AGH University of Science and Technology
Faculty of Computer Science, Electronics and Telecommunications
Department of Computer Science

Algorithms for and Computational Complexity of the Election Control Problems in Restricted Domains

Krzysztof Magiera

PhD Thesis

Supervisor: dr hab. inż. Piotr Faliszewski, prof. AGH

Kraków, 2020

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie
Wydział Informatyki Elektroniki i Telekomunikacji
Katedra Informatyki

Rozprawa doktorska

Algorytmy i złożoność obliczeniowa problemów kontroli wyborów z zawężoną dziedziną preferencji

Krzysztof Magiera

Promotor: dr hab. inż. Piotr Faliszewski, prof. AGH

Kraków, 2020

Abstract

In this thesis we discuss the complexity of election control problems under restricted preference domains. We analyze a wide spectrum of settings, a number of different voting rules, focusing on several well-known domain restrictions, such as single-peakedness, single-crossingness, and top-monotonicity.

Election control problems model scenarios in which a malicious agent wants to change an election outcome by altering the setting under which the election is held (e.g., by changing the set of agents or modifying the set of alternatives). When studying the complexity of such problems, we ask how hard it is to tell whether a given control method can be successful for a selected voting system. In general, for a number of most popular voting systems and control methods, the problems are NP-complete. However, as the existing hardness results focus solely on the worst case complexity, they do not give us insights on how the complexity of control looks like in real-life settings. One of the ways how we can model and analyze more realistic election scenarios is by considering restricted domains. Under restricted domains, we require the votes to have a specific shape, as opposed to considering all the possible preferences. As a result, for the single-peaked domain (arguably the best studied restricted domain) a lot of election control problems become polynomial-time solvable.

In this thesis we pursue this research direction further, discussing three restricted domains: single-peakedness, single-crossingness, and top-monotonicity. We start our discussion by considering the domain recognition problem, that is, the problem of telling whether a given election profile meets the given restricted domain criteria. We present a novel approach to domain recognition that works by reducing the problem to solving SAT-2CNF. As the new approach adapts to top-monotonic profiles well, it constitutes the first polynomial-time algorithm for recognizing top-monotonic profiles.

Next, we move on to examining different variants of election control problem under restricted domains. We start by presenting polynomial-time results for control by adding/deleting agents/alternatives under Plurality, Approval and Condorcet voting under the single-crossing domain. Then, we discuss counting variants of election control problems, which, for example, can be used as a way to predict election winners. We show that for Plurality, k-Approval and Condorcet voting under the single-peaked domain the counting variants of election control by adding/deleting agents/alternatives can be solved in polynomial-time.

Lastly, we discuss the election control problems for elections where we seek a committee of alternatives (so-called *multiwinner* elections). As the field of multiwinner elections is not as well studied as the single winner case, we provide an initial set of results for the unrestricted cases. In particular, we show that a number of control problems are NP-hard for the unrestricted Approval Voting and Satisfaction Approval Voting rules.

Streszczenie

W poniższej rozprawie doktorskiej podejmujemy temat złożoności obliczeniowej problemów kontroli wyborów z zawężoną dziedziną preferencji. Dyskutujemy szereg konfiguracji poprzez rozpatrywanie różnych metod wyborczych w kontekście kilku najbardziej popularnych kryteriów zawężania dziedziny: preferencji jednowierzchołkowych, preferencji z pojedynczym przecięciem, oraz preferencji spełniających warunek top-monotonicity.

W problemach kontroli wyborów rozpatrujemy scenariusz, w którym poprzez działanie czynników zewnętrznych możliwy jest wpływ na wynik głosowania – zmiana taka może być następstwem zmiany listy kandydatów czy konkretnych głosów, które zostaną przyjęte podczas głosowania. Przy badaniu złożoności obliczeniowej problemów kontroli mamy zadany sposób wybierania zwycięzcy oraz dozwoloną metodę kontroli (np. usuwanie kandydatów). Traktując listę kandydatów oraz preferencje wyborców jako dane wejściowe, pytamy jak trudne obliczeniowo jest stwierdzenie czy możliwe jest aby wybrany przez nas kandydat wygrał, wykonując ustaloną liczbę modyfikacji (możliwe jest również rozpatrywanie kontroli w której chcemy doprowadzić do tego aby wybrany kandydat nie wygrał). Dla dużej części popularnych systemów wyborczych i metod kontroli problemy te okazują się być NP-zupełne. Istotnym czynnikiem jest tutaj jednak fakt, że analiza złożoności obliczeniowej dotyczy przypadków pesymistycznych. Wiemy zatem, że dla wielu spośród popularnych systemów wyborczych i metod manipulacji istnieją konfiguracje, dla których stwierdzenie czy możemy wpłynąć na wynik jest bardzo skomplikowane obliczeniowo – nie mamy jednak pewności czy takie pesymistyczne konfiguracje w praktyce będą się w ogóle pojawiać. Aby odnieść się do tego zagadnienia musimy postarać się znaleźć sposób na modelowanie bardziej realistycznych profili wyborczych – jest to w istocie celem omawianych w tej rozprawie zawężonych dziedzin preferencji. Zawężanie dziedziny polega tutaj na wprowadzeniu odgórnych ograniczeń na kształt preferencji głosujących. Jedną z najbardziej popularnych i najczęściej omawianych metod zawężania dziedziny preferencji jest założenie o jednowierzchołkowości — zakładamy w niej, że istnieje liniowe uszeregowanie kandydatów, względem którego preferencje głosujących maleją jednostajnie w obu kierunkach począwszy od najbardziej preferowanego kandydata przez danego głosującego. Jeśli narzucimy takie dodatkowe kryterium na zbiór rozpatrywanych profili wyborczych okazuje się, że duża część problemów kontroli, które w ogólnym przypadku były trudne obliczeniowo, może zostać rozwiązana w czasie wielomianowym.

W poniższej rozprawie koncentrujemy się na kontynuacji idei badania trudności problemów kontroli poprzez wprowadzanie ograniczeń na kształt profili. Staramy się odpowiedzieć na pytanie czy biorąc pod uwagę inne metody zawężania dziedziny preferencji otrzymamy podobne rezultaty co w przypadku jednowierzchołkowości – w tym celu omawiamy również kryteria pojedynczego przecięcia oraz top-monotonicity. W pierwszej części rozprawy skupiamy się nad problemem rozpoznawania czy dany profil spełnia kryteria danej metody zawężania dziedziny preferencji – jeśli nie jesteśmy w stanie w czasie wielomianowym stwierdzić czy dany profil spełnia dane kryterium, to również, nawet jeśli istnieje, nie wiemy kiedy wykorzystać „szybszy" algorytm, dostosowany do działania dla danego kryterium. W rozprawie prezentujemy nowatorską metodę rozpoznawania zawężonych dziedzin, która oparta jest na sprowadzeniu problemu rozpoznawania

do problemu spełnialności SAT-2CNF. Prócz wykorzystania naszego podejścia do rozpoznawania kryterium jednowierzchołkowości i pojedynczego przecięcia, dla których znane są już wielomianowe metody rozpoznawania, pokazujemy również, że może być zastosowane przy rozpoznawaniu kryterium top-monotonic, dla którego wielomianowy algorytm nie był wcześniej znany.

W kolejnych rozdziałach przechodzimy do omawiania konkretnych wariantów problemów kontroli z zawężoną dziedziną preferencji – omawiamy warianty kontroli przez dodawanie i usuwanie kandydatów i głosujących, rozpatrując głosowanie większościowe, aprobatowe oraz metodę Condorcet, przy kryterium pojedynczego przecięcia. Następnie rozpatrujemy rozszerzenie decyzyjnego problemu kontroli polegające na zliczaniu możliwych scenariuszy, w których wybrany kandydat wygrywa. Pokazujemy algorytmy wielomianowe rozwiązujące problem zliczania w kontroli dla głosowania większościowego, k-Approval oraz dla metody Condorceta przy kryterium jednowierzchołkowości.

W końcowej części rozprawy poruszamy temat kontroli dla metody wyborczych których celem jest wyłonienie grupy zwycięzców (tzw. *multiwinner elections*). Ze względu na fakt, że metody multiwinner w ostatnich czasach dopiero zyskują na popularności, na moment powstawania tej pracy nie istniały wyniki dla tych metod nawet bez narzucania dodatkowych kryteriów na profile głosów. Prezentujemy tutaj dowody NP-trudności dla wybranych sposobów kontroli przy metodach multiwinner: Approval Voting oraz Satisfaction Approval Voting.

Contents

Chapter 1

Introduction

Our goal in this thesis is to analyze the complexity of election control problems—i.e., of problems where the goal is to change an election's result by modifying its structure in a given way—under the assumption that the votes come from various restricted domains. We consider election control from several different angles, by approaching different control scenarios, analyzing their decision and counting variants, and examining both single and multiwinner settings. We also show how to recognize the votes coming from several restricted domains and how these domain restrictions can impact the complexity of control problems.

1.1 Overview of Elections and Election Control

Elections provide a way of aggregating various, perhaps conflicting preferences and, thus, are one of the most common tools for supporting collective decision making. For example, societies use elections to choose their leaders and representatives, such as presidents or members of parliaments. Further, the selected parliaments refer to voting when working on bills or establishing new laws. Similarly, voting is used as a mean of making decisions in companies, where a number of stockholders can represent different interests. Elections are also commonly used to select winners of various contests, where the judges or the audience can cast their votes or rate the contestants (e.g., consider selecting the most valuable player in basketball leagues). Elections and voting are also useful in computer science, where they can be applied in multiagent-systems as a framework for agents to collaborate [26, 32, 39, 71]

As seen above, elections are ubiquitous—this has made them the focus of many studies from the fields of political science, economics or mathematics, but also from the computational perspective in social choice, artificial intelligence and operations

research. The common model of elections, applicable across all these fields, is that we are given a set of agents (voters) that express their preferences over a set of alternatives (candidates). We are also given a voting rule which selects a winner (or a committee of winners) based on these preferences. Depending on the setting, we may consider each agent's preferences to be a set of alternatives they approve (dichotomous preferences), or we may assume that each agent ranks the candidates from best to worst (ordinal preferences). In the former scenario, the agent rates all the approved agents equally, whereas in the latter case we have a fine-grained spectrum of appreciation.

With so many different use cases in various areas, it is important to remember that elections can often be abused. A possible scenario is that a malicious actor wants to manipulate the outcome of the election (e.g., make a particular person win a contest). There are many means by which they may wish to achieve this goal. In this thesis we focus on election control where this malicious actor is capable of altering the structure of the election, i.e., the sets of agents or alternatives (this models settings where the malicious actor is, e.g., organizing the elections, or plays some important role in them). For example, they may encourage some people who would have not voted otherwise to cast their votes, or they may prevent a contestant from taking part in the contest. We consider both the constructive variants of election control, where we are given a designated alternative that we want to win the election, and the destructive variants, where we want to ensure that a designated alternative does not win. When discussing election control problems, we typically refer to situations where we are given a complete information about the election, that is, both the sets of alternatives and the agents, their preferences, and the voting rule. On top of that, it is specified what type of control we want to test—it could be either adding or deleting agents, or adding or deleting alternatives, in either the constructive or destructive variants. Additionally, we are given a designated alternative p, and a limit k for the number of changes that we can make (the number of added or deleted agents/alternatives). Given all of that, in an election control problem we ask if it is possible to achieve the specified goal, i.e., ensure that the designated candidate becames a winner (ceases being a winner) by making at most k of the allowed changes. For example, in Constructive Control by Deleting Voters (CCDV)[1] we are given a set of agents, a set of alternatives, a designated alternative p, a number k,

[1]For consistency reasons we use well established naming of the control problems that refers to voters and candidates (e.g., Constructive Control by Deleting Voters or Destructive Control by Adding Candidates) while in this thesis we often use corresponding terms of agents and alternatives.

and we ask if it is possible for p to became a winner in this election by removing at most k agents.

The complexity of election control problems has been studied deeply for various voting rules. The seminal work in this area, presented by Bartholdi et al. [8], shows that a number of control problems for selected voting rules are NP-hard. One of the possible consequences of this computational intractability is that it may be difficult to manipulate the results of elections held under these rules. For example, if given all voters' preferences it is hard to tell if by manipulating the election our designated candidate can win, then the task of determining the actual manipulations that we need to make will be at least as hard, and can get even harder if we consider that the manipulator may not have the full knowledge of the election setting.

The biggest flaw in the above reasoning is that we are assuming that typical elections that may arise in real-world settings yield computationally hard election control problem instances. Unfortunately, this does not seem to be the case. The NP-hardness proofs rely on performing reductions from well known NP-hard problems. That is, we show that each instance of a selected NP-hard problem can be mapped (in polynomial-time) into an instance of the problem that we want to prove NP-hard. Once we find the mapping, we argue that since the NP-hard problem can be solved by expressing it in terms of our control problem, then this control problem is NP-hard too. This is correct, but what we know after performing a reduction is that there exists a group of instances of the control problem that are hard to solve. We do not know how likely it is that such instances occur in real-life scenarios. Moreover, the reduction does not give us any information about the remaining instances, not produced by the mapping, and we do not know if they are also hard to solve or how many such difficult-to-solve instances exist outside of our mapping spectrum.

One of the ways that helps with understanding the real-life complexity of election control problems is to more strictly model the inputs. This is where the concept of restricted domains comes into play. The idea of restricting the domain of agents' preferences is to capture these, which are likely to occur in reality. One of the most broadly studied restricted domains is the notion of *single-peakedness*, introduced by Black [12]. When an ordinal election is single-peaked, it means that there exists a linear order of alternatives that we refer to as the *societal axis*. When considering each agent's preference over the societal axis, it decreases towards both ends of the axis starting from the most preferred alternative, therefore creating a shape with a single peak.

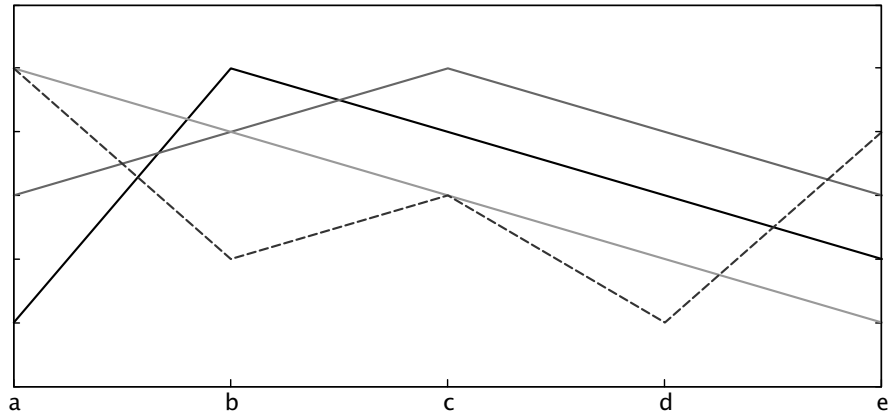

Figure 1.1: Three possible single-peaked preference orders plotted using solid lines along the societal axis ($b \succ c \succ d \succ e \succ a$; $a \succ b \succ c \succ d \succ e$; $c \succ b \approx d \succ e \approx a$), and one preference order (marked with dashed line) that is not single-peaked along the given societal axis ($a \succ e \succ c \succ b \succ d$).

As we can see, the single-peaked restriction limits the number of ways in which an election can be formed for a given number of alternatives and agents. One of the interesting aspects of single-peakedness is that it models some real-life scenarios. For example, we can imagine that the extreme candidates of the societal axis represent the most left-wing candidate and the most right-wing candidate, while the rest of the axis, in between, is ordered based on how close are a given candidate's views to the left- or the right-wing one. Now, each agent's top choice is the candidate that has the closest views to that agent's, and the agent's preference among other candidates monotonically decreases towards both sides of the axis.

Besides the fact that restricted domains model more natural relationships between the agents and the alternatives, they also prevent some problematic settings from arising in the election. For example, elections that are single-peaked are free from the so-called *Condorcet paradox*. This paradox refers to the case where there exist cycles in the collective preferences of the voters. For example, consider three alternatives a, b and c, such that the majority of the agents prefer a to b, another majority prefers b to c, and yet another majority prefers c to a. This can occur despite the fact that such cycles are not possible in each agent's individual preference orders. The existence of Condorcet cycles makes it difficult to argue about the fairness of selected preference aggregation and, hence, modelling elections where such paradox cannot occur is desirable.

Setting rules on how votes can be structured in a form of a restricted domain, such as the single-peaked domain, turns out to have a significant impact on the complexity

of the election control problems. As first shown by Faliszewski et al. [37], many of the original NP-hard results of Bartholdi et al. [8] do not hold when considering single-peaked electorates. An important consequence of this result it that we cannot rely on the election control being hard in an unrestricted setting as a way of safeguarding manipulation. Indeed, we see that in many settings that can arise naturally, the control problems can be solved in polynomial-time.

1.2 Contributions

The unrestricted cases and the single-peaked cases are just two sides of a spectrum of election classes that we can analyze to broaden our understanding of the complexity of election control problems. The main direction we take in this thesis is to discuss other classes of restricted domains, namely: single-crossingness and top-monotonicity. Below we present an overview of the variants of those problems that we consider in the further chapters.

In Chapter 3, we discuss the problem of recognizing restricted profiles, that is, given an election we want to tell if it matches the criteria of the selected restricted domain and, if so, to determine the features of this domain (e.g., find the societal axis in the case of the single-peaked domain). The profile recognition problem is fundamental to any further study on the complexity of control problems under the given preference restriction. Indeed, deciding if the agents in a given election fit a given restriction is the initial step of the algorithms for control problems. Therefore if recognition turns out to be hard, this discredits results for control problems under this domain. Thankfully, as presented in Chapter 3, all the restricted domains we discuss can be recognized in polynomial-time. Our main contribution is a novel approach to solving domain recognition problems that we use to recognize single-peaked and single-crossing profiles, but also to provide the first recognition algorithm for top-monotonic profiles (the polynomial-time algorithms for single-peaked and single-crossing domains were known). Unlike many of the existing recognition algorithms for single-peaked or single-crossing domains, which rely on solving the consecutive ones problem, our approach depends on reducing the recognition problem to solving SAT-2CNF. This technique is powerful enough to let us derive an algorithm for recognizing top-monotonic preferences, which was elusive for several years [2].

