar{x}^1,\dots,\bar{x}^t)$ and that is also not touched by $(x^1,\dots,x^t)$.

Finally, we use the scope of z' to realize (y,z) by setting $\bar{x}^j_{i-z+z'} := x^j_i$ for all $j \in \{1,\dots,t'\}$, $i \in \{y,\dots,z\}$.

By construction of $(\bar{x}^1,\dots,\bar{x}^{t'})$, the number of newly added edges is:

$$\sum_{j=1}^{t'} \delta(\bar{x}^j) = \sum_{j=1}^{t'} \delta(x^j).$$

More precisely, it even holds that the degrees of the newly introduced vertices remain unchanged, that is:

$$\forall j \in \{1,\dots,t\} : \delta(\bar{x}^j) = \delta(x^j).$$

Now, let $\hat{\mathscr{B}} := (\hat{b}_0, \ldots, \hat{b}_d) = \mathscr{B}' \oplus (\bar{x}^1, \ldots, \bar{x}^{t'})$ and $\check{\mathscr{B}} := (\check{b}_0, \ldots, \check{b}_d) = \mathscr{B} \oplus (x^1, \ldots, x^{t'})$. It remains to show that $\hat{\mathscr{B}}$ is k-anonymous. To this end, consider an arbitrary index h in $\hat{\mathscr{B}}$.

Case 1. h is neither in a maximal chain with respect to $(x^1, \ldots, x^{t'})$ nor with respect to $(\bar{x}^1, \ldots, \bar{x}^{t'})$. By construction of $(\bar{x}^1, \ldots, \bar{x}^{t'})$, this solution introduces the same number of new vertices with degree h as the solution $(x^1, \ldots, x^{t'})$ does. Thus, $\hat{b}_h = \check{b}_h$ and $(\check{b}_h = 0) \vee (\check{b}_h \geq k)$.

Case 2. h is in a maximal chain (y, z) with respect to $(x^1, \ldots, x^{t'})$, and in a maximal chain (y', z') with respect to $(\bar{x}^1, \ldots, \bar{x}^{t'})$. By construction of $(\bar{x}^1, \ldots, \bar{x}^{t'})$, it holds that $y' = y$, $z' = z$, and $\forall j \in \{1, \ldots, t'\} : \forall i \in \{y, \ldots, z\} : \bar{x}_i^j := x_i^j$. Furthermore, $(\bar{x}^1, \ldots, \bar{x}^{t'})$ introduces the same number of new vertices with degree h as $(x^1, \ldots, x^{t'})$ does. Hence, $\hat{b}_h = \check{b}_h$ and $(\check{b}_h = 0) \vee (\check{b}_h \geq k)$.

Case 3. h is in a maximal chain (y, z) with respect to $(x^1, \ldots, x^{t'})$, but not in a maximal chain with respect to $(\bar{x}^1, \ldots, \bar{x}^{t'})$. In this case, we claim that $\delta(x^j) \neq h$ (and thus, also $\delta(\bar{x}^j) \neq h$) holds for all $j \in \{1 \ldots, t'\}$, and prove this as follows. Assume towards a contradiction that there is a new vertex x^j with degree $\delta(x^j) = h$. Then, $h \leq s$. However, all positions up to $2s$ have been marked in Step 1 of the reduction rule. In particular, all positions between y and z have been marked and, hence, by construction of $(\bar{x}^1, \ldots, \bar{x}^{t'})$ the pair (y, z) would also be a maximal chain with respect to $(\bar{x}^1, \ldots, \bar{x}^{t'})$. This contradicts the assumption of Case 3, and hence $z > 2s$ and $h > s$, and there is no new vertex with degree h. Analogously, h does not correspond to a bad block since then all positions between y and z would have been marked in Step 2 of the reduction. Thus, $\hat{b}_h = b'_h$ and $(b'_h = 0) \vee (b'_h \geq k)$.

Case 4. h is not in a maximal chain with respect to $(x^1, \ldots, x^{t'})$, but in a maximal chain $(\bar{y}, \bar{z})$ with respect to $(\bar{x}^1, \ldots, \bar{x}^{t'})$. This is only possible if there is a position z with the same scope fingerprint as z' such that $(z - (\bar{z} - \bar{y}), z)$ is a maximal chain with respect to $(x^1, \ldots, x^{t'})$ and:

$$\forall j \in \{1, \ldots, t'\} : \forall i \in \{\bar{y}, \ldots, \bar{z}\} : \bar{x}_i^j := x_{i-\bar{z}+z}^j.$$

Furthermore, neither $(x^1, \ldots, x^{t'})$ nor $(\bar{x}^1, \ldots, \bar{x}^{t'})$ introduce new vertices with degree h since all positions up to $2s$ have been marked in Step 1 of the reduction, and hence, $z > 2s$. That is, $\hat{b}_h = \check{b}_{h-\bar{z}+z}$ and $(\check{b}_{h-\bar{z}+z} = 0) \vee (\check{b}_{h-\bar{z}+z} \geq k)$.

Hence, $\hat{\mathscr{B}}$ is indeed k-anonymous. □

6.B.5. Proof of Lemma 6.8

Lemma 6.8. *Reduction Rule 6.6 is correct.*

Proof. The general idea of this rule is that the size of any such high-degree large gap can be reduced, as no vertex can cross-over this gap.

More formally, we first observe that the entries $b_{\ell+s},\ldots,b_r$ are all 0-entries in $\mathscr{B}$. Moreover, in any $\mathscr{B}^j := \mathscr{B} \oplus (x^1,\ldots,x^j)$, inserting at most s edges to $\mathscr{B}$, these entries $b_{\ell+s},\ldots,b_r$ are all 0-entries.

First, assume that there is a solution $(x^1,\ldots,x^{t'})$, $t' \leq t$, such that $\mathscr{B} \oplus (x^1,\ldots,x^{t'})$ is k-anonymous. Then, obtain $\bar{x}^j$ from x^j by removing the entries $x^j_{\ell+s},\ldots,x^j_r$ for each $1 \leq j \leq t'$. (Note that these entries must be 0-entries.) It is easy to verify that $\mathscr{B}' \oplus (\bar{x}^1,\ldots,\bar{x}^{t'})$ is k-anonymous.

Second, assume that there is a solution $(\bar{x}^1,\ldots,\bar{x}^{t'})$ such that $\mathscr{B}' \oplus (\bar{x}^1,\ldots,\bar{x}^{t'})$ is k-anonymous. Then, obtain x^j from $\bar{x}^j$ by inserting $r-\ell-s$ 0-entries between $\bar{x}^j_{\ell-1}$ and $\bar{x}^j_\ell$. It is easy to verify that $\mathscr{B} \oplus (x^1,\ldots,x^{t'})$ is k-anonymous. □

7. Degree Anonymization by Graph Contractions

In this chapter, similarly to Chapter 6, we study the computational complexity of k-anonymizing a given graph by performing as few graph modification operations as possible (recall that an undirected graph is k-anonymous if, for every vertex in it, there are at least $k-1$ other vertices with the same degree).

In this chapter, however, we do not allow adding new vertices (as we do in Chapter 6), but we allow to perform graph contractions (for example, edge contractions). Graph contractions are natural operations in the context of social networks and they are studied, for example, in the context of clustering algorithms. Moreover, some graphs can be anonymized more efficiently (in terms of the number of operations performed) by performing graph contractions than by adding vertices (a simple example for this phenomenon is given in Figure 7.6).

We show that the problem of degree anonymization by graph contractions is NP-hard even for some very restricted inputs, and, in general, it seems to be computationally harder than the problem of degree anonymization by adding vertices (studied in Chapter 6). Finally, we identify some fixed-parameter tractable cases.

7.1. Illustrating Example

Consider the following example.

Example 15. Recall the group of people discussed in Section 3.1 and the social network representing the friendship relationships between them (Figure 3.1, and given also in Figure 7.1 for convenience).

The corresponding example in Section 6.1 considers anonymizing the social network by adding new vertices. Here, however, corresponding to the computational problem considered in this chapter, we do not allow vertex additions, but we allow graph contractions. Consider the obfuscated social network described in the corresponding example in Section 6.1, which is given here, in Figure 7.2, for

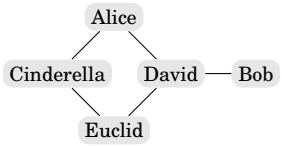


Figure 7.1. The social network used in the illustrating example.

completeness (for example, Euclid's name has been obfuscated to be "E", and his real age (27) is revealed).

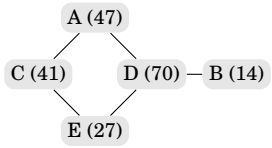


Figure 7.2. The obfuscated social network.

We want to find the minimum number of edge contractions that would make the graph 2-anonymous. In the current example, it is enough to contract only one edge to reach the social network depicted in Figure 7.3 (specifically, we contracted B and D together).

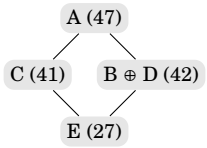


Figure 7.3. The resulting 2-anonymized social network ("B ⊕ D" stands for the resulting vertex from contracting "B" and "D" together.

Indeed, in the resulting social network, each vertex degree appears at least twice (in fact, four times). △

7.2. Introduction

As in Chapter 6, the task in this chapter is to preserve the privacy of the vertices, assuming an adversary which knows the degrees of some vertices. Importantly, even if we hide the identifying information from each vertex (for example, the name of the person corresponding to the vertex), the adversary might be able to de-anonymize some vertices. The task is to preserve the privacy of the entities comprising the given social network, while not modifying it too much. We refer the reader to Chapter 6 for a more thorough introduction of degree anonymization.

Here, in order to k-anonymize the input graph, we allow performing as few graph contractions as possible. That is, the set of our allowed operations is (several variants of) graph contractions. Studying graph contractions in the context of degree anonymization is interesting for several reasons. First, some variants of contractions can preserve original properties of the input graph (for example, connectivity). Second, vertex contraction (where also non-adjacent vertices can be contracted), is the inverse operation of vertex cleaving (as defined by Oxley [128, Chapter 3]; inverse in the sense that vertex cleaving splits vertices while vertex contraction merges them), which is studied in Chapter 6 (there, called *vertex cloning*). We mention also the importance of graph contractions to community detection in social networks and to clustering (see, for example, Delling et al. [54]).

7.2.1. Related Work

This chapter can be seen as complementing the line of research regarding degree anonymization (such as done in Chapter 6) by considering graph contractions, as a natural graph modification operation. For a literature review on anonymizing graphs, we refer the reader to Section 6.2.1.

This chapter also complements research done on the following problem. Given an input graph $G = (V, E)$ and a family $\mathscr{F}$ of graphs, the task is to find a subset of edges $E' \subseteq E$ of minimum size, such that after contracting the edges in E', G would be in the family $\mathscr{F}$. Indeed, in our case, $\mathscr{F}$ is the family of all k-anonymous graphs. Asano and Hirata [3] defined a set of conditions on $\mathscr{F}$, which are sufficient for NP-hardness (that is, they showed that, if some conditions on $\mathscr{F}$ hold, then this problem is NP-hard), while others studied the complexity of this problem for some specific graph classes (acting as $\mathscr{F}$). Planar graphs were considered by Golovach et al. [85], bipartite graphs and paths were considered by Heggernes et al. [94], and trees were considered by Guillemot and Marx [87].

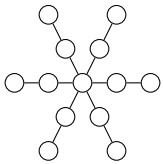


Figure 7.4. A path-star of degree 6 and length 2.

Most relevant to our work is the work done by Belmonte et al. [8], who considered the parameterized complexity of transforming a graph, by performing as few as possible edge contractions, such that the resulting graph would respect some degree constraints, such as being d-regular. Their work is of particular interest, as the concept of k-anonymity is a generalization of the notion of regularity (in particular, it is clear that a graph with n vertices is n-anonymous if and only if it is regular).

7.3. Specific Preliminaries

Graphs. Given a non-simple undirected graph $G = (V, E)$, which may have self-loops and parallel edges, we denote the degree of a vertex $v \in V$ by $\deg(v)$, and define

$$B_d := \{v \in V : \deg(v) = d\}$$

to be the set of vertices of degree d (called the *block* of degree d). As usual, we define the degree of a vertex v with x neighbors and y self-loops to be $x + 2y$ (in particular, we count a self-loop twice).

We define a *path-star* of degree d and length l to be the graph consisting of one center vertex, connected to d disjoint paths of length l each. As an example, consider the path-star depicted in Figure 7.4. Recall that a *caterpillar-tree* is a tree for which removing the leaves and their incident edges leaves a path graph (where a path graph can be defined as a tree that has no vertices of degree larger than 2).

Contractions. Given an undirected graph $G = (V, E)$ and two adjacent vertices, u and v, contracting the vertices u and v (usually referred to as contracting the edge $e = \{u, v\}$), means removing u and v from V, replacing them by one new

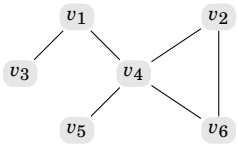
vertex (denoted by $u \oplus v$), and connecting this new vertex to exactly those vertices that were adjacent to at least one of u and v. The resulting graph is denoted by G/e.

More generally, given a set of edges $F \subseteq E$, we denote by G/F the graph obtained from G after contracting all edges in F. An undirected graph $G = (V,E)$ is said to be contractible to an undirected graph $G' = (V',E')$ if there is a set of edges $F \subseteq E$, such that $G/F = G'$. Equivalently, $G = (V,E)$ is contractible to $G' = (V',E')$ if and only if there exists a *witness structure* $V = V_1 \cup \ldots \cup V_{|V'|}$, where each V_i is called a *witness set*, such that for each V_i $(1 \leq i \leq |V'|)$ the subgraph of G induced by V_i is connected, and, for each pair of witness sets V_i and V_j $(1 \leq i \neq j \leq |V'|)$, we have that

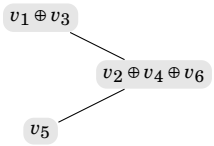
$$\{V_i, V_j\} \in E' \iff \exists v_i \in V_i, v_j \in V_j : \{v_i, v_j\} \in E$$

(indeed, the vertices in each witness set V_i are contracted such that, together, they form a single vertex). We define $\deg(V_i)$ to denote the resulting degree of the vertex corresponding to the contraction of the witness set V_i and we refer to the graph G' as the *witness graph*. An example is given next.

Example 16. Consider the following graph $G = (V,E)$:



and the following graph $G' = (V',E')$:



and notice that G is contractible to G', since $G/F = G'$, where

$$F = \{\{v_1, v_3\}, \{v_2, v_4\}, \{v_4, v_6\}\}.$$

Equivalently, this fact is also apparent by considering the witness structure $\{V_1, V_2, V_3\}$, where the witness sets are: $V_1 = \{v_1, v_3\}$, $V_2 = \{v_5\}$, and $V_3 = \{v_2, v_4, v_6\}$. △

Variants of Contractions. We also define the closely related operation of *vertex contraction*, which is defined similarly to edge contraction, with the only difference that, in vertex contraction, we allow to contract non-adjacent vertices as well (indeed, the vertices consisting a witness set of a vertex contracted graph are not assumed to be connected).

It is clear that a graph contraction operation can sometimes introduce self-loops and parallel edges. We define the following three variants of edge contractions and vertex contractions, differing by the way in which self-loops and parallel edges are treated.

- *Simple Contraction:* both self-loops and parallel edges are removed.
- *Hybrid Contraction:* self-loops are removed but parallel edges are kept.
- *Non-Simple Contraction:* both self-loops and parallel edges are kept.

For the Hybrid and Non-Simple variants, we allow the input graph to be non-simple. See Figure 7.5 and Figure 7.6 for some examples.

7.3.1. Main Problem

Given an undirected input graph G, we are interested in k-anonymizing it by performing at most c edge contractions. Recall that an undirected graph is said to be k-anonymous if every vertex degree in it occurs at least k times; equivalently, if, for each block B_i, it holds that $|B_i| = 0 \vee |B_i| \geq k$.

> DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS
> **Input:** An undirected graph $G = (V, E)$, an anonymization level $k \in \mathbb{N}$, and a budget $c \in \mathbb{N}$.
> **Question:** Can G be made k-anonymous by performing at most c contractions?

As we consider several variants of graph contractions, we consider the following specific variants of DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS.

- DEGREE ANONYMIZATION BY SIMPLE EDGE CONTRACTIONS, abbreviated as SEC-A.
- DEGREE ANONYMIZATION BY HYBRID EDGE CONTRACTIONS, abbreviated as HEC-A.

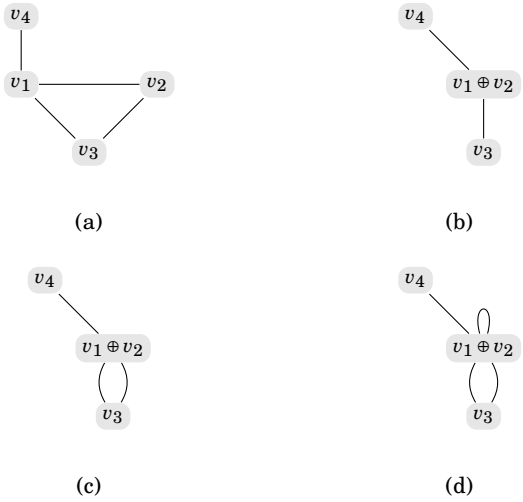


Figure 7.5. Examples for the different variants of graph contractions considered in this chapter. The original graph is depicted in (a) while the resulting graph by contracting v_1 and v_2 is depicted: in (b) for simple contractions; in (c) for hybrid contractions; and in (d) for non-simple contractions. Note that, for the case of non-simple contractions, the sum of degrees of v_1 and v_2 equals the degree of the newly created vertex $v_1 \oplus v_2$.

- Degree Anonymization by Non-Simple Edge Contractions, abbreviated as NEC-A.
- Degree Anonymization by Simple Vertex Contractions, abbreviated as SVC-A.
- Degree Anonymization by Hybrid Vertex Contractions, abbreviated as HVC-A.
- Degree Anonymization by Non-Simple Vertex Contractions, abbreviated as NVC-A.

Interestingly, it is not always possible to anonymize an undirected graph by performing only graph contractions, mainly since performing c contractions on an input graph with n vertices results in a graph with $n - c$ vertices. As an example,

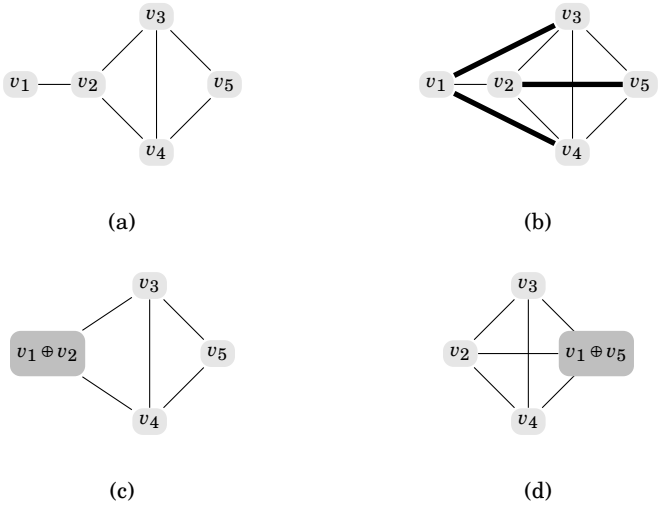


Figure 7.6. Example of 2-anonymizing an input graph. The input graph is depicted in (*a*), an optimal 2-anonymized graph with respect to edge addition is depicted in (*b*), an optimal 2-anonymized graph with respect to simple edge contraction or hybrid edge contraction is depicted in (*c*) (by contracting v_1 and v_2), and an optimal 2-anonymized (indeed, even 4-anonymous) graph with respect to non-simple vertex contraction is depicted in (*d*) (by contracting v_1 and v_5). Note that there is no solution with respect to non-simple edge contraction.

consider n-anonymizing a complete graph which has one missing edge: as the input graph is not n-anonymized, at least one edge needs to be contracted. As a result, the number of remaining vertices will be strictly less than n, thus the graph cannot become n-anonymous. This stands in contrast to the case of degree anonymization by edge addition, as any graph can be made n-anonymous by adding all missing edges to it.

It is interesting to note, however, that some graphs can be anonymized more efficiently (in terms of the number of graph modification operations needed) by using edge contractions than by using edge additions. A simple example of this phenomenon is shown in Figure 7.6. This gives some motivation for anonymizing by graph contractions, as it, presumably, can lead to k-anonymous graphs that do

Table 7.1. Parameterized complexity landscape of DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS. Rows and columns correspond to parameters, such that each cell corresponds to the combination of the corresponding parameters.

	Solution size c	Anonymization level k	Maximum degree Δ
c	W-h[a] [Th. 7.3] XP [Obs. 7.1]	W-h[a] [Th. 7.3] XP [Obs. 7.1]	FPT [Th. 7.5]
k		Para-NP-h[a] [Th. 7.3]	FPT[b] [Th. 7.1]
Δ			Para-NP-h[a] [Th. 7.4]

[a] Only for SEC-A and HEC-A (open for the other variants).
[b] Only for NVC-A (open for the other variants).

not differ by much from the original graph, thus preserve a lot of the structure from the input graph.

7.3.2. Overview of Our Results

We study the parameterized complexity of degree anonymization by graph contractions, considering the solution size c, the anonymity level k, and the maximum degree Δ as parameters. Being the most natural parameters, these are also studied in Chapter 6. From the variants defined in Section 7.3.1, we consider SEC-A and HEC-A as these are based on the most common operations (see, for example, Diestel [55, Chapter 1.7] and Wolle and Bodlaender [142]), and we consider NVC-A as it is equivalent to the underlying number problem (as defined in Section 7.4). Next, we state some important points of our work.

- Even the underlying number problem (NVC-A) is NP-hard. This stands in contrast to the case of degree anonymization by edge addition, for which the underlying number problem can be solved, in polynomial time, by dynamic programming [91]. This also stands in some contrast to the case of degree anonymization by vertex addition, which is considered in Chapter 6, where NP-hardness is proven only for a related problem. Moreover, SEC-A,

HEC-A, and NVC-A are NP-hard even on very restricted graph classes, such as trees and caterpillar trees.

- Parameterizing by either the solution size c, the maximum degree Δ, or the anonymity level k, does not help for tractability. This stands in contrast to anonymization by edge addition, which is fixed-parameter tractable with respect to the maximum degree Δ [91]. However, combining Δ with the solution size c does help for tractability.
- Combining the maximum degree Δ with the anonymity level k helps for tractability for some variants of the problem. For some other variants, we show some evidence suggesting hardness.

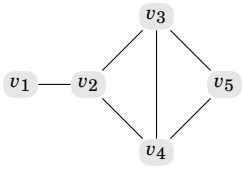
Table 7.1 provides an overview of our results.

7.4. NP-Hardness Results

We begin by considering NVC-A, which is equivalent to a certain number problem, which we call *the underlying number problem*. Recall that, in NVC-A, we allow to contract any two vertices, even if they are not connected by an edge. Moreover, since we keep all self-loops and parallel edges, we have that the degree of a new vertex, corresponding to some witness set, equals the sum of the degrees of the vertices in the witness set; thus, the degree sequence of the resulting graph after performing some contractions only depends on the original degrees of the contracted vertices. Therefore, we do not care about the exact connections between the vertices in the graph, but only care about the degrees. It follows that NVC-A is equivalent to the following number problem. Therein, a multiset of integers is k-anonymous if each integer in it occurs at least k times.