In Chapter 4 we move on to analyzing control problems under the single-crossing domain. We present tractability results for the following control problems, known to be NP-hard in the unrestricted case:

1. constructive and destructive control by adding and deleting candidates for the Plurality voting rule,

2. constructive control by adding and deleting voters for Condorcet and Approval rules.

Next, in Chapter 5 we discuss counting variants of election control problems. In the counting variants, instead of asking whether control is possible under certain circumstances, we ask if given the fact that control is possible, how many different ways are there to achieve it (e.g., in how many ways can we delete k voters). We focus on the single-peaked domain in this chapter, with our main contribution being polynomial-time algorithms for the Plurality, k-Approval, and Condorcet voting rules.

Finally, in Chapter 6 we discuss control problems for voting rules that allow for a group of winners (a committee) to be selected—so called multiwinner voting rules. We present first election control results in the context of multiwinner voting. As this area is relatively unexplored, we take on unrestricted voting at first, showing NP-hardness results for destructive control by deleting voters under Approval Voting and all variants of control by deleting both voters and candidates for Satisfactory Approval Voting rule.

1.3 Related Work

Later in the thesis we discuss existing publications relevant to the content of each chapter. Here, we bring up some interesting research directions that are related to the topics that we study, yet present different ideas from the high level perspective.

At first, when we talk about election control it is important to note other, related, means of manipulating election results. In particular, in *strategic voting* instead of assuming an existence of a malicious agent who has control over the voting process, we consider individual agents casting insincere votes in order to obtain a better outcome. This model has been extended further to consider coalitions of voters deciding on a common, insincere voting strategy that results in a better outcome for all of them. The coalitional manipulation problem has also been studied from the

perspective of computational complexity and NP-hardness results were shown for many of the studied voting rules (e.g., see the survey of Conitzer and Walsh [22]). Another frequently studied mean of manipulating election results is called *bribery*. In the case of bribery, we assume that the briber is an entity external to the election. The way they can impact the outcome this time is by influencing the votes a selected number of agents would cast (e.g., by bribing these agents). Similarly to the aforementioned manipulation methods, bribery has been studied from the computational complexity perspective and proved hard for many voting rules (e.g., see the overview of Faliszewski and Rothe [38]). Additionally, for the bribery case there exist results on complexity under single-peaked profiles (see, e.g., the work of Brandt et al. [15]), showing polynomial time algorithms for bribery under a number of rules for which the problem is hard in the unrestricted case.

Last but not least, we mention two problems that historically were viewed as types of control, but currently are often omitted. These problems are control by partitioning the candidates or the voters. In the voter partitioning scenario, the voting process takes quite a particular shape, with the agent set being divided into two groups, and the election consisting of two rounds. In the first round, we run the election on each of the two disjoint groups of agents separately, which results in two groups of winning alternatives who advance to the second round. In the second round only the selected alternatives compete among each other and the winner of the second round is the overall winner of the election. Given such a setting, we see that the outcome of the election can be manipulated based on how the agents are partitioned. Despite voter partition control being applicable only for very specific cases, it is worth mentioning it here as, similarly to bribery, there exist results showing both hardness of control by partitioning voters and polynomial-time results for the single-peaked case (e.g., results for Llull voting [37]).

In this thesis we present the study of various restricted domains as a way of providing a better understanding of the complexity of election control problems, which often are NP-hard in the unrestricted case. Beside considering restricted domains, there exist several different research directions that, in a sense, aim to answer the same question on how practical the unrestricted hardness results are. One way to approach this is to develop actual algorithms despite the problems being NP-hard. For example, Coleman et al. [20] gave an algorithm for coalitional manipulation under the STV rule, whose runtime is bound by a polynomial of the number of agents and alternatives times the factorial of the number of alternatives. This algorithm can perform well for elections with a relatively small number of

alternatives. In computational social choice there are many examples of heuristic approaches for solving strategic voting problems; see e.g., the work of Procaccia et al. [72] or the work of Faliszewski et al. [36], where heuristic approaches are discussed for weighted bribery under Plurality. In a similar manner, approximation based schemes are broadly discussed, e.g. in the work of Brelsford et al. [18] or the works of Hazon et al. [52, 53]. Another interesting direction is the study on so-called *nearly single-peaked* domain. There are several different models that fits under this category, but from the high level perspective, the definition of nearly single-peaked domain boils down to introducing an additional constant k which is the upper bound for the number of agents whose preferences does not fit a given societal axis, or alternatively the constant is the number of adjustments of agents preference that we can make in order to make it fit our given societal axis. Again, in the case of nearly single-peaked elections a number of control problems, that are hard in an unrestricted case, become polynomial-time solvable [35, 90, 94, 95]. But the study of nearly single-peaked domain yields some other interesting results showing that some control methods that are tractable for single-peaked electorates are NP-hard even with the smallest possible deviation (e.g., for $k = 1$) of the nearly single-peaked variant (see the work of Faliszewski et al. [35]). Finally, the last front of research in the area of complexity of control problems that we wanted to mention, attempts to understand the likelihood of a given control problem instance being polynomial-time solveable by generating random voting profiles. For example, in the work of Erdélyi et al. [33], an experimental approach is presented which shows that a number of control problems can be solved efficiently for Fallback and Bucklin voting despite worst-case complexity being NP-hard.

1.4 Structure of the Thesis

In Chapter 2 we provide the necessary definitions covering the main concepts discussed in the thesis, that is elections, control problems, restricted domains and computational complexity. The further chapters, up to Chapter 6.3, are mostly self-contained and each one starts by giving a short introduction to the topic discussed and outlining our contributions. All of those chapters contain an individual preliminary section, presenting definitions and notations that are specific to the content of each chapter, and ends with a section presenting conclusions and future directions of the matters discussed there.

Chapter 2

Preliminaries

In this chapter we present important notations and definitions used throughout this thesis. In Section 2.1 we formally define elections and related topics such as voting rules and restricted domains. Later, in Section 2.2 we discuss election control problems. Finally, in Section 2.3 we recall fundamental concepts of computational complexity and necessary tools that we use for determining complexity of problems discussed in this thesis.

2.1 Elections

In the ordinal model, an election is a pair $E = (A, N)$ where A is a set of alternatives and N is a set of agents. Each agent i in N has a *preference order* $\succ_i$, i.e., a linear order over A that ranks all the alternatives from the most desirable one to the least desirable one. We refer to $\succ = (\succ_1, ..., \succ_n)$, defined as a set of all the preference orders for all the agents, as a *preference profile* of the election E. In the case where we allow for agents not to have a strong preference between alternatives we use $\succcurlyeq$ to refer to the *preference profile* and for each agent i in N the *preference order* is denoted by $\succcurlyeq_i$ and forms a *weak order* over the set of alternatives. Further, for each agent $i \in N$ with a preference order $\succcurlyeq_i$ and each pair of alternatives $a, b \in A$ if agent i prefers a to b we write $a \succ_i b$, if it prefers b to a we write $b \succ_i a$ and if it does not prefer one of these alternatives over another we write $a \approx_i b$.

For the case of approval voting, elections are defined in the same way, except that agents do not rank the alternatives but simply provide the sets of alternatives that they approve of (they disapprove of the other ones). We refer to such agents as having *dichotomous preferences*. Approval voting profiles can also be modelled

using weak orders with each agent having two equality classes: one for the set of alternatives it approves, and one for the ones it disapproves.

Throughout this thesis we interchangeably refer to agents as voters and to alternatives as candidates. We follow the "voters and candidates" convention in Chapters 4 and 5, where we more often refer to the well established naming for control problems that specifically mentions words "candidates" and "voters" (e.g., Constructive Control by Deleting Voters).

2.1.1 Voting Rules

Below we define the voting rules that we focus on throughout the thesis.

Definition 2.1.1. *The plurality score of an alternative a in election E, denoted $score_E^P(a)$, is the number of agents in N that rank a first. An alternative is a* plurality winner *if it has the highest plurality score.*

An alternative is a *Condorcet winner* if it defeats every other alternative in a pairwise contest. Formally, we have the following definition.

Definition 2.1.2. *For an election $E = (A, N)$ and two alternatives $a, a' \in A$, we write $C_E(a, a')$ to denote the number of agents from N that prefer a to a'. An alternative a is a Condorcet winner if for each alternative $a' \in A \setminus \{a\}$ we have $C_E(a, a') > C_E(a', a)$.*

For a given preference profile different voting rules may yield different election outcomes. Let us take a look at the following example to see how the defined voting rules work.

Example. *Consider a set of alternatives $A = \{a, b, c, d\}$ and five agents with the following preference orders over these alternatives:*

$$a \succ_1 b \succ_1 c \succ_1 d$$
$$a \succ_2 b \succ_2 d \succ_2 c$$
$$b \succ_3 d \succ_3 a \succ_3 c$$
$$c \succ_4 d \succ_4 b \succ_4 a$$
$$d \succ_5 b \succ_5 a \succ_5 c$$

Now, under plurality rule we calculate the score for each alternative by giving one point for each agent that placed such alternative as their top choice. We see that

the score of an alternative a is 2 (it gets one point from agent 1 and 2), whereas the remaining alternatives' score is 1. Therefore, alternative a is the unique winner under the plurality rule. On the other hand, alternative b is the Condorcet winner of the presented election. This comes from the fact that only two agents prefer a to b (namely, agents 1 and 2), two agents prefers d to b (agents 4 and 5), while only one agent prefers c to b (only agent 5), and thus $C_E(b, a) = C_E(b, d) = 3$ and $C_E(b, c) = 4$.

Definition 2.1.3. *The approval rule is defined for elections where the agents have dichotomous preferences. An approval score of alternative a in election E, denoted $score_E^{App}(a)$, is the number of agents in E that approve of a. An* approval winner *is the alternative with the highest approval score.*

To show an example of approval-based elections we turn the previously presented ordinal profile into a dichotomous one by making agents 1 and 2 approve only their top choice while the remaining agents approve their top two choices.

Example. *Again, we take a set of alternatives $A = \{a, b, c, d\}$ and five agents that approves of the following alternatives:*

$$1\colon a \qquad 2\colon a \qquad 3\colon b, d \qquad 4\colon c, d \qquad 5\colon d, b$$

We see that alternative d is the unique winner of the approval election with the given votes. It is because $score_E^{App}(d) = 3$ as d is approved by agents 3, 4 and 5. The score of alternatives a and b is 2, while alternative c is only approved by agent 4 and hence $score_E^{App}(c) = 1$.

For plurality and approval voting one or more winners always exists, whereas in the case of Condorcet rule it is possible that the winner does not exist. However, if the winner exists under Condorcet rule we can be sure that it is also the unique winner.

2.1.2 Restricted domains

When considering elections we will refer to *restricted domains* as to categories of restrictions put on the way agents' preferences over alternatives can be structured. In this thesis we discuss *single-peaked*, *single-crossing* and *top-monotonic* preference profiles.

Definition 2.1.4 (Black [13], Arrow [1]). *A preference profile $\succ$ is single-peaked if there exists a linear order $>$ over the set of the alternatives (the societal axis), such that for each three alternatives x, y, and z, if either it holds that $x > y > z$ or $z > y > x$, then for each agent $i \in N$ we have that $x \succ_i y \implies y \succ_i z$.*

Less formally speaking, in a single-peaked election each agents' preferences always increase, always decrease or first increase and then decrease along the societal axis. As a result, for each agent the alternative that is ranked last is either the first or the last element on the societal axis. Single-peaked profiles arise naturally in settings where agents' preferences are dominated by one factor (e.g., the left-right political spectrum).

Next we define single-crossing domain which is conceptually similar to single-peaked, but instead of relying on the order of alternatives we seek an ordering of the agents.

Definition 2.1.5 (Mirrlees [62] and Roberts [74]). *A profile $\succ = (\succ_1, \ldots, \succ_n)$ is single-crossing with respect to an ordering of agents $(\succ_1, \ldots, \succ_n)$ if for each two alternatives x and y such that $x \succ_1 y$, there is a number $t_{x,y}$ such that $\{i \in N \mid x \succ_i y\} = \{1, \ldots, t_{x,y}\}$. Profile $\succ$ is single-crossing if it is single-crossing with respect to some ordering of the voters.*

The intuition behind single-crossing definition is that when considering agents in the single-crossing order, the relative position of each two alternatives changes ("crosses") at most once. Like in the case of single-peakedness, single-crossing profiles are shown to emerge in some natural settings, e.g. frequently when considering taxation related issues [67].

For both single-peaked [6, 9, 34] and single-crossing domains [16, 27] there exist polynomial-time algorithms that for a given preference profile can recognize if the profile meets the given domain criteria and finds an order of the alternatives (for single-peaked) or agents (for single-crossing) that fulfils the domain definition.

We note that originally single-peakedness and single-crossingness have been defined for the ordinal model. However, in Chapter 4 we introduce and discuss an adoption of the aforementioned notions to the dichotomous profiles.

Let us now take a look at a couple of examples that illustrate how single-peaked and single-crossing profiles look like and demonstrate instances that satisfy one property but not the other and vice verse.

Example. *Consider an election with a set of alternatives* $A = \{a, b, c, d\}$ *and four agents with the following preferences:*

$$b \succ_1 c \succ_1 a \succ_1 d$$
$$b \succ_2 c \succ_2 d \succ_2 a$$
$$c \succ_3 b \succ_3 d \succ_3 a$$
$$c \succ_4 b \succ_4 a \succ_4 d$$

Now, clearly, the presented preference profile is single-peaked with $a > b > c > d$ *being the societal axis. The preferences of agents* 1 *and* 2 *increase monotonically along that axis towards alternative* b *and then decrease monotonically going forward. Agents* 3 *and* 4 *have their "peaks" with alternative* c *and so their preferences increase monotonically up until* c *and then decrease. On the other hand, the profile is not single-crossing. In the presented order of agents we see that, e.g., alternatives* a *and* d *"cross" more than once. The relative order of these agents is flipped two times, we first have that* $a \succ_1 d$*, then* $d \succ_2 a$*, and again for agent* 4 *we have that* $a \succ_4 d$*. It turns out that no order can be found that would satisfy the single-crossing condition. This follows from the fact that in order to fix the aforementioned inconsistency between alternatives* a *and* d*, we would have to move agent* 4 *to be next to agent* 1*. However, if we do that, then we end up with the same problem with respect to alternatives* c *and* b*.*

We now demonstrate an example of a profile that is single-crossing but not single-peaked. For the same number of agents and alternatives let us consider a different set of preferences:

$$a \succ_1 b \succ_1 c \succ_1 d$$
$$b \succ_2 a \succ_2 d \succ_2 c$$
$$b \succ_3 d \succ_3 a \succ_3 c$$
$$b \succ_4 d \succ_4 c \succ_4 a$$

We see that the profile is single crossing according to the presented order of agents. When considering agents in this order, for agent 2 *the alternatives* a *and* b *are flipped as compared to agent* 1*, and the same happens to alternatives* d *and* c*. Then for agent* 3 *the relative order of alternatives* a *and* d *is swapped. Finally, for agent* 4 *the positions of alternatives* c *and* a *are exchanged. As we see, no pair of alternatives*

changes its relative position more than once when considering agents in the listed order. While the profile is single-crossing we note, however, that it does not satisfy the single-peaked condition. The easiest way to see that is by looking at the alternatives that are ranked last—there are three such alternatives d, c and a, which means that a societal axis cannot be built for this profile. We recall that for a single-peaked profile the alternative that is ranked last needs to be at one of the two ends of the societal axis. As there are three such alternatives, in our case we cannot build an axis.

2.2 Control Problems

In essence, election control problems model settings where someone has the power to change the election outcome by modifying this elections' structure. More formally, we are given an election and we ask if it is possible to affect the result of an election in a certain way. In this thesis we discuss four variants of election control: Control by adding candidates (AC), control by deleting candidates (DC), control by adding voters (AV) and control by deleting voters (DV). For each of these control types we are interested both in constructive control (CC), where the goal is to ensure a given candidate's victory, and in destructive control (DC), where the goal is to prevent some candidate from winning.

Definition 2.2.1. *Let R be a voting rule. We are given a set of alternatives A, a collection of agents N, a nonnegative integer k, and a designated alternative $p \in A$. In control by adding voters (AV) in addition we are given a set M of unregistered agents. Similarly, in control by adding candidates (AC) we are given a set B of unregistered candidates. In constructive variants of control problems we ask whether it is possible for p to become a unique winner under voting rule R in the following way:*

1. *In constructive control by adding voters (R-CCAV), we ask whether there exists a set $M' \subseteq M$, such that p is the unique winner of R-election $(A, N \cup M')$, where $\|M'\| \leq k$.*

2. *In constructive control by deleting voters (R-CCDV), we ask whether there exists a set $N' \subseteq N$, such that p is the unique winner of R-election $(A, N \setminus N')$, where $\|N'\| \leq k$.*

3. *In constructive control by adding candidates (R-CCAC), we ask whether there exists a set $B' \subseteq B$, such that p is the unique winner of R-election $(A \cup B, N)$, where $\|B'\| \leq k$.*

4. *In constructive control by deleting candidates (R-CCDC), we ask whether there exists a set $A' \subseteq A$, such that p is the unique winner of R-election $(A \setminus A', N)$, where $\|A'\| \leq k$.*

The destructive variants are defined analogously except that we ask whether it is possible to ensure that p is not the unique winner, and for the deleting candidates variant it is not allowed to delete candidate p.

In addition, for the voting rules where more than one alternative can be a winner (like plurality or approval voting) we can consider the constructive control problem under a unique (as defined above) or a nonunique model. The difference in the nonunique model constructive control problem definition is that we require the designated alternative to be among the winning alternatives. For example, in the case of approval, the designated alternative can have score that is equal to the top ranked alternatives.

Let us take a look at some examples of control problems.