NVC-A (NUMBER VERSION)
Input: A set $V = \{d_1, \ldots, d_n\}$ of n integers ($\forall i : 0 \leq d_i \leq \Delta$) and two integers, k and c.
Question: Is there a partition $V = V_1 \cup V_2 \cup \cdots \cup V_z$ (where $V_{j_1} \cap V_{j_2} = \emptyset$ for $1 \leq j_1 \neq j_2 \leq z$) such that the multiset $S = \{\sum_{d_i \in V_j} d_i : j \in [z]\}$ is k-anonymous and $\sum_{j \in [z]}(|V_j| - 1) \leq c$?

Example 17. As an example, consider the following graph as an input to NVC-A, with anonymity level $k = 2$ and budget $c = 1$.



Since in NVC-A we consider non-simple vertex contractions, we can, informally speaking, disregard the graph structure, and concentrate only on the sequence of the degrees. Specifically, we have the following input to the equivalent formulation of NVC-A. The set V of integers is just the sequence of degrees of the input graph: $V = \{v_1, v_2, v_3, v_4, v_5\} = \{1, 3, 3, 3, 2\}$. The anonymity level k is still 2 and the solution size is still 1. A solution partition is given by:

$$V_1 = \{v_1, v_5\},$$

$$V_2 = \{v_2\},$$

$$V_3 = \{v_3\},$$

$$V_4 = \{v_4\},$$

since, for each $j \in [4]$, it holds that $\sum_{d_i \in V_j} d_i = 3$ (that is, in fact, the resulting sequence is 4-anonymous). △

Informally, the above number problem lies within the heart of the graph anonymization problem (for this reason we call it the underlying number problem). Interestingly, contrary to the situation for other operations (such as edge addition, where the underlying number problem can be solved in polynomial time by a simple algorithm based on dynamic programming [91]), here the underlying number problem is intractable.

Theorem 7.1. NVC-A *is* NP*-hard even on caterpillar trees.*

Proof. We provide a reduction from the following strongly NP-hard problem [84]:

> STRICTLY THREE PARTITION
> **Input:** A set of integers $S = \{a_1, \ldots, a_{3m}\}$ such that $\sum_{a_i \in S} a_i = mB$ and $\forall i \in [3m] : B/4 < a_i < B/2$.
> **Question:** Are there m disjoint sets $S_1, \ldots, S_m$, each of size 3, such that $\forall j \in [m] : \sum_{a_i \in S_j} a_i = B$?

Given an instance of STRICTLY THREE PARTITION, we create an instance of NVC-A. Intuitively, the idea is to create a set of $3m$ vertices, such that each number a_i would have a corresponding vertex whose degree is proportional to a_i. Then, we will add a distinguished vertex with degree proportional to B, making sure that the only way of anonymizing the block containing this distinguished vertex is by contracting m triplets of vertices corresponding to triplets of numbers, such that each of them sums to exactly m.

Specifically, we first scale the input numbers, that is, we define $a'_i := a_i \cdot mB$ and $B' := B \cdot mB$. For each number a'_i, we create a node $v_{a'_i}$ and connect it to a'_i paths of length $c+1$ (consisting of new vertices), such that $\deg(v_{a'_i}) = a'_i$ holds for each i (that is, for each a'_i, we create a path-star of degree a'_i and length $c+1$). We add a path-star of degree B' and length $c+1$. We set $k := m+1$ and $c := 2m$. This finishes the construction, which can be computed in polynomial time. Indeed, the construction as specified above results in a forest. To strengthen the result, we can transform it to be a caterpillar tree by placing all $v_{a'_i}$'s on a path together with the path-star of degree B', adjusting the number of additional new vertices connected to each $v_{a'_i}$ accordingly. An example follows.

Example 18. We provide an example for explaining the general idea underlying the reduction, but, for clarity of presentation, we omit some technical issues, such as the assumption of strictness (that is, that $\forall i : B/4 < a_i < B/2$). Moreover, we do not scale the numbers, and instead of path-stars, we create regular stars. Thus, while the reduction as shown next is not formally correct, it conveys the general idea.

Consider the following set of integers $S = \{1,2,3,4,5,7\}$ as an input to STRICTLY THREE PARTITION. We create the graph depicted in Figure 7.7.

The star of degree $(1+2+3+4+5+7)/2 = 11$, and only it, needs to be anonymized. All other stars correspond to the original integers from S, therefore their degrees are $\{1,2,3,4,5,7\}$, that is, equivalent to the integers from S. A possible way of anonymizing the graph is by contracting the stars of degree 1, 3, and 7, together, to create a star of degree $1+3+7 = 11$, and also to contract the stars of degree 2, 4, and 5, together, to create another star of degree $2+4+5 = 11$. As a result, we would have three stars of degree 11, and the graph would be 3-anonymized. Indeed, this solution corresponds to the partition $\{1,3,7\} \cup \{2,4,5\} = \{1,2,3,4,5,7\}$ of the original STRICTLY THREE PARTITION.

As mentioned above, it is possible to transform the graph into a caterpillar tree. Considering the current example, we can construct the graph depicted

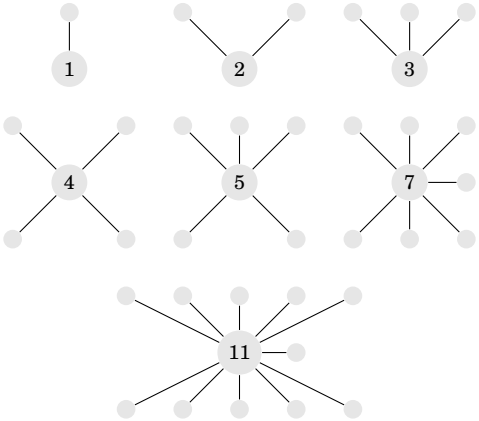


Figure 7.7. The graph used in Example 18.

in Figure 7.8, which is a caterpillar tree. Specifically, we connected the stars to form a path, and adjust the degrees by removing some of the vertices which used to be adjacent to the stars. △

Given a partitioning of S into triplets of integers, such that each triplet sums exactly to B, it is possible to anonymize the graph by contracting each triplet of vertices $v_{a'_i}$, corresponding to a triplet of integers a'_i in the partitioning of S, into a single vertex. Note that we need two contractions for each triplet and that the resulting graph is k-anonymous, as it has m new vertices of degree B', and no other new vertices.

For the other direction, note that, due to the strictness constraint (that is, since $\forall i : B/4 < a_i < B/2$), we have that any witness set of size other than three will have degree which is far away from B'. Combining this observation with the fact that we multiplied each a_i by mB, it holds that, if there is no partitioning of S into triplets, then, in any partitioning, there exists at least one triplet such that the difference between its degree and B' is at least mB.

We conclude that the above way cannot be used in order to anonymize the block containing the distinguished path-star. Moreover, contracting the path-star itself will not anonymize it, as it can decrease its degree by at most c, which is not enough for it to fall to any other degree block. Therefore, any solution must

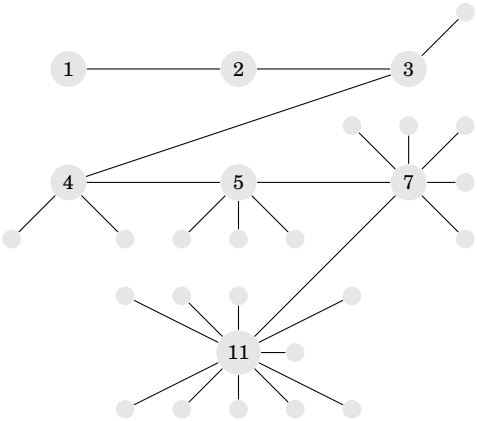


Figure 7.8. The modified caterpillar tree used in Example 18.

introduce at least m new vertices of degree B', each corresponding to a triplet. Thus, a solution must correspond to a correct partitioning of S into triplets.

Finally, the reduction can be computed in polynomial-time, since the STRICTLY THREE PARTITION problem is NP-hard in the strong sense [84], and therefore NP-hard even when the input is given in unary. □

The other variants of DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS considered in this chapter are NP-hard as well. Specifically, we show NP-hardness on trees for SEC-A and HEC-A.

Theorem 7.2. *Both* SEC-A *and* HEC-A *are* NP*-hard even on trees.*

Proof. We provide a reduction from the following strongly NP-hard problem [84]:

> NUMERICAL MATCHING WITH TARGET SUMS
> **Input:** Three sets of integers $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $C = \{c_1, \ldots, c_n\}$.
> **Question:** Can the elements of A and B be paired such that, for each $i \in [n]$, c_i will be equal to the sum of the ith pair?

The variant where all $3n$ input integers are distinct is also known to be NP-hard in the strong sense [99]. Without loss of generality, we assume that all input

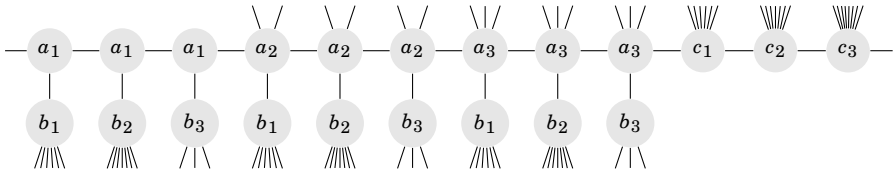


Figure 7.9. Example for the reduction used in the proof of Theorem 7.2. Specifically, the reduction is shown for the following instance of NUMERICAL MATCHING WITH TARGET SUMS: $A = \{3,5,6\}$, $B = \{7,8,4\}$, and $C = \{10,11,12\}$. All drawn edges, except for the edges from a_i's to b_j's, represent paths of length $c+1 = 4$.

integers are greater than three. Given an instance of NUMERICAL MATCHING WITH TARGET SUMS, we create an instance of SEC-A and HEC-A (the same instance for both), as follows. Intuitively, the idea is to create a set of $k-1$ vertices for each c_i and a pair of vertices for each pair of a_i and b_j, such that the only possibility of anonymizing the vertices corresponding to the c_i's is to contract the correct pairs of a_i's and b_j's. Details follow.

We set the anonymity level k to $n-1$ and the budget c to n. We construct some *c-gadgets:* for each c_i, we create $k-1$ path-stars of degree c_i-2 and length $c+1$. We construct some *ab-gadgets:* for each pair of integers, $i \in [n]$ and $j \in [n]$, we create two path-stars, one of degree a_i and another of degree b_j, both of length $c+1$, and connect them by an edge (indeed, the construction as such is a forest; we can transform it into a tree by arbitrarily connecting each pair of disconnected components by a path of length $c+1$). See Figure 7.9 for a visualization of the reduction.

Given a correct pairing of A and B, it is possible to anonymize the input graph by contracting the corresponding ab-gadgets: we will have $n-1$ remaining vertices for each a_i and b_j, as well as $k-1+1$ vertices for each c_i.

For the other direction, note that all c-gadgets must be anonymized in a solution, by contracting some ab-gadgets (it is not a good idea to contract c-gadgets together, since we can assume that each c_i is greater than all a_i's and b_j's; to see this, add the maximum integer to all a_i's and b_j's and add it twice to all c_l's). Further, contracting two ab-gadgets sharing the same a_i (or the same b_j) would result in de-anonymizing the block of degree a_i (or of degree b_j), which

cannot be further anonymized. Therefore, a solution must correspond to a correct pairing.

Finally, the reduction can be computed in polynomial-time, since the NUMERICAL MATCHING WITH TARGET SUMS problem is NP-hard in the strong sense [84], and therefore NP-hard even when the input is given in unary. □

7.5. General Graphs

Following the NP-hardness results from Section 7.4, we continue our quest for tractability by considering parameters. We begin by considering the number c of contractions (that is, the solution size) and the anonymity level k. First, we observe that for constant values of the solution size c, for all variants of DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS considered in this chapter, we can simply enumerate all possible solutions, thus concluding the following.