Example. *We take the set of alternatives $A = \{a, b, c, d\}$, an integer $k = 1$ and the set N of four agents with the following preferences:*

$$a \succ_1 b \succ_1 c \succ_1 d$$
$$c \succ_2 a \succ_2 b \succ_2 d$$
$$b \succ_3 a \succ_3 c \succ_3 d$$
$$c \succ_4 a \succ_4 d \succ_4 b$$

We create an instance $I = (A, N, k, a)$ of Plurality-CCDC. In this instances we ask if it is possible to remove at most $k = 1$ alternative from A such that alternative a becames a unique winner under the Plurality rule. We see that in our case it is indeed possible. In the election (A, N) the Plurality score of alternatives a and b is 1, alternative c has score 2, while alternative d has zero points. If we delete alternative c to construct $A' = \{a, b, d\}$, we see that the Plurality score of alternative a in election (A', N) is now 3, while the remaining alternatives' scores do not change. Therefore deleting alternative c solves I.

2.3 Computational Complexity

We assume the reader is familiar with standard notions of computational complexity such as complexity classes P and NP or the big-O notation. In this section we provide some intuitions behind these concepts and definitions of the most essential notions regarding reducibility and completeness that are used throughout this thesis.

Typically, when referring to problems in the context of computational complexity, we mean *decision problems*. A decision problem is formally defined as a set of instances that can be considered as the input, together with a subset of that set which consists of *positive instances* that are accepted by the problem criteria. An example of a decision problem is deciding if a given natural number is prime. In this case, the set of input instances is the set of all natural numbers, and the set of positive instances is the set of all prime numbers. In later chapters we introduce the concept of *function problems* and also *counting problems* that are more sophisticated classes of problems and yield results other than just accepting or rejecting an instance.

In the context of decision problems, a solution of a problem is an algorithm that takes an element from the set of input instances and decides if this element belongs to the set of positive instances. The *time complexity* of an algorithm refers to the running time of the algorithm given the size of an input. An algorithm is said to be *polynomial-time* if there exists a positive number k such that the running time of the algorithm is in $\mathrm{O}\left(n^k\right)$, where n denotes the size of the input. That is, in the worst-case scenario, the number of operations needed to decide whether an instance of size n belongs to the set of positive instances is bounded by $c \cdot n^k$, where c is some non-negative constant. As an example, let us consider the problem of deciding if a sequence of natural numbers is sorted in the ascending order. The set of input instances contains all possible sequences made of natural numbers; and the size of each instance (denoted previously by n) is simply the length of the sequence which corresponds to the instance. A solution would be to check each pair of neighbouring elements from the sequence and test if the elements of these pairs are in ascending order. For a sequence of size n, we need to test $n - 1$ pairs and therefore the total number of operations can be bounded by $c \cdot n^1$ (with some constant c). Hence the algorithm's running time is in $\mathrm{O}\left(n\right)$ and the algorithm is said to be polynomial-time.

When for a given problem there exists a *polynomial-time* solution, we say that the problem is in class P (we also say that the problem is *tractable* or *polynomial-time solvable*). The class NP refers to all decision problems (including the ones in P) that can be verified in polynomial time. One of the unsolved problems of computer

science is whether the P and NP classes are equal or not. Even though there is no proof to support this statement, it is commonly believed that P $\neq$ NP. This leads to the apparent existence of polynomial-time verifiable decision problems that are computationally harder than the problems from P and, supposedly, are not solvable in polynomial time. We refer to such problems as being *intractable*. The main tool that helps us categorize problems with respect to their computational complexity is called *reduction*.

Definition 2.3.1. *Let P and Q be two decision problems. We say that P reduces to Q if there exists a mapping function $f\colon P \to Q$, such that for each instance $I \in P$, $f(P)$ can be computed in polynomial-time and I is a positive instance of P if and only if $f(P)$ is a positive instance of Q.*

We also sometimes use *Turing* reductions. We say that problem P Turing reduces to problem Q if there is a polynomial-time algorithm that solves P, provided it is given the ability to solve Q in constant time (i.e., provided it has oracle access to Q).

Reductions allow us to show that the problem we are reducing to is not easier than the problem we reduce from. In essence, it gives us a way to transform all instances of one problem to the other. Thanks to that, reductions can be used to show that a given problem is polynomial-time solvable by reducing it to another problem known to be in P. Moreover, when we inverse that logic and start reduction from an intractable problem, we can prove another problem's hardness. In other words, if the problem we reduce from is intractable, then we can state that the problem we reduce to is intractable too. In practice, most of our hardness proofs provide reductions from NP-complete problems. A problem is NP-complete if it belongs to NP and all problems from NP reduce to it. If all problems from NP reduce to a given problem, but we do not require the problem to be in NP, then we say that it is NP-hard. Below we present some NP-complete problems that we use for hardness proofs in this thesis.

Definition 2.3.2. *In the* Set-Cover *problem we are given a set $U = \{1, ..., n\}$, a family $S = \{S_1, ..., S_m\}$ of subsets of U, and a positive integer k. A* set-cover *is a subfamily $C \subseteq S$ such that $\bigcup_{X \in C} X = U$. We ask if there exists a set-cover of size at most k.*

The *Set-Cover* problem is NP-complete, but to prove NP-hardness it is often more convenient to reduce its more restricted variant that is also NP-complete, known as X3C.

Definition 2.3.3. *In* Exact-3-Set-Cover *(X3C) problem we are given a positive integer k, a set $U = \{1, ..., 3 \cdot k\}$ and a family $S = \{S_1, ..., S_m\}$ of subsets of U of size three. We ask if there exists a set-cover of size size at most k.*

In other words, *X3C* is a variant of *Set-Cover*, where the size of the set U is divisible by 3, each subset of U in family S has three elements, and the parameter k is set to $\frac{\|U\|}{3}$.

Chapter 3

Recognizing Restricted Domains

In the context of computational social choice, domain restrictions are constraints put on the way how agents' preferences can be structured. For example, in the case of the single-peaked domain restriction we require that there exists an axis along which all the alternatives are ordered (societal axis) and each agent's preference order needs to first monotonically increase along this axis and, after that, monotonically decrease (each agent's preference has a single peak along the axis). There are several reasons why restricted domains play an important role in computational social choice. The first one is that they allow to model more realistic elections by eliminating the ability to choose preference orders completely at random. For example, in the case of single-peaked elections we can imagine the societal axis to represent the alternatives ordered from the most left-wing to the most right-wing ones. This leads to single-peaked profiles that often arise in political settings. The second reason for considering restricted domains is that some undesirable situations cannot occur. A great example is Condorcet voting which, in the general case, may yield no results, but under both single-peaked and single-crossing domain restrictions is guaranteed to produce a winner (in other words, if a preference profile is either single-peaked or single-crossing, then a weak Condorcet winner certainly exists). Other examples include the impossibility theorems of Arrow [1] and of Gibbard and Satterthwaite [45, 77], which do not hold under single-peaked and single-crossing restrictions. Lastly, considering restricted domains often impacts computational tractability of the election control problems. For example, control by adding or deleting alternatives is intractable in the unrestricted case [8] but can be solved in polynomial-time for both single-peaked and single-crossing elections [37].

In this chapter we discuss restricted domain recognition problem. By that we mean the problem of determining whether a given election fulfills the definition of a

selected restricted domain and, if so, determining the domain's specific attributes, such as the societal axis in the case of single-peaked elections. The recognition problem is fundamental when considering any restricted domain from the computational standpoint. Indeed, without knowing how complex it is to detect if an election belongs to a given restricted domain, it is difficult to argue about the complexity of control problems under this domain. For example, when considering a given control problem under the single-peaked domain, we usually assume that this election is single-peaked and that the agents are ordered according to the given societal axis. These assumptions are based on the fact that we are able to tell if a given election is single-peaked (we can *recognize* single-peaked elections) and we can determine its societal axis in polynomial-time [9, 34].

We focus on the well known notions of single-peaked and single-crossing elections, and in addition we also discuss the notion of top-monotonicity of Barberà and Moreno [7]. Top-monotonicity is a generalization of the above two restricted domains (i.e., all profiles that are single-peaked or single-crossing are also top-monotonic), which allows for the agents to have weak preferences and still guarantees the existence of a weak Condorcet winner. The recognition problems for the cases of single-peaked and single-crossing elections are very well studied, with many efficient polynomial-time algorithms available. By contrast, there is only a handful of publications that discuss top-monotonicity in the context of computational social choice. Moreover, to the best of our knowledge, the existing algorithms for recognizing single-peaked or single-crossing profiles cannot be easily adapted to the case of top-monotonicty, despite these domains being closely related.

In this chapter we present a novel framework for solving recognition problems that covers all the three mentioned domains; as a result, we provide the first polynomial-time algorithm for recognizing top-monotonic profiles. The core idea behind our methodology is that by focusing on triples of alternatives (or agents, for the case of single-crossing elections), we can identify how they need to relate to each other in the case that the profile fulfills the restricted domain's assumptions. As it turns out, these relations can be expressed as 2CNF logical formulas and, therefore, the recognition problems are reduced to solving SAT-2CNF instances.

We mention that during the course of our studies a number of other restricted domains were either defined or gained some attention in the computational social choice literature. These include, e.g., variants of single-peakedness and single-crossingness on trees [25, 69, 82, 96], preferences single-peaked on a circle [70], various Euclidean-based restrictions [30, 31, 68], and the classic notion of group-separable

preferences [27, 49–51]. These all are very interesting, but the recognition problems for them are currently well understood.

Acknowledgments

The main results presented in this chapter, that is, the recognition algorithms for top-monotonic profiles and for single-peaked profiles, have been presented in a paper co-authored by myself and Piotr Faliszewski, titled *Recognizing top-monotonic preference profiles in polynomial time* [60]. All technical results in that paper, including the technique for building recognition algorithms, are my contribution. In this thesis we expand on the use of this technique by adding a section on recognizing single-crossing profiles.

3.1 Recognizing Single-Peaked Elections

In this section we show how the problem of recognizing single-peaked elections can be reduced to solving a SAT-2CNF formula. The high level idea behind this algorithm is that we look at triples of alternatives and based on the single-peaked definition we determine how these alternatives can be positioned relative to each other on the societal axis, assuming it exists. This gives us a set of rules for each triple of alternatives which then can be expressed in a 2CNF form. Finally, we consider all these rules together to build and solve a SAT-2CNF system.

As per Definition 2.1.4, a preference profile $\succ$ is single-peaked if there exists a linear order $>$ over the set of alternatives such that for each three alternatives x, y, and z, it holds that:

$$(x > y > z) \vee (z > y > x) \implies \forall_{i \in N}(x \succ_i y \implies y \succ_i z).$$

We follow the notation from this definition. For each triple of alternatives $S = \{x, y, z\}$ we define the set L_S of legal orderings to be the set of ordered sequences $\sigma = (\sigma_1, \sigma_2, \sigma_3)$, such that:

1. $\{\sigma_1, \sigma_2, \sigma_3\} = S$, and
2. for each $i \in N$ we have $\sigma_1 \succ_i \sigma_2 \implies \sigma_2 \succ_i \sigma_3$.

We see that the set of legal orderings for a given triple tells us what are the possible ways in which the alternatives from the triple can be ordered on the societal axis in

order to fulfill the requirements of single-peakedness. Therefore, if for some triple of alternatives S the set of legal orderings L_S is empty, we know that the profile is not single-peaked as this would mean that there is no way these alternatives can be placed on the axis without violating the definition's requirements. From now on we assume that for every triple S, L_S is non-empty.

We note that if for some set $S = \{\sigma_1, \sigma_2, \sigma_3\}$ a triple $(\sigma_1, \sigma_2, \sigma_3)$ belongs to L_S, then the triple $(\sigma_3, \sigma_2, \sigma_1)$ also belongs to L_S. This follows because the condition for including triple $(\sigma_1, \sigma_2, \sigma_3)$ simply says that if we restrict the preference orders of all the agents to alternatives $\sigma_1, \sigma_2, \sigma_3$, then σ_2 is never ranked last among σ_1, σ_2, σ_3 (so, in terms of Fishburn [42], we have a *never last* restriction). The same condition is satisfied by the triple $(\sigma_3, \sigma_2, \sigma_1)$.

Let us consider a set $S = \{x, y, z\}$ of three alternatives and agent $i \in N$. There are six possible permutations on how alternatives from S can be ordered according to the preference order $\succ_i$ of agent i. Let us first assume that $x \succ_i y \succ_i z$. In such a case, we see that neither (y, z, x) nor (x, z, y) is a part of L_S. This follows from the fact that the least favorable alternative cannot be placed in between the more favorable ones on the societal axis, so such a setup clearly violates the definition of single-peakedness. Thus, regardless of how alternatives from S are ordered according to $\succ_i$, we can always find at least one pair of sequences of alternatives from S that are not included in L_S. Therefore what we are left with are either two or four possible sequences in L_S for each given S (we already assumed that L_S is never empty and we argued that if a sequence belongs to L_S then so does its reverse).

Below we consider two possible cardinalities of L_S for a selected triple $S = \{x, y, z\}$:

1. If L_S contains exactly two elements, then—up to renaming the alternatives—it must be of the form $L_S = \{(x, y, z), (z, y, x)\}$.

2. If L_S contains four elements, then—up to renaming the alternatives—it is of the form $L_S = \{(x, y, z), (z, y, x), (y, x, z), (z, x, y)\}$. Indeed, for each sequence in L_S, its reverse must be included as well, and one can verify that—up to renaming of the candidates—this is the only possible form of L_S.

We now want to express the sets of legal orderings in the form of 2CNF formulas. Our variables will correspond to the ordered pairs of alternatives, however for each pair of alternatives we need to make sure that only one variable is being used and that we do not create symmetrical variables based on the reverse order. For example,

if we take alternatives x and y, then we could create two complementary literals xy and yx, where literal xy is a negation of the literal yx. But as we want to avoid having two variables per alternatives' pair, we choose arbitrarily which literal is represented as a variable and which literal is this variable's negation. For the first variant of the set L_S (with two elements), we note that we can express the set constraints as the following 2CNF formula:

$$(xy \vee zx) \wedge (xz \vee zy) \wedge (yz \vee yx)$$

or, rather, as:

$$(xy \vee \neg xz) \wedge (xz \vee \neg yz) \wedge (yz \vee \neg xy)$$

in order to avoid using symmetrical variables. Similarly, the constrains in the second variant (with four elements) can be expressed as follows:

$$(xz \vee \neg yz) \wedge (yz \vee \neg xz).$$

To summarize the process, for each triple of alternatives $S = \{a, b, c\}$ we compute the set of legal orderings L_S and proceed as follows:

1. If L_S has two elements, then we find a mapping between S and the set $\{x, y, z\}$ such that when we map elements from L_S to $\{x, y, z\}$, then we get $\{(x, y, z), (z, y, x)\}$. In this case we generate a 2CNF formula, $(xy \vee \neg xz) \wedge (xz \vee \neg yz) \wedge (yz \vee \neg xy)$ and translate it back into the domain of alternatives from S using inverse mapping.

2. If L_S has four elements, then we find a mapping between S and the set $\{x, y, z\}$, such that after mapping all the elements from L_S, we get a set $\{(x, y, z), (z, y, x), (y, x, z), (z, x, y)\}$. We generate a 2CNF formula, $(xz \vee \neg yz) \wedge (yz \vee \neg xz)$ and, similarly to the above case, translate it using inverse mapping.

Finally, we form *the global ordering formula* by taking a conjunction of all the generated 2CNF formulas. We claim that if the global ordering formula is satisfiable, then the profile is single-peaked, and otherwise it is not.

To prove the above statement, we first assume that the global ordering formula is satisfiable. In this case, for each pair of alternatives x, y we define a relation $>$ such that $x > y$ if literal xy evaluates to $True$ in the global ordering formula solution, and $y > x$ if literal xy evaluates to $False$. We see that $>$ is a strict order over the set of alternatives because:

(a) It is defined for each pair of distinct alternatives.

(b) It is irreflexive as we do not define the relation between two identical alternatives.

(c) It is asymmetric, as otherwise for some two candidates x and y we would have literal xy which would evaluate to both $True$ and $False$.

(d) It is transitive, because for each triple of alternatives $S = \{x, y, z\}$ transitivity is enforced by how the alternatives are ordered into the three-element sequences from L_S.

We note that $>$ (assuming it exists) is a single-peaked axis for the preference profile. We see that for each set of three alternatives $S = \{x, y, z\}$, if $x > y > z$ or $z > y > x$ then both (x, y, z) and (z, y, x) are included in L_S. This follows from the fact that the rule generated based on the set L_S enforces the relation such that x, y, and z are always placed in one of the orders from L_S. This in turn means that for each agent $i \in N$ we have $x \succ_i y \implies y \succ_i z$, as we require this for both (x, y, z) and (z, y, x) to be included in L_S.

To show the other direction, it is sufficient to note that if a single-peaked order exists then the global ordering formula is satisfiable. We assume $>$ exists and is the single-peaked order for our profile. We now take the global ordering formula and we assign $True$ to each literal xy, where x and y are two distinct alternatives such that $x > y$. For the sake of a contradiction, let us assume that the above assignment of literals does not satisfy the global ordering formula. It means that there is a clause in the formula that has both literals evaluating to $False$. We know that every clause in the formula corresponds to a set of legal orderings for some triple of alternatives. We let $S = \{x, y, z\}$ be a triple of alternatives and L_S be a set of legal orderings the failing clause corresponds to. As L_S contains all orderings of alternatives x, y and z that are allowed to appear on the societal axis, we see that the fact that the clause is not satisfied means that $>$ does not order x, y and z in one of these possible ways. This contradicts the fact that $>$ is a single-peaked axis and completes out proof.

Before we move on to showing how the method described above can be applied to other restricted domains, we demonstrate its use on an example single-peaked profile.

Example. *Consider candidate set* $\{a, b, c, d\}$ *and profile* $\succ$ *of three preference orders:*

$$b \succ_1 c \succ_1 d \succ_1 a, \qquad c \succ_2 b \succ_2 d \succ_2 a, \qquad a \succ_3 b \succ_3 c \succ_3 d.$$

To run our algorithm, we start by defining the sets of legal orderings for each triple of alternatives:

1. *For the set* $S_1 = \{a, b, c\}$*, we have* $L_{S_1} = \{(a, b, c), (c, b, a)\}$*. Clearly,* (b, a, c) *and its reverse cannot be a part of* L_{S_1} *because* $b \succ_2 a$ *holds while* $a \succ_2 c$ *does not hold, which makes it not fulfill the definition as we would expect that* $b \succ_2 a \implies a \succ_2 c$ *is true. Similarly, sequence* (a, c, b) *and its reverse are not included as we have that* $a \succ_3 c$ *but also* $c \not\succ_3 b$*.*

2. *For the set* $S_2 = \{a, b, d\}$*, we have* $L_{S_2} = \{(a, b, d), (d, b, a)\}$*.*

3. *For the set* $S_3 = \{a, c, d\}$*, we have* $L_{S_3} = \{(a, c, d), (d, c, a)\}$*.*

4. *Finally, for the set* $S_4 = \{b, c, d\}$*, we have* $L_{S_4} = \{(b, c, d), (d, c, b), (c, b, d), (d, b, c)\}$*. Here we only eliminated sequence* (c, d, b) *and its reverse, which do not satisfy the single-peakedness condition for any* $i \in \{1, 2, 3\}$*.*

As we see, there are no empty sets of legal orderings so we can move on to the next step.

For each of S_i*,* $1 \leq i \leq 4$*, we output the 2CNF formula that corresponds to* L_{S_i} *and we take the conjunction of these formulas as the global ordering formula. We present this formula below (the following lines represent the clauses generated from* L_{S_1}*,* L_{S_2}*,* L_{S_3}*, and* L_{S_4}*, respectively):*

$$\begin{aligned}(ab \vee ca) \wedge (ac \vee cb) \wedge (bc \vee ba)\\ \wedge(ab \vee da) \wedge (ad \vee db) \wedge (bd \vee ba)\\ \wedge(ac \vee da) \wedge (ad \vee dc) \wedge (cd \vee ca)\\ \wedge(cd \vee db) \wedge (bd \vee dc).\end{aligned}$$

For each pair of alternatives x *and* y *from* $\{a, b, c, d\}$*, we choose one of* xy *and* yx *to be a variable and the other to be its negation, to obtain the following global ordering*

formula:

$$(\underline{ab} \vee \neg ac) \wedge (\underline{ac} \vee \neg bc) \wedge (\underline{bc} \vee \neg ab)$$
$$\wedge(\underline{ab} \vee \neg ad) \wedge (\underline{ad} \vee \neg bd) \wedge (\underline{bd} \vee \neg ab)$$
$$\wedge(\underline{ac} \vee \neg ad) \wedge (\underline{ad} \vee \neg cd) \wedge (\underline{cd} \vee \neg ac)$$
$$\wedge(\underline{cd} \vee \neg bd) \wedge (\underline{bd} \vee \neg cd).$$

We seek a satisfying truth assignment for this formula. This can be done using one of the polynomial-time solvers for SAT-2CNF problem, but in our case it suffices to notice that every clause contains a non-negated variable so we simply set all the variables to be true. In the formula above we underlined the literals that are set to be true.

The fact that the global ordering formula has a solution indicates that the profile is single-peaked. From the solution of the formula we also get single-peaked order $R_> = \{(a,b),(a,c),(a,d),(b,c),(b,d),(c,d)\}$ *that yields societal axis* $a > b > c > d$.

3.2 Recognizing Single-Crossing Elections

The methodology presented in the previous section maps well to the case of recognizing single-crossing elections. However, in this case instead of focusing on triples of alternatives we will look at triples of agents. This has to do with the fact that unlike in the case of single-peaked profiles, the notion of single-corssingness applies to the order of the agents. Recall that the single-crossing order arranges agents in a way that the relative position of each two alternatives changes at most once along that order (see Definition 2.1.5).

We let A be the set of alternatives and N be the set of agents. We start by considering a triple of agents $S = \{a, b, c\}$ and define a set L_S of legal orderings that will consist of ordered sequences of elements from S, which corresponds to the way how agents from S can be ordered in the single-crossing order. In order to build L_S for a given set S, we start from L'_S that consists of all the possible ordered sequences of elements from S. There are exactly six such sequences: (a, b, c), (a, c, b), (b, a, c), (b, c, a), (c, a, b), and (c, b, a). Now, for each pair of alternatives $x, y \in A$ we look at the preference orders of the agents from S over x and y. Depending on the preferences of the selected agents, they can eliminate some of the possible orderings from L'_S, and in order to reflect that we update L'_S such that we keep only those

orderings that are allowed. As each of the agents from S can either prefer x over y or the other way around, there are only two possible scenarios that require analysis and all the other possibilities are symmetrical. The first case is that two of the agents prefer x over y and the third agent prefers y over x. Without loss of generality let us assume that $x \succ_a y$, $x \succ_b y$ and $y \succ_c x$. If this is the case, we see that if the election is single-crossing then in the single-crossing order agent c can never be placed in between agents a and b. This directly violates the definition of single-crossing profiles and intuitively we can see that if we consider ordering (a, c, b), then the alternatives x and y would exchange their relative position twice. As a result, in this scenario we can remove orderings (a, c, b) and (b, c, a) from the set L'_S. The second scenario is that all three agents prefer x to y. In this case, however, there are no constraints set on the order of agents from S in the single-crossing order, assuming it exists.

Once we analyze all pairs of alternatives for each triple of agents S, we get a set L_S that only contains valid orderings of the agents in the single-crossing order. Since for each analyzed pair of alternatives we either remove none or two orderings from L'_S, the resulting set may be of the following shapes:

1. $L_S = \emptyset$ – in this case it means that there are no legal ordering for the agents from S. It therefore means that the election is not single-crossing, as regardless of how agents from S are ordered, they always violate the definition of single-crossingness.

2. $L_S = \{(a, b, c), (c, b, a)\}$ – in this case the set consist of two elements. Note that if this happens then the two possible orderings are reverses of each other. This comes from the fact that when eliminating invalid orderings we always delete two that have the same alternative in the center.

3. $L_S = \{(a, b, c), (c, b, a), (b, a, c), (c, a, b)\}$ – in this case the set has four elements. Moreover it cosists of two pairs of sequences where each pair has the same alternative in the middle. The reasoning behind this is similar as in the previous case.

4. Lastly, the set L_S may contain all the initial six elements.

By following the methodology from the case of single-peaked elections, we now express the sets of legal orderings in a form of a 2CNF formula. However, this time our variables correspond to the ordered pairs of agents. By analogy, we aim to not

use symmetrical variables but make only one variable for each pair of agents and use negation to express the reverse order. We see that for the first case, when the set L_S is empty, there is no need to do that as in such a case we can immediately tell that the election is not single-crossing. For the second and third case, the described sets of legal orderings can be expressed by the following formulas, respectively:

$$(ab \vee \neg ac) \wedge (ac \vee \neg bc) \wedge (bc \vee \neg ab)$$

and

$$(ac \vee \neg bc) \wedge (bc \vee \neg ac)$$

The last case does not put any restrictions on the order of the agents (except that, indeed, it is an order) and therefore no formula can be provided. This case turns out to be problematic as if for some triple we do not contribute any formula into the global ordering formula, then the order generated based on the solution of the global ordering formula is not going to be a complete order. It is because in such a case there may exist pairs of alternatives which do not have associated variables in the global ordering formula and, hence, we will not be able to tell how to position them in the single-crossing order. To workaround this problem we note that if L_S contains all six possible orderings for a given triple of agents S, it means that all three agents have exactly the same preference orders over all the pairs of alternatives—in other words, such three agents have identical preferences. In such a case, we can safely remove two out of these three agents from our election and, once we find the single-crossing order of the agents, we can place them directly next to the one agent (with identical preference order) that was left.

We now form the global ordering formula out of the individual formulas made for each triple of agents (unless for a given triple no formula has been provided). We argue that if the global ordering formula is satisfiable, then the profile is single-crossing, and otherwise it is not.

Let us first assume the global ordering formula is satisfiable. If so, we define a relation $>$ for each pair of agents $a, b \in N$ such that $a > b$ if literal ab evaluates to $True$ and $b > a$ otherwise. Following the same reasoning as in the single-peakedness case, we see that the relation $>$ is a strict order over the set of agents. We also note that $>$ is the single-crossing order. This follows from the fact that for each triple of agents S they are now ordered in the way that is allowed by the set L_S which, in turn, for each pair of alternatives allows for their relative order to be only changed once when following this order. Finally, for showing the other direction we follow

the same methodology as in the single-peaked version.

Again, we illustrate how the method work in practice on an example single-crossing profile.

Example. *Consider alternatives set* $\{a, b, c, d\}$ *and profile* $\succ$ *of four agents:*

$$a \succ_1 b \succ_1 c \succ_1 d$$
$$c \succ_2 a \succ_2 b \succ_2 d$$
$$c \succ_3 b \succ_3 d \succ_3 a$$
$$d \succ_4 d \succ_4 b \succ_4 a$$

Our algorithm for recognizing single-crossing profiles starts by calculating the set of legal orderings. Let us consider a triple of agents $S = \{1, 2, 3\}$*, we start by defining a set that consists of all the possible orderings of these agents* $L'_S = \{(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)\}$*. Now let us consider a pair of alternatives* a *and* b*. We see that agents* 1 *and* 2 *prefers* a *over* b *and that agent* 3 *prefers* b *over* a*. It means that agent* 3 *can never be placed in between agents* 1 *and* 2 *in the single-crossing order and therefore we can eliminate orderings* $(1, 3, 2)$ *and* $(2, 3, 1)$ *from the set* L'_S*. If we now look at alternatives* a *and* c *we note that agent* 1 *prefers* a *over* c *while agents* 2 *and* 3 *prefers* c *over* a*. This further eliminates orderings* $(2, 1, 3)$ *and* $(3, 1, 2)$*. Considering the remaining pairs of alternatives does not result in further elimination of orderings, we are left with the following set of legal orderings* $L_{\{1,2,3\}} = \{(1, 2, 3), (3, 2, 1)\}$*. Similarly, we calculate sets for the remaining triples of agents, the results are as follows:* $L_{\{1,2,4\}} = \{(1, 2, 4), (4, 3, 2)\}$*,* $L_{\{1,3,4\}} = \{(1, 3, 4), (4, 3, 1)\}$ *and* $L_{\{2,3,4\}} = \{(2, 3, 4), (4, 3, 2)\}$*. We now form a 2CNF formulas based on the sets provided above. We use notation* $v_{1<2}$ *as a variable to denote that it represents that agent* 1 *is before* 2 *in the single-crossing order:*

$$(\underline{v_{1<2}} \vee \neg v_{1<3}) \wedge (\underline{v_{1<3}} \vee \neg v_{2<3}) \wedge (\underline{v_{2<3}} \vee \neg v_{1<2})$$
$$\wedge(\underline{v_{1<2}} \vee \neg v_{1<4}) \wedge (\underline{v_{1<4}} \vee \neg v_{2<4}) \wedge (\underline{v_{2<4}} \vee \neg v_{1<2})$$
$$\wedge(\underline{v_{1<3}} \vee \neg v_{1<4}) \wedge (\underline{v_{1<4}} \vee \neg v_{3<4}) \wedge (\underline{v_{3<4}} \vee \neg v_{1<3})$$
$$\wedge(\underline{v_{2<3}} \vee \neg v_{2<4}) \wedge (\underline{v_{2<4}} \vee \neg v_{3<4}) \wedge (\underline{v_{3<4}} \vee \neg v_{2<3}).$$

The final step it to solve the above SAT-2CNF system. We highlighted variables that can be assigned $True$ *in order for the whole system to evaluate to* $True$*. The*

highlighted variables form a lexicographical order $1, 2, 3, 4$ *which is the single-crossing order of agents for the considered election.*

3.3 Recognizing Top-Monotonic Elections

At last we move on to the case of recognizing top-monotonic profiles. Based on the presented methodology we show how the problem can be solved in polynomial-time. As the concept of top-monotonicity is far more sophisticated on its own than single-crossingness or single-peakedness, the proof is much more elaborate and we decided to split it into a few parts. In the first subsection we provide all the necessary definitions that will be used throughout this section. In the next subsection we start our proof by explaining how sets of legal orderings can be constructed — in that step we show how the definition of top-monotonicity can be recast in order for us to only consider triples of alternatives. Later we provide a proof for the case of narcissistic profiles (where inequalities are not allowed) — because of approaching the simplified version of the original problem we are able to explain the necessary methodology in a more approachable way. In the final subsection we present the proof for the main problem.

3.3.1 Preliminaries

Below we provide a formal definition of top-monotonic preferences, and for that we mostly adopt the notation of Barberà and Moreno [7], who introduced this notion. We let A be a finite set of alternatives, N be a finite set of agents, and $\succcurlyeq$ be a preference profile for the agents in N. For all $i \in \mathbb{N}$ and for each $S \subseteq A$, we denote by $t_i(S)$ the set of top choices of the i-th agent among the alternatives from S. That is, $t_i(S) = \{x \in S \mid x \succcurlyeq_i y \text{ for all } y \in S\}$ and we call it the top of i in S according to $\succcurlyeq$. Let $T = \bigcup_{i \in N} t_i(A)$. For each preference profile $\succcurlyeq$, let $A(\succcurlyeq)$ be the family of sets containing A itself and all triples of distinct alternatives where each alternative is top in A for some agent $i \in N$ according to $\succcurlyeq$ (i.e., the triples in $A(\succcurlyeq)$ consist of the candidates from T, but $A(\succcurlyeq)$ also contains A).

Definition 3.3.1 (Barberà and Moreno [7])**.** *A preference profile* $\succcurlyeq$ *is top-monotonic if there exists a linear order* $>$ *over the set of the alternatives, such that:*

(1) $t_i(A)$ *is a finite union of closed intervals for all* $i \in N$.[1]

(2) For all $S \in A(\succcurlyeq)$*, for all* $i, j \in N$*, all* $x \in t_i(S)$*, all* $y \in t_j(S)$*, and all* $z \in S$*, we have that:*

$$[x > y > z \vee z > y > x] \implies \begin{cases} y \succcurlyeq_i z & \text{if } z \in t_i(S) \cup t_j(S) \\ y \succ_i z & \text{if } z \notin t_i(S) \cup t_j(S) \end{cases}$$

Definition 3.3.2. *A linear order* $>$ *over the set of alternatives is a* top-monotonic order *of a preference profile* $\succcurlyeq$ *if* $\succcurlyeq$ *is top-monotonic and* $>$ *fulfills condition (2) from Definition 3.3.1.*

Barberà and Moreno [7] have shown that each single-peaked and each single-crossing profile is top-monotonic. On the other hand, the following example—taken from their work—shows that there is a profile of strict linear orders that is top-monotonic but that is neither single-peaked nor single-crossing.

Example (Barberà and Moreno [7])**.** *Consider candidate set* $\{a, b, c, d\}$ *and profile* $\succ$ *of three preference orders:*

$$a \succ_1 b \succ_1 c \succ_1 d, \qquad c \succ_2 d \succ_2 b \succ_2 a, \qquad d \succ_3 c \succ_3 a \succ_3 b.$$

Note that c *is preferred to each other candidate by a majority of the agents. The profile is not single-peaked because there are three alternatives that are ranked last (*a*,* b*, and* d*). It is not single-crossing because agents* 1 *and* 3 *would have to be next to each other (because of the alternatives* a *and* b*), while also agents* 2 *and* 3 *would have to be next to each other (because of alternatives* b *and* c*), while finally agents* 1 *and* 2 *would also have to be next to each other (because of alternatives* c *and* d*), which is impossible.*

However, the profile is top-monotonic with respect to the order $a > b > c > d$*. To see that the profile is top-monotonic, note that* $A(\succ) = \{\{a, b, c, d\}, \{a, c, d\}\}$*. We have to check each* $S \in A(\succ)$ *and each two agents* i *and* j*. Let us take* $S = \{a, c, d\}$*,* $i = 1$ *and,* $j = 2$*. We have* $x \in t_1(S) = \{a\}$*,* $y \in t_2(S) = \{c\}$*, and we take* $z = d$*. It holds that* $a > c > d$ *and* $z = d \notin t_1(S) \cup t_2(S) = \{a, c\}$*, so it is required that* $c \succ_1 d$*, and this indeed holds. The same can be checked for all the remaining combinations of* S*,* i*,* j*, and* z *in order to verify that the profile is indeed top-monotonic.*

[1] Since we assumed A to be a finite set, this part of the definition is always trivially satisfied. Barberà and Moreno consider also more general sets of alternatives and—to indicate the generality of their definition—we decided to keep this requirement in the text.

We often need to refer to various definitions, lemmas, and conditions for particular instantiations of the object that they refer to. In such cases, we write $x \to y$ to mean that y takes the role of x. For example, if we wanted to speak of the definition of top-monotonic profiles for some specific agents k and ℓ taking the roles of agents i and j, we would indicate this by writing $i \to k$ and $j \to \ell$.

Before we start, let us provide a formal definition for the problem we are solving in this section: Given a finite set of alternatives A, a finite set of agents N, and a preference profile $\succcurlyeq$ (for the agents in N) over A, find a top-monotonic order of $\succcurlyeq$ or decide that no such order exists. In this section we provide an algorithm for this problem.

3.3.2 Interface to Top-Monotonicity

The following definition, and Lemma 3.3.4 a bit later, constitute an interface between the notion of top-monotonicity and our main algorithm, which reconstructs the top-monotonic order from allowed orders over triples of alternatives.