Observation 7.1. DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS *is* XP *with respect to the solution size c.*

There is no hope, however, for fixed-parameter tractability with respect to the parameter solution size c, as even combining it with the anonymity level k does not yield fixed-parameter tractability. Note that, in the corresponding parameterized reduction, the anonymity level k is a constant.

Theorem 7.3. *Both* SEC-A *and* HEC-A *are* NP*-hard and* W*-hard with respect to the solution size c, even for anonymity level $k = 2$.*

Proof. For SEC-A, we provide a reduction from the following W[2]-hard problem, parameterized by the solution size [57]:

> SET COVER
> **Input:** A universe of elements $X = \{x_1, \ldots, x_n\}$, a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of sets over the universe, and a budget h.
> **Question:** Is there a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ of sets such that $|\mathcal{S}'| \leq h$ and $\bigcup_{S \in \mathcal{S}'} S = X$?

Given an instance of SET COVER, we create an instance of SEC-A, as follows. We set $k := 2$ and $c := h$. For each x_i we create a vertex x'_i. For each S_j we create two vertices, S'_j and S''_j, and connect them by an edge. Each S'_j and S''_j (corresponding to a set S_j) are connected to all x'_i's which correspond to

elements $x_i \in S_j$. We add several paths of length $c+1$ to each x'_i such that the degree of each x'_i will be $f(i) := i(c+1)+2$. Similarly, we add several paths of length $c+1$ to each S'_j and S''_j, such that the degree of each S'_j and S''_j will be $f(n+1)$. For every $i \in [n]$ and $z \in [h]$, we add a path-star of degree $f(i)-z$ and length $c+1$, called a *landing gadget*. We add k path-stars of degree $f(n+1)$ and length $c+1$ in order to anonymize the vertices corresponding to the sets.

Given a set cover, that is, a subcollection $\mathscr{S}' \subseteq \mathscr{S}$ which covers the universe, it is possible to anonymize the input graph as follows. For each $S_j \in \mathscr{S}'$, contract together the corresponding pair of vertices S'_j and S''_j. As a result, the degrees of each vertex x'_i, corresponding to a universe element x_i, will decrease by the number of sets covering it. Recalling the landing gadgets, we have that the graph would become anonymized as a result.

For the other direction, note that all of the x_i's need to be anonymized, and we can only decrease their degree to achieve this. Therefore, by a simple exchange argument, a solution must correspond to a set cover.

For HEC-A, we provide a reduction from the following W[1]-hard problem, parameterized by the solution size [57]:

> MULTICOLORED CLIQUE
> **Input:** An undirected graph whose vertices are colored in h colors and an integer h.
> **Question:** Is there a set of h pairwise adjacent vertices such that each vertex is of different color?

We assume, without loss of generality, that there are no monochromatic edges (a monochromatic edge is an edge connecting vertices of the same color). Cai [36] showed that MULTICOLORED CLIQUE remains hard even on regular graphs.

Given an instance of MULTICOLORED CLIQUE, we create an instance of HEC-A. We define the following function, $f(i) = 2^i \cdot 2\binom{h}{2}$, whose domain is the set of colors (that is, $i \in [h]$). We set $k := 2$ and $c := h-1$. We denote the color of vertex v by $\text{color}(v) \in [h]$. For every vertex v, we add $(f(\text{color}(v)) - \deg(v))$ paths of length $c+1$, such that the degree of each vertex colored in color $i \in [h]$ is $f(i)$. We construct $k+1$ copies of this modified graph. We add $k-1$ path-stars of degree $((\sum_{i\in[h]} f(i)) - 2\binom{h}{2})$ and length $c+1$.

Given a multicolored clique of size h, it is possible to contract the vertices of the clique into one vertex: the degree of the new vertex will be equal to the degree of the $k-1$ path-stars, resulting in an anonymized graph, due to the $k+1$ copies.

For the other direction, note that contracting edges of a path-star does not change its degree. Moreover, as there are no monochromatic edges, we can

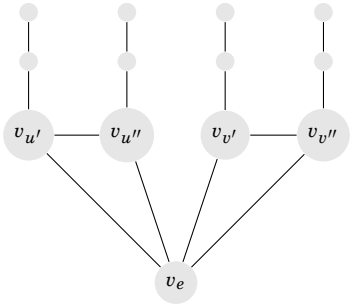


Figure 7.10. Gadget used for the reduction in the proof of Theorem 7.4 (for SEC-A). Specifically, the construction is shown for two vertices, u and v, which are connected by the edge $e = \{u, v\}$ (indeed, as the input graph for the reduction is cubic, the vertices $v_{u'}$,$v_{u''}$,$v_{v'}$,$v_{v''}$ all have degree five, and not three, as is depicted here). Note that contracting $v_{u'}$ and $v_{u''}$ (or $v_{v'}$ and $v_{v''}$) decreases the degree of v_e.

only contract edges of different colors. Due to the way we defined $f(i)$, the only possible way of reaching the degree of the path-star (that is, $\sum_{i\in[h]} f(i) - 2\binom{h}{2}$) is by contracting a multicolored clique, since all colors are needed for the first part (that is, $\sum_{i\in[h]} f(i)$) and all edges between the colors are needed for the second part (that is, $2\binom{h}{2}$). □

7.6. Bounded-Degree Graphs

We go on to consider the maximum degree Δ of the input graph. It is known that degree anonymization by edge addition is fixed-parameter tractable with respect to Δ, and there is even a polynomial kernel with respect to this parameter [91]. In contrast to this, we next show that in case of edge contractions, the parameter maximum degree Δ alone does not help for tractability, as both SEC-A and HEC-A are NP-hard even for constant values of this parameter.

Theorem 7.4. *Both* SEC-A *and* HEC-A *are* Para-NP*-hard with respect to the maximum degree* Δ.

Proof. For SEC-A, we provide a reduction from the following NP-hard problem (where an undirected graph is said to be *cubic* if it is 3-regular) [84]:

CUBIC VERTEX COVER
Input: A 3-regular undirected graph $G = (V(G), E(G))$ and an integer h.
Question: Is there a set of h vertices such that each edge is incident to at least one vertex from the set?

Given an instance of CUBIC VERTEX COVER, we create an instance of SEC-A, as follows. We set $k := |E| + 1$ and $c := h$. For every edge $e \in E$, we create a new vertex v_e. For every vertex $v \in V$, we create a pair of new vertices v'_v and v''_v, and connect each such pair by an edge. For every edge $e = \{u, v\}$, we connect v_e to the four vertices v'_u, v''_u, v'_v, and v''_v. We also connect each v'_v and v''_v to a path of length $c+1$ each. We add k path-stars of degrees 1, 2, 3, and 5, all with length $c+1$. See Figure 7.10 for an example.

The general idea is that contracting the two vertices v'_v and v''_v corresponding to a vertex v, would decrease the degrees of the vertices v_e corresponding to all of its incident edges e. That is, given a vertex cover, we contract each pair of v'_v and v''_v corresponding to a vertex v in the given vertex cover: as a result, the degree of each vertex v_e is decreased from 4 to either 3 or 2, k-anonymizing the graph.

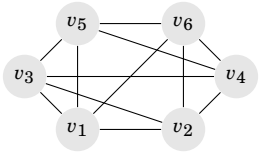
For the other direction, note that the block of degree four needs to be anonymized. However, there is no way of increasing the degree of the vertices in this block (since any contraction would decrease their degree) and no way of increasing the degrees of other vertices to fall into their blocks; thus, their degree decreases in any solution. For an edge $e = \{u, v\}$, the only possibility of decreasing the degree of v_e is by contracting either the pair v'_u and v''_u or the pair v'_v and v''_v. By a simple exchange argument, we get that a solution must correspond to a vertex cover.

For HEC-A, we reduce from the following problem, which was shown to be NP-hard even on 4-regular graphs by van Rooij et al. [140]:

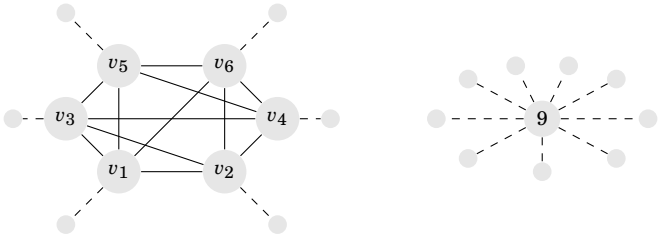
4-REGULAR PARTITION INTO TRIANGLES
Input: An undirected 4-regular graph $G = (V, E)$.
Question: Can V be partitioned into sets $S_1, \ldots, S_{|V|/3}$, such that each S_i consists of three vertices which form a triangle in G?

Given a 4-regular input graph G for 4-REGULAR PARTITION INTO TRIANGLES, we create an input graph G' for HEC-A. We initialize G' by G and add a path of length $c+1$ to each vertex (consisting of new vertices). We create a path-star of degree nine and length $c+1$. We set $k := n/3+1$ and $c := 2n/3$ (we assume, without loss of generality, that $n = 0 \mod 3$). This finishes the construction, which can be computed in polynomial time. An example follows.

Example 19. Consider the following 4-regular graph, given as an input to 4-REGULAR PARTITION INTO TRIANGLES.



We construct the following (unconnected) graph as an input to HEC-A.



Contracting v_1, v_3, and v_5 together, as well as contracting v_2, v_4, and v_6 together, will result in another two stars of degree nine. As a result, the graph would become 3-anonymous, as it would consist of three stars, each of degree nine. △

Given a partition of G into triangles, it is possible to contract the vertices of each triangle together: the degree of the resulting vertex would be nine, therefore the graph will be k-anonymous. For the other direction, note that the path-star of degree nine needs to be anonymized, its degree cannot decrease or increase, and the only way of introducing other vertices of degree nine is by contracting triangles. Finally, as we need $n/3$ triangles, a solution must correspond to a correct partitioning of G into triangles. □

Contrary to the above hardness results, combining the maximum degree Δ together with the solution size c does help for tractability, for all variants of DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS considered in this chapter.

Theorem 7.5. DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS *is* FPT *with respect to* (Δ, c)*, where* Δ *is the maximum degree and* c *is the solution size.*

Proof. Consider a yes-instance I of DEGREE ANONYMIZATION BY GRAPH CONTRACTIONS. Since I is a yes-instance, it means that there exists a set E' of at most c edges such that contracting them would make G k-anonymous.

Consider the set V'' containing all vertices whose degree might change as a result of contracting the edges in E'. Consider the set V' of vertices containing all the endpoints of the edges in E', including also all their neighbors. That is, we define $V' := N[\{u, v : \{u, v\} \in E'\}]$ (recall that the closed neighborhood of a vertex $v \in V$ is $N[v] := \{v\} \cup \{u : \{u, v\} \in E\}$, and that the closed neighborhood of a set of vertices $V' \subseteq V$ is $N[V'] := \cup_{v' \in V'} N[v']$). Since each edge has two endpoints and each vertex has at most Δ neighbors, it follows that $|V'| \leq 2c(\Delta + 1)$. Moreover, and most importantly, it holds that $V'' \subseteq V'$.