Definition 3.3.3. *For each triple of distinct alternatives $S = \{x, y, z\} \subseteq A$ and each pair of agents $i, j \in N$, we define* the set of legal orderings*, denoted as $L_S^{i,j}$, to be the set of ordered sequences of alternatives x, y, z, such that for each sequence $\sigma = (\sigma_1, \sigma_2, \sigma_3)$, where $\{\sigma_1, \sigma_2, \sigma_3\} = S$, all the following criteria are met:*

(1)

$$[\sigma_1 \in t_i(S) \text{ and } \sigma_2 \in t_j(S)] \implies \begin{cases} \sigma_2 \succcurlyeq_i \sigma_3 & \text{if } \sigma_3 \in t_i(S) \cup t_j(S) \\ \sigma_2 \succ_i \sigma_3 & \text{if } \sigma_3 \notin t_i(S) \cup t_j(S) \end{cases}$$

(2)

$$[\sigma_1 \in t_j(S) \text{ and } \sigma_2 \in t_i(S)] \implies \begin{cases} \sigma_2 \succcurlyeq_j \sigma_3 & \text{if } \sigma_3 \in t_i(S) \cup t_j(S) \\ \sigma_2 \succ_j \sigma_3 & \text{if } \sigma_3 \notin t_i(S) \cup t_j(S) \end{cases}$$

(3)

$$[\sigma_3 \in t_i(S) \text{ and } \sigma_2 \in t_j(S)] \implies \begin{cases} \sigma_2 \succcurlyeq_i \sigma_1 & \text{if } \sigma_1 \in t_i(S) \cup t_j(S) \\ \sigma_2 \succ_i \sigma_1 & \text{if } \sigma_1 \notin t_i(S) \cup t_j(S) \end{cases}$$

(4)

$$[\sigma_3 \in t_j(S) \text{ and } \sigma_2 \in t_i(S)] \implies \begin{cases} \sigma_2 \succcurlyeq_j \sigma_1 & \text{if } \sigma_1 \in t_i(S) \cup t_j(S) \\ \sigma_2 \succ_j \sigma_1 & \text{if } \sigma_1 \notin t_i(S) \cup t_j(S) \end{cases}$$

(5)

$$\left.\begin{array}{c} [\sigma_1 \in t_i(S) \text{ and } \sigma_3 \succ_i \sigma_2] \\ \vee \\ [\sigma_1 \in t_j(S) \text{ and } \sigma_3 \succ_j \sigma_2] \\ \vee \\ [\sigma_3 \in t_i(S) \text{ and } \sigma_1 \succ_i \sigma_2] \\ \vee \\ [\sigma_3 \in t_j(S) \text{ and } \sigma_1 \succ_j \sigma_2] \end{array}\right\} \implies \sigma_2 \notin T$$

To illustrate how the set of legal orderings is constructed, let us consider the following example. We take $S = \{x, y, z\}$ and two agents i and j with preference orders

$$x \succ_i y \succ_i z \text{ and } z \succ_j x \succ_j y.$$

Now we need to consider six different orderings of candidates x, y, z. Let us consider ordering $\sigma = (x, z, y)$. We can see that it does not satisfy condition (1) as $x \in t_i(S)$, $z \in t_j(S)$ but $z \succ_i y$ is not true. Similarly, if we consider ordering $\sigma' = (x, y, z)$, and if we assume that $x, y, z \in T$, then we can see that it does not satisfy condition (5) as $z \in t_j(S)$ and $x \succ_j y$ (so the left-hand side of the implication is true) but $y \in T$. On the other hand, if we consider ordering $\sigma'' = (z, x, y)$ then we can see that it satisfies all five conditions and therefore is going to be a part of the set of legal orderings $L_S^{i,j}$. By analyzing the remaining three possible orderings we get that the complete set of legal orderings for the setup under consideration is $L_S^{i,j} = \{(z, x, y), (y, x, z)\}$.

Let Q^T be a family of *sets of legal orderings* for every triple x, y, z such that $x, y, z \in T$ and for every $i, j \in N$. Let Q^{NT} be another family of *sets of legal orderings*, for every triple x, y, z, where $x, y \in T$ and $z \in (A \setminus T)$, and every $i, j \in N$ such that $x \in t_i(A)$ and $y \in t_j(A)$. Note that for both definitions we allow for agents i and j to be the same. In other words, family Q^T regards sets of legal orderings for all agents and all triples of candidates that appear on top of some preference orders, whereas Q^{NT} is defined analogously, but for triples of candidates that contain two candidates from tops of some preference orders and one candidate that never appears on top (of any preference order).

Lemma 3.3.4. *Let A be a set of alternatives, N be a set of agents, and $\succcurlyeq$ be a preference profile over A. Let $Q = Q^T \cup Q^{NT}$. Preference profile $\succcurlyeq$ is top-monotonic with $>$ as the top-monotonic order if and only if for each set $X \in Q$, there exists an element $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in X$ such that $\sigma_1 > \sigma_2 > \sigma_3$.*

Proof. We first focus on proving that if top-monotonic order $>$ exists for a given preference profile $\succcurlyeq$, then for each set X from Q there exists an element $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ such that $\sigma_1 > \sigma_2 > \sigma_3$. Let us assume that preference profile $\succcurlyeq$ has a top-monotonic order $>$, but for some set $X' \in Q$ there is no element $\sigma' = (\sigma'_1, \sigma'_2, \sigma'_3) \in X'$ such that $\sigma'_1 > \sigma'_2 > \sigma'_3$. Set X' may be a part of Q^T or Q^{NT}. Let us first assume that $X' \in Q^T$ and, so, each element $\sigma' \in X'$ is a triple of elements $x, y, z \in T$. We also assume that X' corresponds to a pair of agents $k, \ell \in N$. Without loss of generality we assume that $x > y > z$. Now, as per our assumption, sequence (x, y, z) is not a part of X', which means that one of the conditions from Definition 3.3.3 is not met for it. With set $S = \{x, y, z\}$ we consider each of the conditions from Definition 3.3.3:

(a) Since all x, y, z belong to T, condition (5) is not satisfied when the left-hand side of the implication is true. When $x \in t_k(S)$ and $z \succ_k y$, we get a contradiction with the assumption that $\succcurlyeq$ is top-monotonic, because then condition (2) from Definition 3.3.1 is violated for $i \to k$ and for some j such that $y \in t_j(S)$ (we know that such an agent j exists as $y \in T$). Since all the clauses from the left-hand side of condition (5) are symmetric up to the exchange of i with j and σ_1 with σ_3, we see that condition (5) is always satisfied for the sequence (x, y, z), assuming $\succcurlyeq$ is top-monotonic.

(b) If condition (1) is not met, then for $S = \{x, y, z\}$ we have:

$$x \in t_k(S) \text{ and } z \succ_k y$$

$$\textit{or}$$

$$x \in t_k(S) \text{ and } y \in t_\ell(S) \text{ and } z \notin t_k(S) \cup t_\ell(S) \text{ and } z \succcurlyeq_k y$$

However if that were true, $>$ could not be a top-monotonic order of the profile $\succcurlyeq$. The reason is that it directly violates condition (2) from Definition 3.3.1 (for $i \leftarrow k$ and $j \leftarrow \ell$). We therefore get a contradiction and condition (1) has to be satisfied for the selected sequence (x, y, z).

(c) Condition (2) is symmetric to the condition (1) with i and j swapped, and therefore it can be shown in a similar way that it has to be satisfied for a selected sequence (x, y, z). The same applies to condition (3) that is symmetric to condition (1) with σ_1 and σ_3 swapped. Lastly, we can apply the same methodology to condition (4), which can be constructed by swapping both i with j and σ_1 with σ_3 from condition (1).

In consequence, we see that if $X' \in Q^T$, then (x, y, z) has to be a part of X', which stands against our assumption that (x, y, z) is not a part of X'. We therefore get that $X' \in Q^{NT}$. Let us assume that X' corresponds to agents $k, \ell \in N$ and a triple of alternatives x, y, w such that $x \in t_k(A)$, $y \in t_\ell(A)$ and $w \in A \setminus T$. Now we need to consider six possible cases for how agents x, y, w are ordered under $>$:

$$x > y > w, \quad y > x > w, \quad w > x > y, \quad w > y > x, \quad x > w > y, \quad y > w > x.$$

These six options map to the following sequences:

$$(x, y, w), \qquad (y, x, w), \qquad (w, x, y), \qquad (w, y, x), \qquad (x, w, y), \qquad (y, w, x).$$

We set $S = \{x, y, w\}$. We note that $w \notin t_k(S)$ which stands true because x is top of agent k and we know that w is not top of any agent. Similarly, we note that $w \notin t_\ell(S)$. We make the following observations:

(I) If $x > w > y$ or $y > w > x$, we note that sequence $(x, w, y) \in X'$ or $(y, w, x) \in X'$ respectively. This is so because since $w \notin T$, condition (5) is satisfied. Also, all the remaining conditions from (1) to (4) are satisfied because $w \notin t_k(S)$ and $w \notin t_\ell(S)$.

(II) When $x > y > w$, we see that for the corresponding sequence (x, y, w) both conditions (3) and (4) are satisfied, because $w \notin t_k(S)$ and $w \notin t_\ell(S)$. Clearly, condition (2) is also satisfied because $y \succ_k w$ (which comes from the fact that $y \in t_k(A)$ and $w \notin t_k(A)$). Additionally, condition (1) is not satisfied if and only if $w \succcurlyeq_k y$ and condition (5) cannot be satisfied if and only if $w \succ_k y$. However, since $>$ is a top-monotonic order of the profile $\succcurlyeq$, from condition (2) in Definition 3.3.1 given $i \to k$, $j \to \ell$ and $S \to A$ (clearly $x \in t_i(S)$ and $y \in t_j(S)$ and $w \in S$), we get that if $x > y > w$, then it has to be that $y \succ_i w$ (because w is not a top for any agent). Therefore, we get a contradiction with both the assumption that $w \succ_k y$ (corresponding to condition (1)) and the assumption that $w \succcurlyeq_k y$ (corresponding to condition (5)). It means that conditions (1) and (5) are both satisfied, and therefore if $x > y > w$ then $(x, y, w) \in X'$.

(III) When $y > x > w$, we again use the fact that the rules from Definition 3.3.3 are symmetric up to the exchange of i and j, which—based on the above—proves that if $y > x > w$ then $(y, x, w) \in X'$. We follow the same methodology for

the remaining orderings, $w > x > y$ and $w > y > x$, that are symmetric up to the exchange of σ_1 and σ_3.

We therefore see that it is impossible to find a set X' that would violate the assumption from Lemma 3.3.4 when $\succcurlyeq$ is top-monotonic.

The final step of this proof is to show that Lemma 3.3.4 is also true in the opposite direction. That is, when there exists a linear order $>'$ such that for each $X \in Q$ there exists an element $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in X$, such that $\sigma_1 >' \sigma_2 >' \sigma_3$ then $\succcurlyeq$ is top-monotonic and that $>'$ is a top-monotonic order over $\succcurlyeq$. We assume that $>'$ exists and that $\succcurlyeq$ is not top-monotonic, and we will reach a contradiction, showing that $>'$ satisfies all the requirements to be a top-monotonic order over $\succcurlyeq$. If, as per our assumption, $\succcurlyeq$ is not top-monotonic then there exists a set $S \in A(\succcurlyeq)$, a pair of agents $i, j \in N$, and three alternatives x, y, z, $x \in t_i(S)$, $y \in t_j(S)$ and $z \in S$, such that:

$$(x >' y >' z \vee z >' y >' x) \text{ and } (z \succ_i y \vee (z \approx_i y \text{ and } z \notin t_i(S) \cup t_j(S))). \quad (3.1)$$

Let $S' = \{x, y, z\}$. We note that $z \in t_i(S) \iff z \in t_i(S')$. It stands true because we know that $x \in t_i(A)$ which is also true for any subset of A containing x, therefore $x \in t_i(S)$ and $x \in t_i(S')$. Now if either $z \in t_i(S)$ or $z \in t_i(S')$ then we have $x \approx_i z$ which means that $z \in t_i(A)$. We can similarly show that $z \in t_j(S) \iff z \in t_j(S')$. If Eq. (3.1) is satisfied then there exists a set $X'' \in Q$ that contains either sequence (x, y, z) or (z, y, x). It is easy to note that when $z \in T$ then $X'' \in Q^T$ and when $z \notin T$ then $X'' \in Q^{NT}$. We now consider two cases depending on whether (x, y, z) or (z, y, x) is in X''.

(I) Let us first assume $(x, y, z) \in X''$. Now, from condition (1) of Definition 3.3.3, we see that if $z \notin t_i(S') \cup t_j(S')$, then $y \succ_i z$, which makes it impossible to satisfy Eq. (3.1) (the second segment of the equation requires that $z \succcurlyeq_i y$). On the other hand, when $z \in t_i(S') \cup t_j(S')$, then from condition (1) we get that $y \succcurlyeq_i z$, which again makes it impossible for Eq. (3.1) to be satisfied (here we refer to the fact that if $z \in t_i(S') \cup t_j(S')$ then $z \in t_i(S) \cup t_j(S)$).

(II) When $(z, y, x) \in X''$, then we follow the same methodology, leveraging the symmetry of Definition 3.3.3 with respect to the exchange of σ_1 with σ_2 (instead of condition (1) we use condition (3)).

We see that it is impossible for Eq. (3.1) to be satisfied for each of the aforementioned combinations of agents and alternatives, assuming that $>'$ exists. Therefore $>'$ is a top-monotonic order for $\succcurlyeq$.

□

The greatest advantage of using Lemma 3.3.4 and Definition 3.3.3 over directly applying Definition 3.3.1 is that it allows us to focus on sets of three candidates only (Definition 3.3.1 also uses the whole set A, as $A \in A(\succcurlyeq)$). This property is crucial for our technique.

3.3.3 The Case of Narcissistic Profiles

Before we get to our main theorem, we present a proof for a simpler variant. The full proof, presented in the next section, uses a very similar approach as the variant presented below.

Bartholdi and Trick [9] refer to profiles where each candidate is ranked first by at least one agent as *narcissistic*, and we extend their definition to the case of weak orders in a natural way: A profile is narcissistic if every alternative is top of some agent. Below we show an algorithm for recognizing narcissistic top-monotonic profiles.

Theorem 3.3.5. *Let A be a set of alternatives, N be a set of agents, and $\succcurlyeq$ be a narcissistic preference profile. The problem of determining if a top-monotonic order of $\succcurlyeq$ exists (and computing it) is polynomial-time solvable.*

Proof. Due to Lemma 3.3.4, it suffices to demonstrate that finding an order $>$ over A, such that for each $X \in Q$ there exists $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in X$ such that $\sigma_1 > \sigma_2 > \sigma_3$, can be done in polynomial time. From now on we focus on this task.

Since each of the alternatives is a top for some $i \in N$ (with respect to A), we have $T = A$ and $Q = Q^T$. Let us now consider some set of orderings from Q^T. As they all correspond to triples of alternatives from T, there are only a few possible cases of how they may relate to each other in the preference orders of pairs of agents. Let us take some three alternatives $x, y, z \in T$; there are 21 different combinations of pairs of preference orders that we need to consider (see second column of Table 3.1). All these combinations have their entries in Table 3.1, precomputed according to Definition 3.3.3. To obtain a set of legal orderings for some triple of candidates S and some agents i and j, it suffices to assign these candidates to variables x, y, z

	Comb. of agents $i, j \in N$	Set of legal orderings	2CNF ordering formula
1	$x \succ_i y \succ_i z$ and $x \succ_j y \succ_j z$	$\{(y,x,z),(x,y,z),(z,x,y),(z,y,x)\}$	$(xz \vee zy) \wedge (yz \vee zx)$
2	$x \succ_i y \succ_i z$ and $x \succ_j y \approx_j z$	$\{(y,x,z),(x,y,z),(z,x,y),(z,y,x)\}$	$(xz \vee zy) \wedge (yz \vee zx)$
3	$x \succ_i y \succ_i z$ and $x \approx_j y \succ_j z$	$\{(y,x,z),(x,y,z),(z,x,y),(z,y,x)\}$	$(xz \vee zy) \wedge (yz \vee zx)$
4	$x \succ_i y \succ_i z$ and $x \approx_j y \approx_j z$	$\{(y,x,z),(x,y,z),(z,x,y),(z,y,x)\}$	$(xz \vee zy) \wedge (yz \vee zx)$
5	$x \succ_i y \succ_i z$ and $x \succ_j z \succ_j y$	$\{(y,x,z),(z,x,y)\}$	$(xy \vee xz) \wedge (yz \vee zx) \wedge (yx \vee zy)$
6	$x \succ_i z \succ_i y$ and $x \approx_j y \succ_j z$	$\{(y,x,z),(z,x,y)\}$	$(xy \vee xz) \wedge (yz \vee zx) \wedge (yx \vee zy)$
7	$x \succ_i y \succ_i z$ and $y \succ_j x \succ_j z$	$\{(y,x,z),(x,y,z),(z,x,y),(z,y,x)\}$	$(xz \vee zy) \wedge (yz \vee zx)$
8	$y \succ_i x \succ_i z$ and $x \succ_j y \approx_j z$	$\{(y,x,z),(z,x,y)\}$	$(xy \vee xz) \wedge (yz \vee zx) \wedge (yx \vee zy)$
9	$x \succ_i y \succ_i z$ and $y \succ_j z \succ_j x$	$\{(x,y,z),(z,y,x)\}$	$(xy \vee zx) \wedge (xz \vee zy) \wedge (yz \vee yx)$
10	$z \succ_i x \succ_i y$ and $x \approx_j y \succ_j z$	$\{(y,x,z),(z,x,y)\}$	$(xy \vee xz) \wedge (yz \vee zx) \wedge (yx \vee zy)$
11	$y \succ_i z \succ_i x$ and $x \succ_j y \approx_j z$	$\{(y,z,x),(x,z,y)\}$	$(xy \vee yz) \wedge (xz \vee yx) \wedge (zx \vee zy)$
12	$x \succ_i y \succ_i z$ and $z \succ_j y \succ_j x$	$\{(x,y,z),(z,y,x)\}$	$(xy \vee zx) \wedge (xz \vee zy) \wedge (yz \vee yx)$
13	$x \succ_i y \approx_i z$ and $x \approx_j y \succ_j z$	$\{(y,x,z),(z,x,y)\}$	$(xy \vee xz) \wedge (yz \vee zx) \wedge (yx \vee zy)$
14	$x \succ_i y \approx_i z$ and $y \succ_j x \approx_j z$	$\{(x,z,y),(y,z,x)\}$	$(xy \vee yz) \wedge (xz \vee yx) \wedge (zx \vee zy)$
15	$z \succ_i x \approx_i y$ and $x \approx_j y \succ_j z$	$\{(y,x,z),(x,y,z),(z,x,y),(z,y,x)\}$	$(xz \vee zy) \wedge (yz \vee zx)$
16	$x \approx_i y \succ_i z$ and $x \approx_j y \succ_j z$	$\{(y,x,z),(x,y,z),(z,x,y),(z,y,x)\}$	$(xz \vee zy) \wedge (yz \vee zx)$
17	$x \approx_i y \succ_i z$ and $x \approx_j y \approx_j z$	$\{(y,x,z),(x,y,z),(z,x,y),(z,y,x)\}$	$(xz \vee zy) \wedge (yz \vee zx)$
18	$x \approx_i y \succ_i z$ and $x \approx_j z \succ_j y$	$\{(y,x,z),(z,x,y)\}$	$(xy \vee xz) \wedge (yz \vee zx) \wedge (yx \vee zy)$
19	$x \succ_i y \approx_i z$ and $x \succ_j y \approx_j z$	all permutations of $\{x,y,z\}$	n/a
20	$x \approx_i y \approx_i z$ and $x \succ_j y \approx_j z$	all permutations of $\{x,y,z\}$	n/a
21	$x \approx_i y \approx_i z$ and $x \approx_j y \approx_j z$	all permutations of $\{x,y,z\}$	n/a

Table 3.1: All possible combinations of pairs of agents from Q^T, with corresponding sets of legal orderings (third column). The last column shows a 2CNF representation of each ordering formula (note that we write, e.g., yx instead of $\neg xy$).

and choose an entry in Table 3.1 corresponding to the preference orders of agents i and j.

With Table 3.1 available, we can compute the set Q^T by looking up appropriate values. We illustrate this process with the example below.

Example. *Let the set of alternatives be $A' = \{a, b, c, d\}$ and let the preference profile $\succcurlyeq'$ be as follows ($N' = \{1, 2\}$):*

$$a \approx'_1 b \approx'_1 c \succ'_1 d, \quad c \approx'_2 d \succ'_2 a \succ'_2 b.$$

We see that $T = \{a, b, c, d\} = A'$, as each of the alternatives is top for some agent. We have four possible triples of alternatives and three possible pairs of agents (note that we can make a pair that consists of two copies of the same agent). Let us start with triple $\{a, b, c\}$ and agents 1 and 2. We get the following relation between alternatives from this triple: $a \approx'_1 b \approx'_1 c$ and $c \succ'_2 a \succ'_2 b$, which matches rule no. 4 from Table 3.1 (where $x \leftarrow c$, $y \leftarrow a$ and $z \leftarrow b$) and generates the following set of legal orderings:

$$\{(a,c,b),(c,a,b),(b,c,a),(b,a,c)\}.$$

Similarly, if we now take triple $\{a, b, d\}$ *and agents* 1 *and* 2*, the relation looks as follows:* $a \approx'_1 b \succ'_1 d$ *and* $d \succ'_2 a \succ'_2 b$*, which matches rule no. 10 (with* $x \leftarrow a$*,* $y \leftarrow b$ *and* $z \leftarrow d$*). Therefore it generates the set of legal orderings* $\{(b, a, d), (d, a, b)\}$*. If we follow similar steps for triples* $\{a, c, d\}$ *and* $\{b, c, d\}$*, we will match rule no. 14 for the former and rule no. 6 for the latter, generating the two corresponding sets of legal orderings (*$\{(a, c, d), (d, c, a)\}$ *and* $\{(b, c, d), (d, c, b)\}$*). On top of that, we have to consider pairs that are made of two copies of the same agent (that is* 1 *and* 1*;* 2 *and* 2*). As a result, family* Q *will have* 12 *elements and will be:*

$$
\begin{aligned}
Q = \{&\{(a,c,b),(c,a,b),(b,c,a),\underline{(b,a,c)}\},\\
&\{\underline{(b,a,d)},(d,a,b)\},\\
&\{\underline{(a,c,d)},(d,c,a)\},\\
&\{\underline{(b,c,d)},(d,c,b)\},\\
&\{\underline{(b,a,c)},(b,c,a),(c,a,b),(a,c,b),(c,b,a),(a,b,c)\},\\
&\{\underline{(b,a,d)},(a,b,d),(d,a,b),(d,b,a)\},\\
&\{(c,a,d),\underline{(a,c,d)},(d,a,c),(d,c,a)\},\\
&\{(c,b,d),\underline{(b,c,d)},(d,b,c),(d,b,a)\},\\
&\{(c,b,a),(c,a,b),(a,b,c),\underline{(b,a,c)}\},\\
&\{(d,b,a),(d,a,b),(d,b,c),\underline{(b,a,d)}\},\\
&\{(d,c,a),(d,a,c),\underline{(a,c,d)},(c,a,d),(a,d,c),(c,d,a)\},\\
&\{\underline{(b,c,d)},(c,b,d),(d,c,b),(d,b,c)\}\}
\end{aligned}
$$

Taking order $b >' a >' c >' d$*, we see that for each set* $X \in Q$ *there is at least one sequence* $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in X$ *such that* $\sigma_1 >' \sigma_2 >' \sigma_3$ *(corresponding items for each set are underlined on the listing above). By Lemma 3.3.4, we conclude that* $\succcurlyeq'$ *is top-monotonic and* $>'$ *is its top-monotonic order.*

Set Q consists of $|N|^2 \times \binom{|A|}{3}$ elements, where each element is of the form of one of the sets from the third column of Table 3.1. We want to find a linear order $>$ over the set of alternatives such that for each $X \in Q$ we can find at least one sequence $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in X$ with $\sigma_1 > \sigma_2 > \sigma_3$. It turns out that we can express this problem as an instance of the SAT-2CNF problem.

To illustrate this approach, let us consider rule no. 5 from Table 3.1, with output $\{(y, x, z), (z, x, y)\}$. If the desired order $>$ exists, it has to satisfy condition:

$$(y > x > z) \vee (z > x > y) \tag{3.2}$$

We can create a similar formula for each set of legal orderings from Q, and then expect the order $>$ (if it exists) to satisfy the conjunction of these formulas.

The crucial observation is that Eq. (3.2) (as well as formulas corresponding to all the other rules from Table 3.1) can be expressed in conjunctive normal form with two literals per clause (2CNF). To this end, we take our logical variables to be xy, xz, and yz and we interpret them as representing appropriate relations under order $>$ (e.g., xy is true if $x > y$ holds; we also write, e.g., yx as an abbreviation for $\neg xy$). Then, Eq. (3.2) can be equivalently expressed as:

$$(yx \wedge xz \wedge yz) \vee (zx \wedge xy \wedge zy)$$

or, equivalently, as:

$$(\neg xy \wedge xz \wedge yz) \vee (\neg xz \wedge xy \wedge \neg yz).$$

This formula, on the other hand, is true if and only if the following one is (see comments below):

$$(xz \vee xy) \wedge (yz \vee \neg xz) \wedge (\neg xy \vee \neg yz). \tag{3.3}$$

To check that the two formulas above are indeed equivalent, one may try all possible truth assignments for our variables (as there are three variables only, we need to consider eight possible assignments). For example, if we take the following one:

$$xy \leftarrow True, \quad xz \leftarrow True, \quad \text{and} \quad yz \leftarrow False$$

then both formulas evaluate to $False$, whereas if we take:

$$xy \leftarrow False, \quad xz \leftarrow True, \quad \text{and} \quad yz \leftarrow True$$

then they both evaluate to $True$. Formula (3.3) is in the 2CNF form and, in fact, we can represent each of the possible sets of legal orderings using 2CNF formulas, as presented in Table 3.1 (the only exception is that rules 19, 20, and 21 do not

impose any restrictions on the orderings and, thus, do not generate formulas at all; we show later why this does not affect the properties of the order that we want to compute).[2]

To summarize, we proceed as follows. For each three alternatives x, y, and z and each two agents i and j, we look-up the SAT-2CNF formula in Table 3.1 that corresponds to this setting (for each pair of alternatives p and q, we arbitrarily choose which of the literals pq or qp that arises is represented as a variable and which is represented as this variable's negation; note that if some literal does not arise ever in any of the rules, we do not create its variable). We form the *global ordering formula* by taking the conjunction of all these formulas. We now show that a top-monotonic order exists if and only if the global ordering formula is satisfiable.

Observation 1. *A top-monotonic order for $\succcurlyeq$ exists if and only if there is an assignment for the variables that satisfies the global ordering formula.*

To prove this result, we first show that if $\succcurlyeq$ has a top-monotonic order then the global ordering formula is satisfiable. Let $>'$ be this top-monotonic order. Now consider a variable assignment for the global ordering formula to be such that for each two candidates p and q, we set the literal $pq = 1$ if $p >' q$ (note that it may, in fact, mean setting variable qp to $False$, depending which one of pq and qp is used as a variable in the global ordering formula and which one is represented as its negation). It is easy to see that such a variable assignment is a valid solution for the global ordering formula.

Now it remains to show that if there is a satisfying assignment for the variables of the global ordering formula, then the profile is top-monotonic. Let us assume that the global ordering formula has a satisfying assignment and let relation $>'$ be defined for each pair of alternatives $p, q \in A$ as follows: We set $p >' q$ exactly if the literal pq evaluates to $True$. By definition, relation $>'$ satisfies the global ordering formula and we only need to show that it, indeed, is an order over A. We show that this is the case by considering the three requirements that a strict order must satisfy:

(a) Relation $>'$ is *irreflexive* because we do not have literals of the form xx in our global ordering formula.

[2]The reader may wonder how we derived Formula (3.3) from Formula (3.2), or how we deduced that it can be translated to an equivalent 2CNF form. We are afraid that wishful thinking is our only response here; we wanted to find a 2CNF formula and we made a brute-force search for one that would work.

(b) Relation $>'$ is *asymmetric* because for each $x, y \in A$, if $x >' y$ then it is not the case that $y >' x$ because for each $x, y \in A$, $xy \equiv \neg yx$ and, so, it cannot be that both xy and yx are set to 1 at the same time.

(c) Relation $>'$ is *transitive*, that is, for every triple $x, y, z \in A$ if $x >' y$ and $y >' z$ then $x >' z$. This follows from the fact that for every triple $x, y, z \in A$ we need to satisfy a formula that corresponds to a set of legal orderings. It is clear that any element from the set of legal orderings has to satisfy the transitivity condition (as such an element represents an order of alternatives). The only exception is when a triple $x, y, z \in A$ matches either of the rules 19, 20, or 21 from Table 3.1 for every possible pair of agents $i, j \in N$. However, in such a case it is easy to see that it must be the case that $y \approx_k z$ for every $k \in N$. Therefore we can ignore this case as alternatives y and z are indistinguishable form each other for all the agents. We pick one of them to use in the algorithm (and remove the other one from the profile); if it turns out that a top-monotonic order exists, then we place these candidates next to each other in this order (the algorithm computes the position of one of them in the top-monotonic order, and the other one can be put on either of its sides).

So far, we have not argued that $>'$ is a total order and, indeed, it may be partial. We let $>^*$ be a linear extension of $>'$; we know that such an extension exists due to the order-extension principle. Since $>^*$ satisfies the global ordering formula (as it is an extension of $>'$) and it is a linear order, $>^*$ is a top-monotonic order for $\succcurlyeq$.

Finally, we note that, since the global ordering formula is in conjunctive normal form with at most two variables per clause, there is a simple polynomial-time algorithm that checks if it is satisfiable and, if so, produces a satisfying assignment. Further, the formula itself is of length polynomially bounded in the number of candidates and agents (we need $O(N^2 \cdot |A|^3)$ subformulas from Table 3.1, with at most $O(|A|^2)$ variables). □

Example. *Let us consider the same setting as in Example 3.3.3, that is, let* $A' = \{a, b, c, d\}$ *be a set of alternatives and let* $\succcurlyeq'$ *be a preference profile defined as follows:* $N' = \{(a \approx'_1 b \approx'_1 c \succ'_1 d), (c \approx'_2 d \succ'_2 a \succ'_2 b)\}$. *Based on the family* Q *obtained in Example 3.3.3, we compute the conjunction of 2-CNF ordering formulas which looks*

as follows (we removed duplicate clauses):

$$(ab \vee bc) \wedge (ba \vee cb) \wedge (ac \vee da) \wedge (ad \vee dc) \wedge (cd \vee ca) \wedge (ab \vee ad) \wedge$$
$$(bd \vee da) \wedge (ba \vee db) \wedge (bc \vee db) \wedge (bd \vee dc) \wedge (cd \vee cb) \wedge (ad \vee db) \wedge$$
$$(cd \vee da) \wedge (cd \vee db) \wedge (ab \vee bc) \wedge (ab \vee bd) \wedge (cb \vee bd)$$

Now we create an instance of SAT-2CNF problem with six variables, ab, ac, ad, bc, bd, and cd:

$$(ab \vee \underline{bc}) \wedge (\underline{\neg ab} \vee \neg bc) \wedge (\underline{ac} \vee \neg ad) \wedge (\underline{ad} \vee \neg cd) \wedge (\underline{cd} \vee \neg ac) \wedge (ab \vee \underline{ad}) \wedge$$
$$(\underline{bd} \vee \neg ad) \wedge (\underline{\neg ab} \vee \neg bd) \wedge (\underline{bc} \vee \neg bd) \wedge (\underline{bd} \vee \neg cd) \wedge (\underline{cd} \vee \neg bc) \wedge (\underline{ad} \vee \neg bd) \wedge$$
$$(\underline{cd} \vee \neg ad) \wedge (\underline{cd} \vee \neg bd) \wedge (ab \vee \underline{bc}) \wedge (ab \vee \underline{bd}) \wedge (\neg bc \vee \underline{bd})$$

We note that the above SAT-2CNF instance is satisfiable and one of the possible assignments is $ab \leftarrow False$, $ac \leftarrow True$, $ad \leftarrow True$, $bc \leftarrow True$, $bd \leftarrow True$, $cd \leftarrow True$. *Literals that are true according to this assignment are underlined. We see that each clause has at least one literal that is true. Based on the assignment we now get a strict partial order* $>'$ *defined as follows:* $>' = \{(b >' a), (a >' c), (a >' d), (b >' c), (b >' d), (c >' d)\}$, *which happens to also be a linear order over* A'. *Thus the computed order is* $b >' a >' c >' d$. *As* $>'$ *satisfies our global ordering formula, we conclude that it is a top-monotonic order for* $\succcurlyeq'$.

3.3.4 Main Proof

Our main result holds without the assumption that the profile is narcissistic. The full proof is more complicated due to the fact that there are more rules to be considered, with more cases where it is not clear if the rules indeed lead to a strict linear order. Fortunately, all these obstacles can be dealt with using arguments that, in principle, are similar to those presented already. The proof follows exactly the same path as that of Theorem 3.3.5, but takes into account that the set Q^{NT} may be non-empty. Thus, in addition to creating 2CNF formulas out of the sets of legal orderings from Q^T (see Table 3.1), we also do the same with the sets from Q^{NT} (using Table 3.2). As in the proof of Theorem 3.3.5, we show that the global formula has a satisfiable assignment if and only if a top-monotonic order exists.

	Comb. of agents $i, j \in N$	Set of legal orderings	2CNF ordering formula
1	$x \succ_i \omega \succ_i y$ and $y \approx_j x \succ_j \omega$	$\{(\omega,x,y),(y,x,\omega),(x,\omega,y),(y,\omega,x)\}$	$(xy \vee y\omega) \wedge (\omega y \vee yx)$
2	$x \succ_i y \approx_i \omega$ and $y \succ_j x \succ_j \omega$	$\{(\omega,x,y),(y,\omega,x),(x,\omega,y),(y,x,\omega)\}$	$(xy \vee y\omega) \wedge (\omega y \vee yx)$
3	$x \succ_i y \succ_i \omega$ and $y \succ_j \omega \succ_j x$	$\{(y,\omega,x),(\omega,y,x),(x,\omega,y),(x,y,\omega)\}$	$(yx \vee x\omega) \wedge (\omega x \vee xy)$
4	$x \succ_i y \approx_i \omega$ and $y \succ_j \omega \succ_j x$	$\{(x,\omega,y),(y,\omega,x)\}$	$(xy \vee y\omega) \wedge (x\omega \vee yx) \wedge (\omega x \vee \omega y)$
5	$x \succ_i y \approx_i \omega$ and $y \approx_j x \succ_j \omega$	$\{(\omega,x,y),(y,x,\omega),(x,\omega,y),(y,\omega,x)\}$	$(xy \vee y\omega) \wedge (\omega y \vee yx)$
6	$x \succ_i y \approx_i \omega$ and $y \succ_j x \approx_j \omega$	$\{(y,\omega,x),(x,\omega,y)\}$	$(xy \vee y\omega) \wedge (x\omega \vee yx) \wedge (\omega x \vee \omega y)$
7	$x \succ_i y \succ_i \omega$ and $y \succ_j x \succ_j \omega$	all permutations of $\{x,y,\omega\}$	n/a
8	$x \succ_i y \succ_i \omega$ and $y \approx_j x \succ_j \omega$	all permutations of $\{x,y,\omega\}$	n/a
9	$x \approx_i y \succ_i \omega$ and $y \approx_j x \succ_j \omega$	all permutations of $\{x,y,\omega\}$	n/a

Table 3.2: All possible settings of preferences for pairs of agents $i, j \in N$ over the set of alternatives x, y, ω, where $x \in t_i(A)$, $y \in t_j(A)$ and $\omega \in A \setminus T$, with corresponding sets of legal orderings (third column). The last column shows 2CNF representations of each ordering formula (note that we can write yx instead of $\neg(xy)$).

Theorem 3.3.6. *Let A be a set of alternatives, N be a set of agents, and $\succcurlyeq$ be a preference profile over A. The problem of determining whether a top-monotonic order of $\succcurlyeq$ exists (and computing it) is polynomial-time solvable.*

Proof. We extend the proof of Theorem 3.3.5 by considering that Q^{NT} may be non-empty (if $Q^{NT} = \emptyset$ we can use Theorem 3.3.5 directly). This will be reflected in the number of additional combinations of alternatives that we have to consider. This will lead to an extended list of rules for the sets of legal orderings that, as we will show later on, can also be represented in the 2CNF form. This will allow us to use the same argument as in the proof of Theorem 3.3.5 to reduce the problem of determining a top-monotonic order to the SAT-2CNF problem.