The main point now is that it is enough to find the subgraph induced by V'. To this end, we consider all possible graphs H containing at most $2c(\Delta + 1)$ vertices. For each such graph H, we consider all possible sets C of at most c edges to be contracted. For each such pair of graph H and set C, we compute the degree changes in H, incurred by contracting the edges in C. If these degree changes make the graph k-anonymous, then we try to find this graph H as a subgraph in G. Indeed, we are looking for a subgraph whose number of vertices is upper-bounded by the parameters, inside a graph whose maximum degree is upper-bounded by the parameters; this problem is known to be fixed-parameter tractable, as shown by Cai et al. [37, Theorem 1]. □

We consider now the combined parameter Δ and k. From a technical point of view, the situation here is more involved. Moreover, we can only show fixed-parameter tractability for NVC-A, while for SEC-A and HEC-A, we show some evidence suggesting otherwise. We begin by showing that, for NVC-A, the solution size c can be upper-bounded by a function dependent only on the values of the parameters Δ and k. Then, we will be able to use Theorem 7.5 to obtain fixed-parameter tractability for NVC-A, with respect to the combined parameter Δ and c.

Lemma 7.1. *For any yes-instance* (V, k, c) *of* NVC-A *it holds that* (V, k, c')*, with* $c' = k \cdot (\Delta \cdot \Delta!)^{\Delta+1}$*, is also a yes-instance.*

Proof. Let (V, k, c) be a yes-instance of NVC-A. Denote by $c_{\text{opt}} \leq c$ the smallest number such that (V, k, c_{opt}) is still a yes-instance. Moreover, let the partition $P = \{V_1, \ldots, V_i\}$ of V be a solution which corresponds to c_{opt} (in other words, let P be the witness structure corresponding to a solution of (V, k, c_{opt})).

In what follows, we define two operations on P, with the property that whenever at least one of them is applicable, applying it would result in another solution with less contractions. We will also show that, as long as $c_{\text{opt}} > k \cdot (\Delta \cdot \Delta!)^{\Delta+1}$, at least one of them is applicable. This implies that $c_{\text{opt}} \leq k \cdot (\Delta \cdot \Delta!)^{\Delta+1}$, thus proves Lemma 7.1.

To formally describe our operations, we associate with each witness set V_i a *witness vector* $\vec{v_i} \in \mathbb{N}^{\Delta}$ with $\vec{v_i}[j]$ being equal to the number of vertices of degree j in the witness set V_i. Recall that the degree of a witness set is defined to be the sum of the degrees of the vertices in the witness set (that is, the degree of the vertex corresponding to contracting all the vertices in the witness set).

Operation 1. This operation is applicable to P if there are at least k witness sets in P, all of equal degree, such that in each of them, say V_i, there is at least one j with $\vec{v_i}[j] \geq \Delta!$. If there exists such a collection of witness sets, then consider such a collection $P' \subseteq P$ which is maximal with respect to containment, and do the following: for each witness set V_i in P' choose any integer j with $\vec{v_i}[j] \geq \Delta!$, remove $(\Delta!/j)$-many vertices of degree j from V_i (note that $\Delta!/j$ is always an integer), and form a new witness set containing these vertices. This finishes the description of the operation.

Note that we introduced at least k new witness sets, all of degree exactly $\Delta!$. Moreover, we decreased the degree of each of the initial witness sets by the same number $\Delta!$. Since there are at least k such witness sets, it follows that performing this operation results in a partition of V that is still a solution for (V, k, c_{opt}) which requires less edge contractions than P requires.

Operation 2. This operation is applicable to P if there is a collection of at least k witness sets in P, such that the witness sets in this collection all have the same witness vector, and the Hamming weight of this witness vector is at least 2 (that is, these witness sets are not singletons). If such a collection exists, then choose an arbitrary integer j occurring in this same witness vector and do the following: for each witness set V_i in this collection, remove one vertex of degree j from V_i, and form a new witness set containing only this vertex of degree j (that is, form a new singleton witness set). This finishes the description of the operation.

Note that there are at least k witness sets, where rom each of them, a vertex of the same degree j is being cut out. Therefore, the resulting partition is a solution for (V, k, c_{opt}) which requires less edge contractions than P requires.

Applicability. It remains to argue that, as long as $c_{\text{opt}} > k \cdot (\Delta \cdot \Delta!)^{\Delta+1}$, at least one of the two operations described above is applicable. First, assume that P contains a witness set V_i of degree at least $(\Delta \cdot \Delta!)$. Then, since P is k-anonymous, it holds that there are at least k witness sets of the same degree, which is at least $(\Delta \cdot \Delta!)$. It follows, by the pigeon-hole principle, that each of these witness sets must contain at least one integer j which occurs at least $\Delta!$ times in it. Thus, Operation 1 is applicable.

So, let us assume now that the degree of each witness set in P is at most $(\Delta \cdot \Delta!)$. Then, there can be at most $(\Delta \cdot \Delta!)^{\Delta}$ different witness vectors, such that none of these witness vectors is of degree greater or equal to $(\Delta \cdot \Delta!)$. It follows, by the pigeon-hole principle, that if P contains at least $k \cdot (\Delta \cdot \Delta!)^{\Delta}$ witness sets of size at least two, then Operation 2 is applicable.

Finally, a solution for which $c_{\text{opt}} > k \cdot (\Delta \cdot \Delta!)^{\Delta+1}$ edge contractions have been performed either contains a set of size at least $(\Delta \cdot \Delta!)$ or it contains at least $k \cdot (\Delta \cdot \Delta!)^{\Delta}$ witness sets of size at least two. □

We can combine the results of Theorem 7.5 and Lemma 7.1 to show the following fixed-parameter tractability result.

Corollary 7.1. NVC-A *is* FPT *with respect to* (Δ, k)*, where* Δ *is the maximum degree and* k *is the anonymity level.*

Proof. For a given instance (V, k, c) of NVC-A we decide the instance $(V, k, \min\{c, k \cdot (\Delta \cdot \Delta!)^{\Delta+1}\})$ using the FPT-algorithm with respect to (Δ, c) (as described in Theorem 7.5). By Lemma 7.1, it follows that these two instances are equivalent and the corresponding running time proves fixed-parameter tractability with respect to (Δ, k). □

We mention that, since we perform a series of operations (Operation 1 and Operation 2), and each of them decrease the size of the graph, the general idea resembles kernelization. In fact, if we would have a kernel for NVC-A, when parameterized by the combined parameter (Δ, c), then we would also have a kernel for NVC-A, when parameterized by the combined parameter (Δ, k). It is not clear, however, how to obtain a kernel for NVC-A, when parameterized by the combined parameter (Δ, c), since Theorem 7.5 is based on a branching algorithm.

We do not know whether SEC-A and HEC-A are fixed-parameter tractable with respect to the combined parameter Δ and k. We can, however, rule out the possibility of a similar proof technique as used in the proof for NVC-A (in Corollary 7.1) for these variants. The reason is that, contrary to Lemma 7.1, both for

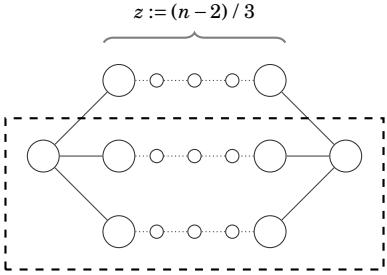


Figure 7.11. An input for SEC-A and HEC-A with small maximum degree $\Delta = 3$ and small anonymity level $k = 3$, but where the solution size c cannot be upper-bounded by a function dependent only on Δ and k. The small nodes are there to represent a lot of nodes (specifically, $z-2$ on each path). The marked region represents an optimal solution, that is, contracting all of the vertices in the marked region together results in a 3-anonymized graph, as the corresponding vertex would have degree 2.

SEC-A and for HEC-A, there are instances for which the maximum degree Δ is small, the anonymity level k is small, but the solution size c is large.

Proposition 7.1. *There are instances for* SEC-A *and* HEC-A *for which the solution size c cannot be upper-bounded by a function dependent only on the maximum degree Δ and the anonymity level k.*

Proof. Consider the instance depicted in Figure 7.11. There, the maximum degree Δ is 3. If we set the anonymity level k to 3, then the graph is certainly not anonymous. The only way of anonymizing it is by contracting a big part of the graph, specifically, the minimum c for which the instance is a yes-instance is $c = 2z + 1$. Therefore, in this graph, the solution size c is $\Omega(n)$.

To see this, note that by contracting the marked region (in Figure 7.11), the graph becomes 3-anonymous. Moreover, note that there is no way of introducing further vertices of degree 3, and that the only way of transforming the odd-degree vertices to be of even degree is by contracting them to be in the same witness set, as shown in the marked region (in Figure 7.11). Specifically, contracting together all the vertices in this marked region would result in a new vertex of degree 2. □

Following Proposition 7.1, we conclude that, for SEC-A and HEC-A, either a different proof technique is needed to prove fixed-parameter tractability with

respect to the combined parameter Δ and k, or, as we conjecture, these variants are not fixed-parameter tractable for the combined parameter (Δ, k).

7.7. Outlook

We state several possibilities for future research, motivated by the work presented in this chapter.

- There are several open questions, as can be identified by looking at Table 7.1. The most intriguing question is whether the FPT algorithm presented in Corollary 7.1 can be extended to the other variants of contractions considered in this chapter.

- An immediate extension of the work presented in this chapter is to identify and study other parameters. Some of the parameters, naturally, will not help for tractability. For example, most of the hardness reductions in this chapter do not require large witness sets; thus, the parameter "maximum size of a witness set" does not look promising for tractability. Surely, however, other parameters might help for tractability.

  Of most interest are parameters, such as the average pairwise distance and the diameter, which are known to be small in certain social networks [124].

- It is natural to consider other graph operations which are related to graph contractions. Some examples are *structure contraction* (contracting a whole subgraph at the cost of one operation), *edge twisting* (see [128, Chapter 3]), and *vertex dissolution* (see [128, Chapter 3] and the work done by van Bevern et al. [12]).

- It is also natural to combine several operations. For example, as each graph contraction causes the number of vertices to be decreased by one, it might be useful in some scenarios to "compensate" for this by also adding vertices: one way to achieve this is to require that the resulting graph would have the same number of vertices as the input graph, and to allow both graph contractions and vertex additions to be performed on the input graph.

- As mentioned in the introduction to this chapter, some graphs, for some k's, can be k-anonymized more efficiently by contracting edges than, say, by adding edges (in terms of the number of graph modification operations

needed). A better understanding of the conditions for this type of phenomena is of interest: a k-anonymized graph obtained by performing few graph contractions might preserve more of the structure of the original graph than, say, a k-anonymized graph obtained by adding a lot of edges. Of course, one might also compare the efficiency of degree anonymization by graph contractions to the operation of vertex addition, discussed in Chapter 6, and to other graph modification operations of interest.

- Bazgan and Nichterlein [7] studied graph anonymization with edge deletions and vertex deletions from the viewpoint of approximation algorithms. They mainly obtained inapproximability results when the goal of the approximation algorithm is to minimize the number of edit operations. One possible way of extending this line of research would be to study whether their results transfer to edge contractions and to other operations. A further line of research might be to study different notions of approximations. As examples, consider the following types of approximations (indeed, this discussion about approximation applies also to the problem of degree anonymization by vertex addition, discussed in Chapter 6).

    1) **Partial anonymization:** the goal here is to partially anonymize an input graph. In other words, the task is to maximize the number of vertices which are anonymized, that is, the number of vertices for which at least $k-1$ vertices of the same degree exist.
    2) **Rough anonymization:** the goal here is to roughly anonymize an input graph. In other words, the task is to have, for each vertex, at least $k-1$ other vertices of roughly the same degree; specifically, to minimize the value of α such that, for each vertex of degree d, there will be at least $k-1$ other vertices whose degree is at least $k-\alpha$ and at most $k+\alpha$.

8. Outlook and Conclusion

This final chapter begins, in Section 8.1, with some ideas for future research motivated by this thesis, and ends, in Section 8.2, with some concluding remarks.

8.1. Outlook

When taking a general perspective on the contents of this thesis, some orthogonal ingredients of the study presented here can be revealed. Specifically, it stands out that the computational problems considered here can be, abstractly, described as follows:

"an agent,
towards its goal,
can perform a small number of operations
from a set of predefined operations".