Let us consider some element X from a non-empty set Q^{NT}. By the definition of Q^{NT}, X is a set of legal orderings for a pair of agents $i, j \in N$ and a triple of distinct alternatives x, y, ω, such that $x \in t_i(A)$, $y \in t_j(A)$, and $\omega \in A \setminus T$. There are exactly nine different possible combinations of preferences for the pair of agents i, j over the alternatives x, y, ω. We list all these combinations along with their corresponding sets of legal orderings in Table 3.2. Note that there are much fewer combinations than in Table 3.1 and this is because set Q^{NT} has a more strict definition than Q^T. Specifically, it is required that x is among the most preferred alternatives for agent i, y is among the most preferred alternatives for agent j, and ω can never be the most preferred alternative.

Similarly to how we proceeded in the proof of Theorem 3.3.5, we can represent each of the sets of legal orderings from Table 3.2 as a 2CNF ordering formula. This time rules 7–9 do not introduce any constraints on the *top-monotonic* order.

We see now that we can make an instance I of a SAT-2CNF problem by taking the conjunction of the 2CNF formulas for the matching sets of legal orderings for all the elements from both Q^T and Q^{NT} by using the rules from Table 3.1 and 3.2 correspondingly, and by following the methodology from the proof of Theorem 3.3.5. We make similar claim as in that proof that a top-monotonic order exists for $\succcurlyeq$ if and only if I has a solution, and, if so, that the top-monotonic order is a linear extension of the order $>'$ induced by the assignment of the variables for the solution of I (in the same way as in the proof of Theorem 3.3.5). We can use almost all the same arguments as in the proof of Theorem 3.3.5 to prove that our reduction to SAT-2CNF is correct. The only exception regards showing that the transitivity property is fulfilled for the relation $>'$. We see that the transitivity property is fulfilled for all the triples $x, y, z \in T$ (by the argument from the proof of Theorem 3.3.5). We also see that it is fulfilled for all the triples x, y, ω, such that there are agents i and j such that $x \in t_i(A)$, $y \in t_j(A)$, $\omega \in A \setminus T$ and the preferences of these agents map to one of the rules 1–6 from Table 3.2 (the rules are "enforcing" transitivity, in the same way as in Theorem 3.3.5). The only situation that remains to be handled occurs if there exist $x', y' \in T$, $\omega' \in A \setminus T$, such that for each pair of agents $i, j \in N$, such that $x' \in t_i(A)$ and $y' \in t_j(A)$, the preferences of agents i and j over alternatives x', y', ω' map to the rules 7–9 from Table 3.2. In this case there is no 2CNF formula we can add to instance I that would "enforce" the transitivity between x', y' and ω', yet I may contain variables that correspond to the relations between each pair of these alternatives. We address this issue in the following lemma.

Lemma 3.3.7. *Let A be a set of alternatives, N be a set of agents, and $\succcurlyeq$ be a preference profile over A. Let T be a subset of A such that it contains all the alternatives that are top in A of some agent from N. If there exists a triple of alternative x, y, ω, where $x, y \in T$ and $\omega \in A \setminus T$, such that for every possible pair of candidates $i, j \in N$, with $x \in t_i(A)$ and $y \in t_j(A)$, the agents i and j's preferences over alternatives x, y, ω always match one of the rules 7–9 from Table 3.2, then SAT-2CNF instance I obtained for the set of agents N and the set of alternatives A will either not contain variables that correspond to the relations between alternatives (x, y), (x, ω) and (y, ω) or each order $>$ that is obtained from each solution of I will fulfill transitivity property for the alternatives x, y, ω.*

Proof. It is sufficient to show that if instance I is satisfiable and contains variables that correspond to at least one of (x, y), (x, ω) or (y, ω), then for all satisfying

truth assignments for I the transitivity property is fulfilled for x, y and ω.[3] To the contrary, let us assume that instance I is satisfiable and for some solution of I the transitivity is not fulfilled for x, y, ω. For that to be true, I has to contain all three literals that map to (x, y), (x, ω) and (y, ω) as otherwise it would be impossible to encode the loss of transitivity. Without loss of generality, let us assume that the variables are xy, $x\omega$ and $y\omega$. Now, for the transitivity not to be satisfied they can have one of the two following assignments in the solution of I:

$$\begin{gathered} xy \leftarrow True, \quad x\omega \leftarrow False, \quad y\omega \leftarrow True, \\ \text{or} \\ xy \leftarrow False, \quad x\omega \leftarrow True, \quad y\omega \leftarrow False. \end{gathered} \tag{3.4}$$

We have assumed in the lemma statement that for all possible pairs of agents when considering triple $\{x, y, \omega\}$, we always get rules 7–9 from Table 3.2, which do not output any clauses. Yet, we need literals xy, $x\omega$ and $y\omega$ to be a part of the 2CNF formula that we built. Therefore there have to be additional alternatives in our election that, when considered jointly with x, y and ω, match rules that output these literals. So, let us assume that there is an alternative a such that when considering $\{a, x, \omega\}$, the triple matches one of the rules 1–6 from Table 3.2 and, therefore, generates clauses for our formula that include literal $x\omega$. Let us also assume that there exists an alternative b such that when considering triple $\{b, y, \omega\}$, we match rules that generate literal $y\omega$. It might be the case that $a = b$ but this is irrelevant for the rest of the proof, so we will not consider it as a separate case. We note that triples $\{a, x, \omega\}$ and $\{b, y, \omega\}$ can only match rules from Table 3.2 (because ω is never top in A), and hence both a and b must be in T. Finally, we also need literal xy to be generated. We will show later that for that to happen we do not need any additional alternatives in our election, and that either for $\{x, y, a\}$ or $\{x, y, b\}$ we will always get a rule that uses xy (note that a, b, x, and y are in T, so for the rules generated by aforementioned triples we use Table 3.1).

Based on the above discussion, we claim that if for a set of alternatives A that includes x, y, ω, a and b, set of agents N, and preference profile $\succcurlyeq$, there exists an instance I that includes literals xy, $x\omega$, and $y\omega$ and that can be satisfied with one of the assignments from Eq. 3.4, then we can find $N' \subseteq N$, $\|N'\| \leq 4$, such that instance I' computed for agents N', alternatives $A' = \{x, y, \omega, a, b\}$, and preference

[3]Note that the existence of x, y and ω in the set of alternatives alone does not imply the listed variables will be a part of I because we only add variables when needed by at least one of the rules we generate

profile $\succcurlyeq$ is satisfiable, the set of literals in I' is a subset of literals from I, and the matching assignment for the solution of I is a solution for I'. In other words, we are going to show that if our lemma were false, then there would be an election with four agents and five candidates that would form a counterexample.

Let us first consider a pair of agents i, j from N that, when considered, would output a rule that includes literal $x\omega$ (if there are many such pairs, we pick one arbitrarily). It means that both agents i and j have at least one of x and a as top in A, and that ω is not top in A of neither i nor j (in fact, it is not top of any agent from A). Formally, we have:

$$t_i(A) \cap \{x, a\} \neq \emptyset \quad , \quad t_j(A) \cap \{x, a\} \neq \emptyset \quad , \text{ and } \quad \omega \notin t_i(A) \cup t_j(A).$$

We note that the same holds true for A'. This follows from the fact that if x were top in A of some agent, it would also be top in all subsets of A that include x. Also ω cannot be top in A' of neither i nor j. This is true because if x is top in A of i, then since ω is not top in A of i, agent i has to prefer x over ω, and therefore ω is not top in A' for i too. Otherwise, if x is not top in A of i, then a is for sure and the same reasoning can be used to show that ω is not top in A' of i. Finally, we can show the same for j, that is, that ω is not top in A' of j.

We select the second pair k, l of agents from N that are responsible for generating literal $y\omega$ in I in a similar way. We now see that the rules generated for agents $N' = \{i, j, k, l\}$ and alternatives $A' = \{x, y, \omega, a, b\}$ are the same under both I' and I. This is true because the preferences of the agents in N' with respect to the alternatives from A' remain unchanged in I'. Therefore, given our assumption that there is a solution of I where transitivity is not fulfilled for x, y, ω we see there exists a matching solution for I' with the same characteristic.

As a result of the above, we can see that assuming I exists we can find an instance I' that only has five alternatives and no more than four agents for which a satisfying truth assignment breaks transitivity. If we could show that no such instance I' exists, then we would prove our lemma by showing our assumption on the existence of I is not true.

So far we have not been able to find an easy-to-follow proof for showing that a counterexample for our lemma cannot be found in an election with up to four agents and five candidates. But since the number of possible problem instances is bounded by an acceptably small constant, we were able to perform a computer-assisted proof by exhaustive search where we tried all possible instances I'. The computer program

we used works as follows:

- ▷ For a set of alternatives $A = \{x, y, \omega, a, b\}$, each possible set of agents N of size no bigger than four, and each possible preference profile $\succcurlyeq$ such that ω is not top on A for any agent from N and the profile leads to the literals $x\omega$ and $y\omega$ being generated, do:
  - ▷ Verify that literal xy is generated for $\succcurlyeq$.[4]
  - ▷ If $\succcurlyeq$ is *top-monotonic* (we use Definition 3.3.1 and test every possible order of alternatives) then check if instance I can be satisfied for one of the initial assignments of literals from Eq. 3.4; if it can, then we have found a profile that fails our criteria.

Below we describe the methodology that our program follows for scanning through all the possible combinations in the steps described above. Along the way we also provide an upper bound on the number of cases the program needs to consider.

For the initial step, it suffices to consider orderings that do not satisfy transitivity for x, y and ω, that is, orderings which for pairs (x, y), (x, ω) and (y, ω) match one of the options from Eq. (3.4). This brings the number of orderings that we need to consider from $2^{10} = 1024$ (for each of the 10 pairs of alternatives we need to choose in which of two ways it is ordered) down to $2 \cdot 2^7 = 256$ (we have two versions for arranging the three crucial pairs and we need to consider all possible arrangements for the remaining 7 pairs).

To get a sense for how many possible preference profiles we need to consider, let us start with an upper bound on the number of possible votes over five alternatives that we need to consider. For that we take any possible arrangement of five alternatives (in one of 5! ways) and in four places in between agents we place $\succ$ or $\approx$ symbol in 2^4 ways. This yields $5!\,2^4 = 1920$ possible combinations. This number is pretty large, given that we need to consider four agents. This would lead to $1920^4 \approx 10^{13}$ possible profiles.

To reduce this number, we take advantage of the fact that we do not want to consider votes that place ω among top choices. Furthermore, we note that we only want to consider profiles in which no rules are being generated for triple (x, y, ω), so for each pair of agents we always expect that triple to fall under rules 7–9 from Table 3.2. For that to happen, if x or y is among top choices in a vote we also

[4]For every profile where literals $x\omega$ and $y\omega$ were generated, literal xy was generated as well.

require that $x \succ \omega$ and $y \succ \omega$. This follows from the fact that a unique feature of rules 7–9 is that for both agents considered for generating the rule we always have $x \succ \omega$ and $y \succ \omega$.

To illustrate what votes are worth considering for our cases, let us look at the following examples. Vote $x \approx b \succ y \approx a \succ \omega$ is fine, as even though x is top of that agent, we also have $y \succ \omega$. Vote $a \succ \omega \approx b \succ x \succ y$ is also fine because neither x nor y is top of that agent. On the other hand, vote $y \approx a \succ x \approx b \approx \omega$ is not fine, because y is top while $x \approx \omega$.

To count the number of possible votes given all the constraints described above, we list four possible shapes of a valid vote:

Shape A: $\underline{x\ \ } \succ \underline{y\ \ } \succ \underline{\omega\ \ }$ – each group (marked with an underline) represents a part of a vote where remaining alternatives can be added. Alternatives within a group can be reordered and separated with either $\approx$ or $\succ$. The only exception is the first group, where we can only use $\approx$ and hence the order of alternatives does not matter.

Shape B: $\underline{y\ \ } \succ \underline{x\ \ } \succ \underline{\omega\ \ }$ – a symmetric case to the previous one, where x and y are swapped.

Shape C: $\underline{x \approx y\ \ } \succ \underline{\omega\ \ }$ – similarly here groups can be extended by adding remaining alternatives (a and b); in the first group only equality ($\approx$) can be used between the alternatives.

Shape D: $\underline{\ \ } \succ \underline{x\ ?\ y\ ?\ \omega\ \ }$ – in this case we require that the first group contains at least one alternative. We use ? in the second group to indicate the alternatives there can be reordered and either $\succ$ or $\approx$ can be used in place of ?.

We want to count the number of possible votes for each of the cases defined above. But first let us consider some simplified cases that will help with further calculations. If we have a group that contains two alternatives (say a and b), and in the group we can use either $\approx$ or $\succ$ to separate the alternatives, then there are three possible ways to order the alternatives in such a group, $a \succ b$, $b \succ a$ and $a \approx b$. Now let us consider a group of three alternatives. In such a case, there are 6 ways in which only $\succ$ is used, 6 ways with one $\succ$ and one $\approx$ (we have 3 unique ways when $\approx$ is between the first pair and 3 ways when it is between the second pair), and a single way with two $\approx$ symbols. In total, a group of three alternatives yields 13 unique ordering combinations. In a similar way, we can calculate the result for a group of four alternatives, which has 85 unique orderings.

Using the above we can count the number of possible orderings for each shape of a valid vote presented above:

Shape A: We first consider that we choose two different groups to add a and b. This can be done in 6 different ways. In that case, alternatives in these two groups (of two elements each) can be ordered in 3 unique ways (see above). This gives $6 \cdot 3 \cdot 3 = 54$ combinations in total. Another option is to select one group in 3 possible ways and add both a and b to it. Groups will have three alternatives so that gives $3 \cdot 13 = 39$ possible orderings. We sum up both cases to get $54 + 39 = 93$ possible combinations.

Shape B: This case is symmetric to Case 1 and similarly yields 93 possible combinations.

Shape C: In this case we can put both a and b in the first group in only one way (we need to use equality). If we decided to place both a and b in the second group we can do that in 13 ways (a group of three alternatives). Finally, if we put one alternative in the first and one in the second group, we get $2 \cdot 3 = 6$ ways. This sums up to $1 + 13 + 6 = 20$ possible votes.

Shape D: Similarly, for the last shape we consider placing both a and b in the first group. Since there are three alternatives in the second group, we can have 13 ordering combinations in that case. As we always need to place one alternative in the first group the only other way to create a valid ordering is by placing one alternative in the first group and the other one in the second. This can be done in 2 ways and the group will contain four alternatives so it can be ordered in 85 ways. In total we get $13 + 2 \cdot 85 = 183$ possible votes.

Now, let us get an upper bound for the number of possible profiles of four agents we want to consider. Given the above, we have $93 + 93 + 20 + 183 = 389$ different votes that we need to consider. Among these, we only have $93 + 20 = 113$ possible votes that rank x on top. We want to enforce that x and y are both top on A for some agents from N, so we can pick first agent's vote from this selection of 113 votes, and then the remaining ones from the full list of 389 possibilities. Note that, while following this methodology, we still may get combinations in which y is not on top for any of the agents. If we eliminated this possibility too, the upper bound would go down even further, but it would complicate the way of generating profiles. Instead, we filter out profiles that do not match our requirements. With that taken

into account, we have $113 \cdot 389^3 < 7 \cdot 10^9$ possible profiles we should consider, which puts it in a reasonable range for a middle-end desktop hardware to process within a matter of seconds. As mentioned earlier, the profiles we can generate this way may not meet all the constraints required by the algorithm. Also, the generated profile may not lead to all the required literals being generated. We can therefore rule out even more profiles at the early step and avoid further calculations.

Our program has verified all possible combinations of instances with five alternatives and four or fewer agents and have not encountered an instance that would satisfy the algorithm. □

By Lemma 3.3.7, we see that the case in which transitivity property cannot be fulfilled even though solution for I exists is not possible. We therefore see that the transitivity property is satisfied for $>'$. Similarly to the proof of Theorem 3.3.5, we state now that I has a solution if and only if $\succcurlyeq$ is top-monotonic, and if the solution exists, then a linear extension of the order $>'$ induced by the solution is a top-monotonic order of $\succcurlyeq$. We also see that I can be computed and solved in polynomial-time with respect to the numbers of agents and alternatives. The only difference comparing to the proof of Theorem 3.3.5 is that in the rules lookup phase we also consider some triples made of candidates that are not top in A for any agent. The upper bound for the number of literals and variables in I are the same, $3 \cdot N^2 \cdot \|A\|^3$ and $\|A^3\|$ correspondingly, as each combination of agents and alternatives will match at most one rule (either from Table 3.1 or from Table 3.2). □

3.4 Conclusions

We have presented a novel way of modeling restricted domain problems by reducing them to boolean satisfiability problems and used this approach to provide a first polynomial-time algorithm for recognizing top-monotonic profiles. The presented methodology is more universal than the existing restricted domain recognition algorithms as it can be applied to single-peaked, single-crossing and top-monotonic domains. On top of that, the formulation of the problem in terms of a SAT-2CNF instance provides a more approachable interface for dealing with top-monotonic profiles from the computational social choice standpoint than the original definition. We hope for our method to enable further researchers to consider top-monotonicity when studying algorithmic impact of restricted domains in different areas of computational social choice. For example, it is natural to ask if the Chamberlin–Courant

rule is polynomial-time solvable under top-monotonic preferences (it is under single-peaked [11] and single-crossing ones [81]). It is also interesting to compare the notion of top-monotonicity to that of value-restricted profiles [78].

Chapter 4

Election Control under Single-Crossing Domain

4.1 Introduction

In this chapter we discuss the complexity of election control problems under the single-crossing domain. We present our polynomial-time results for plurality, Condorcet and approval voting rules (although, to be precise, for the case of approval we use voter-interval domain that adapts the notion of single-crossingness to systems with dichotomous preferences) for all the combinations of constructive and destructive control by adding and deleting candidates or voters.