This view suggests some routes for future research.

The set of allowed operations. Recall that, while the goal of the agent in both Chapter 3 and Chapter 4 is to make its preferred candidate win the election, its set of allowed operations is different for each of these chapters individually. Specifically, in Chapter 3 the agent is able to change the set of candidates in the election, while in Chapter 4 the agent is able to change some votes. It is natural to consider different sets of operations, for example, the operation of changing the set of voters in the election.

Similarly, recall that, while the goal of the agent in both Chapter 6 and Chapter 7 is to k-anonymize the social network, its set of allowed operations is different for each of these chapters individually. Specifically, in Chapter 6 the agent is allowed to add new vertices, while in Chapter 7 the agent is allowed to perform graph contractions. It is natural to consider different sets of operations, for example, performing vertex dissolutions.

More generally, it makes sense to combine some operations, that is, to allow the agent to perform several types of operations. For example, one might combine the candidate control operation considered in Chapter 3 together with the shift bribery operation considered in Chapter 4. As another example, one might combine the vertex addition operation considered in Chapter 6 together with the graph contraction operation considered in Chapter 7.

The agent's goal. As mentioned above, the goal of the agent in both Chapter 3 and Chapter 4 is to make its preferred candidate win the election. One might consider further goals. For example, the agent's goal could be to have a specific ranking of the candidates in the resulting election. This makes sense, for example, in the context of multi-winner elections, where the agent might want a specific set of (possibly ranked) candidates to win the election.

Similarly, as mentioned above, the goal of the agent in both Chapter 6 and Chapter 7 is to k-anonymize the social network. One might consider further goals. For example, the agent's goal could be to have that each vertex will have at least $k-1$ other vertices with the same neighborhood structure. As another example, one might require from the agent to preserve some properties of the social network. Indeed, this is done to some extent in Chapter 6, by requiring pairwise distances, connectivity, and the diameter of the graph to be preserved. This is also done to some extent in Chapter 5, by requiring the winner of the election to be preserved. Naturally, one might consider further properties to be preserved.

On a different note, it is natural to require from the agent to only approximate its goal. That is, while there are some approximation algorithms in this thesis, devising more approximation algorithms for the problems considered here is a natural direction for future research.

Assuming stronger adversaries, pursuing harder goals, and allowing different sets of operations, might help in bridging the modeling gap between real-world problems and their mathematical abstractions. That is, pushing the goals of our agents and allowing them to perform more complex operations might render the algorithms to be developed for these problems more practical and useful.

We now take a different general perspective on the contents of this thesis, concentrating on the topics which were studied. Specifically, the aim of the study conducted here can be, abstractly, described as follows.

"bringing the study of manipulating elections to social networks
and
bringing the study of anonymizing social networks to social choice".

This view suggests some further routes for future research.

From social choice to social networks. One of the main goals of Chapter 3 and Chapter 4 is to lift the existing research on control and bribery in social choice to a more general (and, therefore, more applicable) scenario; specifically, to a scenario where an elections is held over an underlying social network.

Indeed, the models suggested in Chapter 3 and in Chapter 4 only scratch the surface of understanding the complexity of manipulating elections over social networks. Two orthogonal ingredients come to play here: (1) the mathematical structures modeling the interactions between the entities participating in the elections, and (2) how these mathematical structures define the interactions between these entities.

The mathematical structures. In Chapter 3 and Chapter 4, the mathematical structures studied are basically sets of voters or candidates that, to some extent, behave similarly. One future research direction is then to expand this research to additional mathematical structures. The most natural structures are maybe graphs, that is, general graphs or different graph classes. As another example, it might also be interesting to consider hypergraphs: hypergraphs are more general than graphs, and can model the well-motivated scenario where groups of voters or candidates interact with other groups of voters or candidates.

The interactions. In Chapter 3, each candidate is associated with a set of candidates, such that whenever a candidate is added to the election, each candidate in his associated set of candidates is also added to the election. This is, perhaps, the simplest way of defining interactions between candidates, based on these sets of candidates. A more complex type of interaction might be, for example, to assume that whenever a candidate is added to the election, each candidate in his associated set of candidates is added to the election with a certain probability.

Further, if some candidates are indeed added as a result, then the candidates in their associated sets are added to the election, with a certain, possibly diminishing, probability. More generally, it is worth studying how actions which manipulate elections propagate through social networks. For example, assume that a diffusion process begins whenever a voter changes her vote. This study is closely related to the study of diffusion in social networks, but where the information which is propagated is related to a given election.

From social networks to social choice. One of the main goals in Chapter 5 is to lift the existing research on anonymizing social networks to a different scenario; specifically, to a scenario where data regarding an election is to be published. Indeed, privacy is well-studied in the context of social networks, but less studied in the context of elections and social choice. As elections are interesting and important structures, there is certainly more room for research on privacy in elections.

Another example where a concept from social choice is used in the context of social networks is the study of degree anonymization by vertex cloning, described in Chapter 6: the concept of cloning is used quite extensively in social choice.

Indeed, bringing more concepts from social choice to the study of social networks and vice versa might be fruitful.

Finally, we take a last general perspective on the contents of this thesis, concentrating on the methods. This thesis is mainly concerned with parameterized algorithms and (to a smaller extent) approximation algorithms. It is natural to further investigate the approximability of the problems considered here. Another route to take is to consider randomized algorithms, which might sometimes return a wrong answer, or might have only probabilistic running time guarantees.

A further research direction is to study game-theoretic aspects of the problems considered here. The computational problems studied here assume only one agent, and concentrate on the hardness of achieving the goals of this agent. It might be interesting to assume several agents, and so, to study games of manipulation and anonymization in social choice and social networks.

8.2. Concluding Remarks

My aim in this thesis was to devise efficient algorithms and understand the computational complexity of some combinatorial problems regarding manipulation and anonymization in social choice and social networks. My goal was always two-fold: to devise efficient algorithms and to gain a better understanding of the structure of these problems.

It turned out that most of the computational problems I considered in this thesis are NP-hard (and sometimes even hard to approximate). Most of the times, in order to cope with this intractability, I considered parameterizations of these problems. Using the framework of parameterized complexity, I could devise parameterized algorithms for some special cases and gain an even better understanding of the structure of these problems.

From this point of view, each chapter can be seen as a quest for tractability; specifically, as a quest for finding special cases and parameters for which the problems considered become tractable, with the goal of devising efficient algorithms for these special cases and parameterizations.

Indeed, the framework of parameterized complexity fits well with the problems I considered. First, there are several natural parameters, motivated by real-world scenarios, for most of the problems. Second, most of these problems are of high intractability (indeed, some are even non-approximable), but some of them "break-down" when certain parameters are assumed to be small.

The combinatorial problems I considered are mathematical abstractions of real-world problems related to social choice and social networks. Social choice and social networks provide a fertile ground for scientific exploration, among other fields, also in the study of algorithms and computational complexity.

Hopefully, this thesis might serve as a bridge between research done on social choice (specifically on manipulating elections) and research done on social networks (specifically on anonymizing social networks) and might stimulate interesting future research.

I hope that this thesis also sheds light on the border between tractability and intractability for some interesting combinatorial problems regarding social choice and social networks, and that the results give some hope for efficient and exact algorithms, even for problems that seem highly intractable at first sight.

Bibliography

[1] Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. *Algorithmica*, 61(3): 638–655, 2011. (cited on 191)

[2] K. Arrow, A. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*. Elsevier, 2002. (cited on 19, 186)

[3] Takao Asano and Tomio Hirata. Edge-contraction problems. *Journal of Computer and System Sciences*, 26(2):197–208, 1983. (cited on 231)

[4] Tomer Ashur and Orr Dunkelman. Poster: On the anonymity of Israel's general elections. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13)*, pages 1399–1402. ACM, 2013. (cited on 170)

[5] John J. Bartholdi, III, Craig A. Tovey, and Michael A. Trick. How hard is it to control an election. *Mathematical and Computer Modelling*, 16(8–9): 27–40, 1992. (cited on 28, 31, 32, 35, 108)

[6] Dorothea Baumeister, Piotr Faliszewski, Jérôme Lang, and Jörg Rothe. Campaigns for lazy voters: Truncated ballots. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*, pages 577–584. IFAAMAS, 2012. (cited on 108)

[7] Cristina Bazgan and André Nichterlein. Parameterized inapproximability of degree anonymization. In *Proceedings of the 9th International Symposium on Parameterized and Exact Computation (IPEC '14)*, volume 8894 of *LNCS*, pages 75–84. Springer, 2014. (cited on 254)

[8] Rémy Belmonte, Petr A. Golovach, Pim van't Hof, and Daniël Paulusma. Parameterized complexity of three edge contraction problems with degree constraints. *Acta Informatica*, 51(7):473–497, 2014. (cited on 232)

[9] Nadja Betzler and Johannes Uhlmann. Parameterized complexity of candidate control in elections and related digraph problems. *Theoretical Computer Science*, 410(52):5425–5442, 2009. (cited on 32, 35, 38)

[10] Nadja Betzler, Jiong Guo, and Rolf Niedermeier. Parameterized computational complexity of Dodgson and Young elections. *Information and Computation*, 208(2):165–177, 2010. (cited on 32)

[11] Nadja Betzler, Robert Bredereck, Jiehua Chen, and Rolf Niedermeier. Studies in computational aspects of voting—a parameterized complexity perspective. In *The Multivariate Algorithmic Revolution and Beyond*, volume 7370 of *LNCS*, pages 318–363. Springer, 2012. (cited on 32)

[12] René van Bevern, Robert Bredereck, Jiehua Chen, Vincent Froese, Rolf Niedermeier, and Gerhard J. Woeginger. Network-based vertex dissolution. *SIAM Journal on Discrete Mathematics*, 29(2):888–914, 2015. (cited on 253)

[13] René van Bevern, Jiehua Chen, Falk Hüffner, Stefan Kratsch, Nimrod Talmon, and Gerhard J. Woeginger. Approximability and parameterized complexity of multicover by c-intervals. *Information Processing Letters*, 115(10):744–749, 2015. (cited on xii)

[14] Therese Biedl, Franz J. Brandenburg, and Xiaotie Deng. On the complexity of crossings in permutations. *Discrete Mathematics*, 309(7):1813–1823, 2009. (cited on 180)

[15] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*, pages 551–560. ACM, 2009. (cited on 191)

[16] Daniel Binkele-Raible, Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Transactions on Algorithms*, 8(4):38, 2012. (cited on 68)

[17] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artifical Intelligence Research*, 21:135–191, 2004. (cited on 32, 108)

[18] S. Brams and P. Fishburn. Voting procedures. In K. Arrow, A. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare, Volume 1*, pages 173–236. Elsevier, 2002. (cited on 8)

[19] Felix Brandt, Markus Brill, Edith Hemaspaandra, and Lane A. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*, pages 715–722, 2010. (cited on 31)

[20] Felix Brandt, Paul Harrenstein, Keyvan Kardel, and Hans Georg Seedig. It only takes a few: On the hardness of voting with a constant number of agents. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '13)*, pages 375–382. IFAAMAS, 2013. (cited on 28, 32)

[21] Robert Bredereck and Nimrod Talmon. NP-hardness of two edge cover generalizations with applications to control and bribery for approval voting. *Information Processing Letters*, 2015. (cited on xi)

[22] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Large-scale election campaigns: Combinatorial shift bribery. *Journal of Artifical Intelligence Research*, . (cited on xiii)

[23] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Complexity of shift bribery in committee elections. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI '16)*, . (cited on 130)

[24] Robert Bredereck, Sepp Hartung, André Nichterlein, and Gerhard J. Woeginger. The complexity of finding a large subgraph under anonymity constraints. In *Proceedings of the 24th International Symposium Algorithms and Computation (ISSAC '13)*, volume 8283 of *LNCS*, pages 152–162. Springer, 2013. (cited on 190)

[25] Robert Bredereck, André Nichterlein, and Rolf Niedermeier. Pattern-guided k-anonymity. *Algorithms*, 6(4):678–701, 2013. (cited on 170)

[26] Robert Bredereck, Jiehua Chen, Piotr Faliszewski, Jiong Guo, Rolf Niedermeier, and Gerhard J. Woeginger. Parameterized algorithmics for computational social choice: nine research challenges. *Tsinghua Science and Technology*, 19(4):358–373, 2014. (cited on 19)

[27] Robert Bredereck, Jiehua Chen, Piotr Faliszewski, André Nichterlein, and Rolf Niedermeier. Prices matter for the parameterized complexity of shift bribery. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI '14)*, pages 1398–1404, July 2014. (cited on 32, 106, 107, 108, 111, 118, 130)

[28] Robert Bredereck, Vincent Froese, Sepp Hartung, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon. The complexity of degree anonymization by vertex addition. In *Proceedings of the 10th International Conference on Algorithmic Aspects in Information and Management (AAIM '14)*, volume 8546 of *LNCS*, pages 44–55. Springer, 2014. (cited on viii, x, xii)

[29] Robert Bredereck, André Nichterlein, Rolf Niedermeier, and Geevarghese Philip. The effect of homogeneity on the computational complexity of combinatorial data anonymization. *Data Mining and Knowledge Discovery*, 28(1):65–91, 2014. (cited on 170)

[30] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. Elections with few candidates: Prices, weights, and covering problems. In *Proceedings of the 4th International Conference on Algorithmic Decision Theory (ADT '15)*, volume 9346 of *LNCS*, pages 414–431. Springer, 2015. (cited on xi, 137)

[31] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Large-scale election campaigns: Combinatorial shift bribery. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*, pages 67–75. IFAAMAS, 2015. (cited on vii, ix)

[32] Robert Bredereck, Vincent Froese, Sepp Hartung, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon. The complexity of degree anonymization by vertex addition. *Theoretical Computer Science*, 2015. (cited on xii)

[33] Laurent Bulteau, Jiehua Chen, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Combinatorial voter control in elections. *Theoretical Computer Science*, 589:99–120, 2015. (cited on xi, 28, 33, 108, 120, 123, 144)

[34] Laurent Bulteau, Stefan Fafianie, Vincent Froese, Rolf Niedermeier, and Nimrod Talmon. The complexity of finding effectors. In *Proceedings of the*

12th Annual Conference on Theory and Applications of Models of Computation (TAMC '15), volume 9076 of *LNCS*, pages 224–235. Springer, 2015. (cited on xi)

[35] Laurent Bulteau, Vincent Froese, and Nimrod Talmon. Multi-player diffusion games on graph classes. In *Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC '15)*, volume 9076 of *LNCS*, pages 200–211. Springer, 2015. (cited on xi)

[36] Leizhen Cai. Parameterized complexity of cardinality constrained optimization problems. *The Computer Journal*, 51(1):102–121, 2008. (cited on 245)

[37] Leizhen Cai, Siu Man Chan, and Siu On Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IPEC '06)*, volume 4169 of *LNCS*, pages 239–250. Springer, 2006. (cited on 249)

[38] David Cary. Estimating the margin of victory for instant-runoff voting. Presented at the 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections, 2011. (cited on 108)

[39] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicenç Torra. An algorithm for k-degree anonymity on large networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '13)*, pages 671–675. ACM, 2013. (cited on 190)

[40] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981. (cited on 169)

[41] Jiehua Chen, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Combinatorial voter control in elections. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS '14)*, volume 8635 of *LNCS*, pages 153–164. Springer, 2014. (cited on xi, xiii, 32, 108)

[42] Jiehua Chen, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Elections with few voters: Candidate control can be easy. In *Proceedings*

of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI '15), pages 2045–2051, 2015. (cited on vii, ix, xiii)

[43] Yiling Chen, Stephen Chong, Ian A. Kash, Tal Moran, and Salil Vadhan. Truthful mechanisms for agents that value privacy. In *Proceedings of the 14th ACM Conference on Electronic Commerce (EC '13)*, pages 215–232. ACM, 2013. (cited on 170)

[44] Sean Chester, Bruce M. Kapron, Ganesh Ramesh, Gautam Srivastava, Alex Thomo, and S. Venkatesh. Why Waldo befriended the dummy? k-anonymization of social networks with pseudo-nodes. *Social Network Analysis and Mining*, 3(3):381–399, 2013. (cited on 189, 190, 191, 192, 220)

[45] Sean Chester, Bruce M. Kapron, Gautam Srivastava, and S Venkatesh. Complexity of social network anonymization. *Social Network Analysis and Mining*, 3(2):151–166, 2013. (cited on 220)

[46] Kenneth L. Clarkson, Kun Liu, and Evimaria Terzi. Towards identity anonymization in social networks. In *Link Mining: Models, Algorithms, and Applications*, pages 359–385. Springer, 2010. (cited on 190)

[47] Chris Clifton and Tamir Tassa. On syntactic anonymity and differential privacy. In *Proceedings of the 29th International Conference on Data Engineering Workshops (ICDE Workshops '13)*, pages 88–93. IEEE, 2013. (cited on 168)

[48] Vincent Conitzer. Should social network structure be taken into account in elections? *Mathematical Social Sciences*, 64(1):100–102, 2012. (cited on 32)

[49] Vincent Conitzer. The maximum likelihood approach to voting on social networks. In *Proceedings of the 51st Annual Allerton Conference on Communication, Control, and Computing (ALLERTON '13)*, pages 1482–1487. IEEE, 2013. (cited on 32)

[50] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):14, 2007. (cited on 28)

[51] Vincent Conitzer, Jérôme Lang, and Lirong Xia. How hard is it to control sequential elections via the agenda? In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '10)*, pages 103–108. AAAI Press, 2009. (cited on 108)

[52] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS '13)*, volume 8134 of *LNCS*, pages 481–498. Springer, 2013. (cited on 170)

[53] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Parameterized algorithms, 2015. (cited on 14)

[54] Daniel Delling, Robert Görke, Christian Schulz, and Dorothea Wagner. Orca reduction and contraction graph clustering. In *Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management (AAIM '09)*, volume 5564 of *LNCS*, pages 152–165. Springer, 2009. (cited on 231)

[55] Reinhard Diestel. *Graph Theory*. Springer, 2010. (cited on 17, 237)

[56] Britta Dorn and Ildikó Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 64(1):126–151, 2012. (cited on 32, 108)

[57] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. (cited on 14, 15, 22, 23, 40, 124, 128, 145, 197, 198, 221, 244, 245)

[58] Bhaskar Dutta, Matthew O. Jackson, and Michel Le Breton. Strategic candidacy and voting procedures. *Econometrica*, 69(4):1013–1037, 2001. (cited on 30)

[59] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2013. (cited on 168)

[60] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pages 613–622. ACM, 2001. (cited on 29, 180)

[61] Edith Elkind and Piotr Faliszewski. Approximation algorithms for campaign management. In *Proceedings of the 6th International Workshop On Internet And Network Economics (WINE '10)*, volume 6484 of *LNCS*, pages 473–482. Springer, 2010. (cited on 106, 107, 108, 111, 118, 119, 137, 138, 139)

[62] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Swap bribery. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT '09)*, volume 5814 of *LNCS*, pages 299–310. Springer, 2009. (cited on 106, 107, 108, 111, 118, 119, 137, 138)

[63] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Distance rationalization of voting rules. *Social Choice and Welfare*, pages 1–33, 2010. (cited on 185)

[64] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. On the role of distances in defining voting rules. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '10)*, pages 375–382. IFAAMAS, 2010. (cited on 168, 170, 179, 181)

[65] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Cloning in elections: Finding the possible winners. *Journal of Artificial Intelligence Research*, 42:529–573, 2011. (cited on 35)

[66] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Clone structures in voters' preferences. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC '12)*, pages 496–513, 2012. (cited on 191)

[67] Gábor Erdélyi, Markus Nowak, and Jörg Rothe. Sincere-strategy preference-based approval voting fully resists constructive control and broadly resists destructive control. *Mathematical Logic Quarterly*, 55(4): 425–443, 2009. (cited on 32)

[68] Gábor Erdélyi, Michael R. Fellows, Jörg Rothe, and Lena Schend. Control complexity in Bucklin and fallback voting: A theoretical analysis. *Journal of Computer and System Sciences*, 81:632–660, 2014. (cited on 31, 75)

[69] Gábor Erdélyi, Michael R Fellows, Jörg Rothe, and Lena Schend. Control complexity in Bucklin and fallback voting: An experimental analysis. *Journal of Computer and System Sciences*, 81:661–670, 2014. (cited on 31, 75)

[70] Gabor Erdélyi, Edith Hemaspaandra, and Lane A. Hemaspaandra. More natural models of electoral control by partition. In *Proceedings of the 4th International Conference on Algorithmic Decision Theory (ADT '15)*, volume 9346 of *LNCS*, pages 396–413. Springer, 2015. (cited on 32, 33)

[71] Paul Erdős and Paul Kelly. The minimal regular graph containing a given graph. *American Mathematics Monthly*, 70:1074–1075, 1963. (cited on 200, 202)

[72] Paul Erdős and Tibor Gallai. Graphs with prescribed degrees of vertices (in Hungarian). *Matematikai Lapok*, 11:264–274, 1960. (cited on 202)

[73] Piotr Faliszewski and Jörg Rothe. Control and bribery in voting. In Felix Brandt et al., editor, *Handbook of Computational Social Choice*, chapter 7. Cambridge University Press, 2015. To appear. (cited on 28, 31)

[74] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. How hard is bribery in elections? *Journal of Artificial Intelligence Research*, 35: 485–532, 2009. (cited on 106, 107)

[75] Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Llull and Copeland Voting Computationally Resist Bribery and Constructive Control. *Journal of Artificial Intelligence Research*, 35: 275–341, 2009. (cited on 28, 32, 35, 38, 107)

[76] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Using complexity to protect elections. *Communications of the ACM*, 53(11): 74–82, 2010. (cited on 28, 31)

[77] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Multimode control attacks on elections. *Journal of Artificial Intelligence Research*, 40:305–351, 2011. (cited on 28, 32, 35, 38, 76, 102)

[78] Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. *Information and Computation*, 209(2):89–107, 2011. (cited on 31)

[79] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. The complexity of manipulative attacks in nearly single-peaked electorates. *Artificial Intelligence*, 207:69–99, 2014. (cited on 35)

[80] Piotr Faliszewski, Yannick Reisch, Jörg Rothe, and Lena Schend. Complexity of manipulation, bribery, and campaign management in Bucklin and fallback voting. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '14)*, pages 1357–1358. IFAAMAS, 2014. (cited on 108)

[81] Michael R. Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. (cited on 22)

[82] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. (cited on 14, 15)

[83] Harold N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC '83)*, pages 448–456. ACM, 1983. (cited on 140)

[84] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979. (cited on 15, 22, 23, 53, 61, 112, 132, 196, 239, 242, 244, 246)

[85] Petr A. Golovach, Pim van't Hof, and Daniël Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013. (cited on 231)

[86] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. (cited on 22, 176)

[87] Sylvain Guillemot and Dániel Marx. A faster FPT algorithm for bipartite contraction. *Information Processing Letters*, 113(22):906–912, 2013. (cited on 231)

[88] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38:31–45, 2007. (cited on 17)

[89] Louis S. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. *Journal of the Society for Industrial & Applied Mathematics*, 10(3):496–506, 1962. (cited on 202)

[90] Sepp Hartung and Nimrod Talmon. The complexity of degree anonymization by graph contractions. In *Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC '15)*, volume 9076 of *LNCS*, pages 260–271. Springer, 2015. (cited on viii, x, xii)

[91] Sepp Hartung, André Nichterlein, Rolf Niedermeier, and Ondřej Suchỳ. A refined complexity analysis of degree anonymization in graphs. *Information and Computation*, 243:249–262, 2015. (cited on 190, 237, 238, 239, 246)

[92] Vaclav Havel. A remark on the existence of finite graphs. *Časopis pro Pěstování Matematiky*, 80(477-480):1253, 1955. (cited on 202)

[93] Noam Hazon, Raz Lin, and Sarit Kraus. How to change a group's collective decision? In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13)*, pages 198–205. AAAI Press, 2013. (cited on 107)

[94] Pinar Heggernes, Pim Van't Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul. Contracting graphs to paths and trees. *Algorithmica*, 68(1):109–132, 2014. (cited on 231)

[95] Edith Hemaspaandra, Lane Hemaspaandra, and Jörg Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(5–6):255–285, 2007. (cited on 31, 32, 35)

[96] Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Hybrid elections broaden complexity-theoretic resistance to control. *Mathematical Logic Quarterly*, 55(4):397–424, 2009. (cited on 31)

[97] Lane A. Hemaspaandra, Rahman Lavaee, and Curtis Menton. Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '13)*, pages 1345–1346. IFAAMAS, 2013. (cited on 28, 32)

[98] Danny Hermelin, Judith Kubitza, Nimrod Talmon, Dvir Shabtay, and Gerhard J. Woeginger. Scheduling two competing agents when one agent has significantly fewer jobs. In *Proceedings of the 10th International Symposium on Parameterized and Exact Computation (IPEC '15)*, 2015. (cited on xii)

[99] Heather Hulett, Todd G. Will, and Gerhard J. Woeginger. Multigraph realizations of degree sequences: Maximization is easy, minimization is hard. *Operations Research Letters*, 36(5):594–596, 2008. (cited on 132, 242)

[100] Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113:58–97, 2014. (cited on 17, 191)

[101] Jérôme Lang and Lirong Xia. Voting in combinatorial domains. In Felix Brandt et al., editor, *Handbook of Computational Social Choice*, chapter 9. Cambridge University Press, 2015. To appear. (cited on 108)

[102] Jérôme Lang, Nicolas Maudet, and Maria Polukarov. New results on equilibria in strategic candidacy. In *Proceedings of the 6th International Symposium on Algorithmic Game Theory (SAGT'13)*, volume 8146 of *LNCS*, pages 13–25. Springer, 2013. (cited on 30)

[103] Jean-Francois Laslier. *Tournament Solutions and Majority Voting*. Springer, 1997. (cited on 169)

[104] Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983. (cited on 16, 99, 136, 143, 182, 184, 210, 212)

[105] Hong Liu and Daming Zhu. Parameterized complexity of control problems in maximin election. *Information Processing Letters*, 110(10):383–388, 2010. (cited on 32)

[106] Hong Liu, Haodi Feng, Daming Zhu, and Junfeng Luan. Parameterized computational complexity of control problems in voting systems. *Theoretical Computer Science*, 410(27):2746–2753, 2009. (cited on 32)

[107] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*, pages 93–106. ACM, 2008. (cited on 190, 208)

[108] Andrea Loreggia, Nina Narodytska, Francesca Rossi, Brent K. Venable, and Toby Walsh. Controlling elections by replacing candidates: Theoretical and experimental results. In *Proceedings of the 8th Multidisciplinary Workshop on Advances in Preference Handling*, 2014. (cited on 35, 38)

[109] Xuesong Lu, Yi Song, and Stéphane Bressan. Fast identity anonymization on graphs. In *Proceedings of the 23th International Conference on Database and Expert Systems Applications (DEXA '12)*, volume 7446 of *LNCS*, pages 281–295. Springer, 2012. (cited on 190)

[110] George S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical report, Computer Science Laboratory, Princeton University, 1975. (cited on 23, 204)

[111] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD '07)*, 1(1):3, 2007. (cited on 168, 186)

[112] Krzysztof Magiera and Piotr Faliszewski. How hard is control in single-crossing elections? In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI '14)*, pages 579–584. IOS Press, August 2014. (cited on 31)

[113] Thomas R. Magrino, Ronald L. Rivest, Emily Shen, and David Wagner. Computing the margin of victory in IRV elections. Presented at the 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections, 2011. (cited on 108)

[114] Luke Mathieson and Stefan Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. *Journal of Computer and System Sciences*, 78(1):179–191, 2012. (cited on 150)

[115] Nicholas Mattei, Maria Silvia Pini, Brent K. Venable, and Francesca Rossi. Bribery in voting over combinatorial domains is easy. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*, pages 1407–1408. IFAAMAS, 2012. (cited on 107, 108)

[116] Nicholas Mattei, Maria Silvia Pini, Francesca Rossi, and Brent K. Venable. Bribery in voting with CP-nets. *Annals of Mathematics and Artificial Intelligence*, 68(1-3):135–160, 2013. (cited on 32)

[117] Nicholas Mattei, Judy Goldsmith, Andrew Klapper, and Martin Mundhenk. On the complexity of bribery and manipulation in tournaments with uncertain information. *Journal of Applied Logic*, 2015. (cited on 107)

[118] Reshef Meir, Ariel D. Procaccia, Jeffrey S. Rosenschein, and Aviv Zohar. Complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research*, 33:149–178, 2008. (cited on 31)

[119] Curtis Menton. Normalized range voting broadly resists control. *Theory of Computing Systems*, 53(4):507–531, 2013. (cited on 32)

[120] Curtis Menton and Preetjot Singh. Control complexity of Schulze voting. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13)*, pages 286–292. AAAI Press, 2013. (cited on 31, 32)

[121] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 223–228. ACM, 2004. (cited on 170)

[122] Jayadev Misra and David Gries. A constructive proof of Vizing's theorem. *Information Processing Letters*, 41(3):131–133, 1992. (cited on 54)

[123] Elchanan Mossel, Joe Neeman, and Omer Tamuz. Majority dynamics and aggregation of information in social networks. *Autonomous Agents and Multi-Agent Systems*, 28(3):408–429, 2014. (cited on 33)

[124] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010. (cited on 8, 253)

[125] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. (cited on 14)

[126] Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS '10)*, pages 17–32, 2010. (cited on 15)

[127] Hannu Nurmi, Arto Salomaa, and Lila Santean. Secret ballot elections in computer networks. *Computers & Security*, 10(6):553–560, 1991. (cited on 169)

[128] James G. Oxley. *Matroid Theory*. Oxford University Press, 2006. (cited on 231, 253)

[129] David C. Parkes and Lirong Xia. A complexity-of-strategic-behavior comparison between Schulze's rule and ranked pairs. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI '12)*, pages 1429–1435, 2012. (cited on 31)

[130] Maria Polukarov, Svetlana Obraztsova, Zinovi Rabinovich, Alexander Kruglyi, and Nicholas R. Jennings. Convergence to equilibria in strategic candidacy. In *the 2nd Workshop on Exploring Beyond the Worst Case in Computational Social Choice. Held as part of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, 2015. (cited on 30)

[131] Ariel D. Procaccia, Nisarg Shah, and Eric Sodomka. Ranked voting on social networks. In *ijcai15*, 2015. (cited on 32)

[132] Sridhar Rajagopalan and Vijay V. Vazirani. Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, 28(2):525–540, 1998. (cited on 139)

[133] Yannick Reisch, Jörg Rothe, and Lena Schend. The margin of victory in Schulze, Cup, and Copeland elections: Complexity of the regular and exact variants. In *Proceedings of the Seventh European Starting AI Researcher Symposium (STAIRS '14)*, volume 264 of *Frontiers in Artificial Intelligence and Applications*, pages 250–259. IOS Press, 2014. (cited on 108)

[134] Jörg Rothe and Lena Schend. Challenges to complexity shields that are supposed to protect elections against manipulation and control: a survey. *Annals of Mathematics and Artificial Intelligence*, 68(1–3):161–193, 2013. (cited on 31)

[135] Alexander Schäfer, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5):883–891, 2012. (cited on 68)

[136] Ildikó Schlotter, Piotr Faliszewski, and Edith Elkind. Campaign management under approval-driven voting rules. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI '11)*, pages 726–731, 2011. (cited on 108)

[137] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012. (cited on 1)

[138] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10:557–570, 2002. (cited on 168, 170)

[139] Nimrod Talmon. Privacy in elections: k-anonymizing preference orders. In *Proceedings of the International Symposium on Fundamentals of Computation Theory (FCT '15)*, volume 9210 of *LNCS*, pages 299–310. Springer, 2015. (cited on viii, ix, xiii)

[140] Johan M. M. van Rooij, Marcel E. van Kooten Niekerk, and Hans L. Bodlaender. Partition into triangles on bounded degree graphs. *Theory of Computing Systems*, 52(4):687–718, 2013. (cited on 23, 247)

[141] Vadim G. Vizing. Critical graphs with a given chromatic class. *Metody Diskretnogo Analiza*, 5(1):9–17, 1965. (cited on 54)

[142] Thomas Wolle and Hans L. Bodlaender. A note on edge contraction. Technical report, Technical Report UU-CS-2004, 2004. (cited on 237)

[143] Lirong Xia. Computing the margin of victory for various voting rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC '12)*, pages 982–999. ACM Press, 2012. (cited on 108)

[144] Lirong Xia and Vincent Conitzer. Strategy-proof voting rules over multi-issue domains with restricted preferences. In *Proceedings of the 6th International Workshop on Internet and Network Economics (WINE '10)*, volume 6484 of *LNCS*, pages 402–414. Springer, 2010. (cited on 32)

Schriftenreihe **Foundations of computing**
Hrsg.: Prof. Dr. Stephan Kreutzer, Prof. Dr. Uwe Nestmann, Prof. Dr. Rolf Niedermeier
ISSN 2199-5249 (print)
ISSN 2199-5257 (online)

01: **Bevern, René van: Fixed-Parameter Linear-Time Algorithms for NP-hard Graph and Hypergraph Problems Arising in Industrial Applications.** - 2014. - 225 S.
ISBN **978-3-7983-2705-4** (print) EUR **12,00**
ISBN **978-3-7983-2706-1** (online)

02: **Nichterlein, André: Degree-Constrained Editing of Small-Degree Graphs.** - 2015. - xiv, 225 S.
ISBN **978-3-7983-2705-4** (print) EUR **12,00**
ISBN **978-3-7983-2706-1** (online)

03: **Bredereck, Robert: Multivariate Complexity Analysis of Team Management Problems.** - 2015. - xix, 228 S.
ISBN **978-3-7983-2764-1** (print) EUR **12,00**
ISBN **978-3-7983-2765-8** (online)

Universitätsverlag der TU Berlin

Algorithmic Aspects of Manipulation and Anonymization in Social Choice and Social Networks

This thesis presents a study of several combinatorial problems related to social choice and social networks. The main concern is their computational complexity, with an emphasis on their parameterized complexity. As most of these problems are computationally intractable (that is, NP-hard or even hard to approximate), some restricted cases and parameterizations for these problems are considered. The goal is to devise exact algorithms for these problems, which are efficient for considered special cases or when the considered parameters are small, or to prove that, under widely-accepted assumptions, such algorithms cannot exist. This kind of study allows the exploration of some boundaries between tractability and intractability. One type of problems studied in this thesis is about manipulating elections which occur on top of underlying social networks, connecting either the voters participating in these elections or the candidates that the voters vote on. The other type of problems studied in this thesis is concerned with preserving the privacy of the entities of a social network, when the structure of the network is to be published, or the privacy of the voters of an election, when the preferences of the voters are to be published.

ISBN 978-3-7983-2804-4 (print)
ISBN 978-3-7983-2805-1 (online)

ISBN 978-3-7983-2804-4

http://verlag.tu-berlin.de