Election control is an important aspect that requires consideration every time we think of using voting systems (regardless of whether we speak about political elections, use of voting in multiagent-systems [32], recommendation algorithms [44, 58] or in any other aspects where voting can be used). In essence, election control problems model settings where someone wishes to impact a result of the election by changing the setting under which the election is held. The impact made can be either constructive (we want to ensure that some candidate wins the election) or destructive (we want a given candidate not to win). In our case we focus on the means of control by modifying the sets of alternatives or agents taking part in the election.

The study on computational complexity aspects in the area of election control problems has been started by Bartholdi, Tovey, and Trick [8]. They argued that some of the popular voting rules are "resistant" to the considered election control problems because computing a successful control strategy is NP-hard. Their results

have become a subject to further research aiming to determine how strong of an assumption it is by providing heuristic [72] or approximation methods [18] bypassing worst-case hardness results. One of the most interesting directions in this stream of research was presented by Faliszewski et al. [37] who were the first to consider restricted voting profiles in the context of election control (earlier Walsh [84] considered single-peaked elections in the context of strategic voting and Conitzer [21] considered it for winner determination[1]). In their work they show that when considering single-peaked electorates, many of the control problems proven to be NP-hard in the unrestricted case, can be solved in polynomial-time. A natural extension of their work is to consider other restricted domains, such as single-crossing elections, and to study the complexity of election problems in this context. Both single-peaked and single-crossing electorates, as argued by economists [67] may naturally arise in many settings. This makes them equally viable to provide an insight on how the election control algorithms can perform in the settings that resemble more real life scenarios.

It is important to note that the notions of single-peakedness and single-crossingness do not relate much to each other. They are two disjoint domains, and hence it is not possible to directly transfer any of the results achieved within one domain to the other. Despite that, for both of these notions contructive/destructive control problems by adding/deleting candidates/voters are polynomial-time solvable under Plurality, Condorcet and approval voting rules. We provide the summary of our results for single-crossing domain in Table 4.1 along with the single-peaked results and the results for the unrestricted cases.

Acknowledgments

Results presented in this chapter have been published in a paper co-authored by myself and Piotr Faliszewski, titled *How Hard is Control in Single-Crossing Elections?* [59]. My contributions to that paper include all polynomial-time algorithms for the discussed control problems under the single-crossing domain.

[1]Naturally, we mention here computational aspects of the uses of restricted domains. In classic social choice restricted domains were used much earlier and to a much greater extent [12,13,62,74].

Problem		Plurality Un./SP/SC	Condorcet Un./SP/SC	Approval Un./SP/VI
CC	AC	NP-com./P/**P**	–	–
	DC	NP-com./P/**P**	P/P/P	P/P/P
	AV	P/P/P	NP-com./P/**P**	NP-com./P/**P**
	DV	P/P/P	NP-com./P/**P**	NP-com./P/**P**
DC	AC	NP-com./P/**P**	P/P/P	P/P/P
	DC	NP-com./P/**P**	–	–
	AV	P/P/P	P/P/P	P/P/P
	DV	P/P/P	P/P/P	P/P/P

Table 4.1: The complexity of control problems for plurality, Condorcet, and approval rules, for the unrestricted domain (denoted "Un.") [8, 47], the single-peaked domain (denoted "SP") [15, 37], the single-crossing domain (denoted "SC"), and the voter-interval domain [28] (denoted "VI"). Symbol "–" means that the given system is *immune* to the type of control in question (i.e., it is impossible to achieve the desired effect by the action this control problem allows; see the works of Bartholdi et al. [8] and Hemaspaandra et al. [47] for the immunity results). Highlighted results (in bold) are presented in this thesis.

4.2 Preliminaries

For the most part of this chapter we discuss single-crossing elections under ordinal model of elections. For that, the reader may wish to recall Definition 2.1.5 of a single-crossing profile by Mirrlees [62] and Roberts [74] (a separate definition of the single-crossing notion is discussed in section 4.5 where we discuss approval voting).

Further, for a single-crossing election $E = (C, V)$, by a vector $\hat{V} = (v_1, ..., v_n)$ we denote the single-crossing order of the agents. That is, vector $\hat{V}$ contains all the agents from V and they are ordered so that the election is single-crossing with respect to this ordering. Throughout this chapter we also use the notation $t_{x,y}$ as introduced in Definition 2.1.5. When combined with the $\hat{V}$ notation, we note that for each pair of alternatives $x, y \in C$ the number $t_{x,y}$ is an index in vector $\hat{V}$ such that all the agents from $\hat{V}$ up to this index prefer x to y.

Example. *Consider an election* $E = (C, V)$*, where* $C = \{a, b, c\}$ *and* $V = \{v_1, v_2, v_3\}$*. The voters have preference orders:*

$$v_1 \colon a \succ b \succ c,$$
$$v_2 \colon b \succ a \succ c,$$
$$v_3 \colon c \succ b \succ a.$$

We see that E is single-crossing and that $\hat{V} = (v_1, v_2, v_3)$ is a single crossing order of the voters in E. We also have that $t_{a,b} = 1$, $t_{a,c} = 2$ and $t_{b,c} = 2$.

Finally, we note that it is not possible to form a non single-crossing election only by deleting agents or alternatives from an election that is already single-crossing. That is, if election (C, V) is single-crossing, then for each $V' \subseteq V$, (C, V') is single-crossing, and also for each $C' \subseteq C$, (C', V) is single-crossing. This property makes it possible to think of election control problems in the context of single-crossing domain. We always require an election with all the agents and all the alternatives taken into account for the control problem to be single-crossing. That is, when we speak of a control problem for single-crossing elections, we require that all the agents in a given instance (both registered and not registered) have single-crossing preferences regarding all the alternatives (both registered and not registered). For example, in control by adding candidates we require election $(C \cup A, V)$ to be single-crossing (and hence for any subset $A' \subseteq A$ the election $(C \cup A', V)$ is single-crossing too).

4.3 Plurality

We begin with our results for the Plurality voting rule. As the voter control problems are polynomial-time solvable in the unrestricted case for Plurality [8, 47] we focus on candidate control only.

Before we move on to the algorithms we introduce some new terminology.

Definition 4.3.1. *Let $I = (C, A, V, p, k)$ be an instance of the control by adding candidates problem for the Plurality rule. For a set $A' \subseteq A$ and an agent $a \in C \cup A$, we write that a* reduces *the score of p in election $(C \cup A', V)$ if and only if $score^P_{((C \cup A') \setminus \{a\}, V)}(p) > score^P_{(C \cup A' \cup \{a\}, V)}(p)$.*

In other words, if candidate $a \in A'$ reduces the score of p in $(C \cup A', V)$, then adding a to election $(C \cup A' \setminus \{a\}, V)$ will decrease the score of p. It is important to note that under Plurality, introducing a new alternative to the election can never increase the score of the candidates already in the election. That is, for each alternative already in the election it can either *reduce* its score (as defined above) or have no impact on the alternative's score.

4.3.1 Constructive Control Cases

Our constructive control results follow by designing a dynamic-programming algorithm. As both adding and deleting cases can be solved using almost identical approaches, later on we define a more general variant of these problems and reduce both of the cases to it. Before we do that, for our dynamic-programming approach we need some pattern in which agents should be considered. A natural choice in our case is to use the single-crossing order of agents denoted by $\hat{V}$. Specifically for that order we make the following observation that will prove to be helpful later on.

Lemma 4.3.2. *In a single-crossing election $E = (C, V)$, if $\hat{V} = (\hat{v}_1, ..., \hat{v}_n)$ is a single-crossing order for E, then for each agent $c \in C$ the set of alternatives who rank c first is a contiguous subsequence of $\hat{V}$.*

Proof. Assume to the contrary that there exist two alternatives $a, b \in C$ and integers k_1, k_2, and k_3, $1 \leq k_1 < k_2 < k_3 \leq n$, such that $\hat{v}_{k_1}$ and $\hat{v}_{k_3}$ rank a first while $\hat{v}_{k_2}$ ranks b first. Clearly, this means that $\hat{V}$ is not a single-crossing order for E (the relative order of a and b changes more than once as we consider the alternatives in the single-crossing order $\hat{V}$). □

The next theorem serves as a tool for solving both adding and deleting cases. It is a variant of Plurality-CCAC with an additionally strengthened requirements. In this case we consider the set of unregistered alternatives to only contain such alternatives that cannot alter the score of our designated alternative p after being added to the election. In addition, we require an exact number of alternatives from the set of unregistered ones to be added, as opposed to the regular control problem where we can add up to some number of alternatives. These assumptions make our settings more convenient to work with. In particular, they guarantee that we know what the score of p is before and after adding new alternatives (we are sure it will not change) and we also know how many alternatives we need to add. Despite the fact that these additional requirements seem very restrictive, we later show how using this restricted variant of our problem we can solve both Plurality-CCAC and Plurality-CCDC in polynomial-time (for single-crossing elections).

Theorem 4.3.3. *A variant of Plurality-CCAC for the case of single-crossing elections, where the set of unregistered candidates does not contain candidates which would reduce the score of the preferred candidate and where it is required to add exactly a given number of candidates, is in* P.

Proof. We give an algorithm based on dynamic programming. Let (C, A, V, p, k) be an instance of Plurality-CCAC satisfying the requirements of the theorem, where C is the set of registered candidates, A is the set of unregistered candidates, V is the set of voters, p is the preferred candidate, and k is the number of candidates that we need to add. Further, let $\hat{V} = (\hat{v}_1, \ldots, \hat{v}_n)$ be a single-crossing order of voters from V (for the candidate set $C \cup A$). For each candidate $c \in C \cup A$, let $\mathit{first}(c)$ and $\mathit{last}(c)$ be, respectively, the minimal and the maximal index of a voter from $\hat{V}$ that ranks c first in election $(C \cup \{c\}, V)$ (it is easy to see that, without loss of generality, we can assume that these values are always well-defined[2]). From Lemma 4.3.2, we know that for a given $c \in C \cup A$ the interval $\hat{v}_{\mathit{first}(c)}, \ldots, \hat{v}_{\mathit{last}(c)}$ consists of all the voters from V which rank c first in $E = (C \cup \{c\}, V)$; let us refer to this interval as the *visibility interval* of candidate c. Considering the election from Example 4.2, we see that visibility interval for candidate a is (v_1), for candidate c is (v_2, v_3), and candidate b is not visible. We also see that the *first* and *last* indices for candidates a and c are:

$$\mathit{first}(a) = \mathit{last}(a) = 1 \qquad \text{and} \qquad \mathit{first}(c) = 2,\ \mathit{last}(c) = 3.$$

We provide further examples of the notation introduced here and below as Example 4.3.1, a bit later.

For each set $X \subseteq C \cup A$ and each set $Y \subseteq V$, we define:

$$\mathit{maxscore}(X, Y) = \max_{c \in X \setminus \{p\}} \left(\mathit{score}^P_{(X,Y)}(c)\right)$$

In other words, $\mathit{maxscore}(X, Y)$ is the highest possible plurality score of a candidate from X, but other than p, in election $E = (X, Y)$.

Next, we define a linear order L over the set of candidates A: For each pair a, b of candidates from A, we have $a\ L\ b$ if and only if the following holds (by $\succ_{\mathit{last}(a)}$ we mean the preference order of voter $\hat{v}_{\mathit{last}(a)}$):

$$(\mathit{last}(a) < \mathit{last}(b)) \vee (\mathit{last}(a) = \mathit{last}(b) \wedge a \succ_{\mathit{last}(a)} b).$$

[2]Strictly speaking, candidates which cannot score any points might still be needed. Consider a case where A contains a single candidate who is ranked last by all the voters, p is the current winner, and we ask if there is a set A' of size exactly one, such that p is a winner in $(C \cup A', V)$. We would have to take $A' = A$ and that would mean taking the candidate for whom *first* and *last* are not defined. However, with the way we use Theorem 4.3.3 in the paper, such a situation is never relevant. Handling this possibility is easy, but we omit it for the sake of clarity and brevity.

So the order L lines up unregistered candidates by the index of the last voter in their *visibility interval*. In case two *visibility intervals* for two candidates end with the same voter, the candidate that is preferred by this voter takes precedence.

For a candidate $c \in A$, let $PreA(c)$ be the subset of A containing c and all the candidates that precede c under L. For each integer u and each candidate $c \in A$, we define $f(u, c)$ as follows:

$$f(u,c) = \min_{\substack{A' \subseteq PreA(c) \\ \|A'\| = u}} \{maxscore(C \cup A', V)\}$$

In other words, $f(u, c)$ is the minimal possible score that the highest-scoring candidate (not counting p) can have after adding exactly u candidates from $PreA(c)$ to election (C, V).

Observation 2. *Let a_{last} be the last candidate from A under L. Since no candidate from A can "steal" points from candidate p (this follows from our assumption on the input data), p can be made the unique winner of the election by adding exactly k candidates from A if and only if $f(k, a_{last}) < score^{P}_{(C,V)}(p)$.*

The above is true because $PreA(a_{last}) = A$, so the value of $f(k, a_{last})$ describes the minimal possible score of the best candidate other than p in election $(C \cup A', V)$ across all possible sets $A' \subseteq A$ of size k. Therefore if that score is lower than the score of p in (C, V), then p will be the winner of $(C \cup A', V)$ (by assumption the score of p cannot change after adding candidates).

Example. *To illustrate the above definitions, let us consider an instance of Plurality-CCAC with the set of registered candidates $C = \{x, p\}$, where p is the designated candidate, with the set of unregistered candidates $A = \{c, d, e\}$, and with the voter set $V = \{v_1, \ldots, v_9\}$:*

$$v_1 \colon e \succ d \succ c \succ x \succ p, \quad v_2 \colon d \succ c \succ x \succ p \succ e, \quad v_3 \colon c \succ x \succ d \succ p \succ e,$$
$$v_4 \colon c \succ x \succ d \succ p \succ e, \quad v_5 \colon x \succ c \succ d \succ p \succ e, \quad v_6 \colon x \succ c \succ d \succ p \succ e,$$
$$v_7 \colon p \succ x \succ c \succ d \succ e, \quad v_8 \colon p \succ x \succ c \succ d \succ e, \quad v_9 \colon p \succ x \succ c \succ d \succ e.$$

We note that $E = (C \cup A, V)$ is single-crossing and that $\hat{V} = (v_1, ..., v_9)$ is a single crossing order of the voters that witnesses this fact. Further, one can verify that sets C, A, and V satisfy all the constraints of Theorem 4.3.3 (in particular, adding the

candidates from the set $\{c, d, e\}$ to election (C, V) cannot affect the score of p). Let $k = 2$ be the number of candidates that we need to add.

The first$(\cdot)$ and last$(\cdot)$ values of the unregistered candidates are as follows:

$$\begin{aligned} first(c) &= 1, \quad last(c) = 4, \\ first(d) &= 1, \quad last(d) = 2, \\ first(e) &= 1, \quad last(e) = 1. \end{aligned}$$

Thus the order L is e L d L c and we have that $PreA(e) = \{e\}$, $PreA(d) = \{e, d\}$ and $PreA(c) = \{e, d, c\} = A$.

Let us calculate the value $f(2, d)$. By definition, it is the highest score of a candidate other than p in election $(C \cup \{e, d\}, V)$ (note that there is only one size-two subset in $PreA(d)$). This score is four (it is the score of candidate x in election $(C \cup \{e, d\}, V)$). To calculate the value $f(2, c)$, we need to consider three two-element subsets of $PreA(c)$, namely, $\{e, d\}$, $\{e, c\}$ and $\{d, c\}$. We already know that we have $maxscore(C \cup \{e, d\}, V) = 4$. Routine calculations show that we also have:

$$maxscore(C \cup \{c, d\}, V) = 2 \qquad and \qquad maxscore(C \cup \{c, e\}, V) = 3.$$

In effect, we have $f(2, c) = 2$. We note that it is possible to ensure p's victory by adding candidates c and d (because the score of p is greater than two).

For each voter $v \in V$, let $PreV(v)$ be the set of voters that includes v and all the voters that precede v (with respect to the single crossing order $\hat{V}$). For each integer u, each candidate $c \in A$, and each voter $v \in V$, let $\mathcal{L}(u, c, v)$ be the family of subsets of candidates such that for each $X \in \mathcal{L}(u, c, v)$ it holds that: (a) X has exactly u elements, (b) $c \in X$, (c) $X \subseteq PreA(c)$, and (d) candidate c scores a point from voter v in election $(C \cup X, V)$. For a given integer u, a candidate $c \in A$, and a voter $v \in V$, we define a helper function g as follows:

$$g(u, c, v) = \min_{A' \in \mathcal{L}(u,c,v)} \{maxscore(C \cup A', PreV(v))\}$$

For a given candidate $c \in A$ and a given integer u, the expression for $g(u, c, \hat{v}_{last(c)})$ is very similar to that for $f(u, c)$. The only differences are that it considers subsets of $PreA(c)$ that contain c, and that *maxscore* is computed on a limited set of voters (only voter v and all the voters that precede v under $\hat{V}$).

We now show how function f can be expressed using function g. For each candidate $c \in A$, by $PreV(c)$ we mean $PreV(\hat{v}_{last(c)})$, that is, $PreV(c)$ contains all the voters from $\hat{V}$ starting from the first one and up to the last one from the *visibility interval* of c. To express f using g correctly, we need to reconcile the differences between the definitions of f and g mentioned above. As g considers only subsets of $PreA(c)$ of size u that contain c (as opposed to f which does not have the limitation of c being a part of the subset), we scan through all possible candidates $c' \in PreA(c)$ in order to calculate f. The second difference is that f considers all the voters from V in order to compute *maxscore*, while g takes into account only voters up to and including v. We note that neither of the remaining voters (that is, neither of the voters that follow v under $\hat{V}$) can rank candidates from $PreA(c)$ as their top choices (otherwise the *visibility intervals* of these candidates would extend beyond v). Taking this into account, we express f using g as follows:

$$f(u,c) = \min_{c' \in PreA(c)} \left\{ \max(g(u,c',\hat{v}_{last(c')}), maxscore(C, V \setminus PreV(c'))) \right\} \tag{4.1}$$

This means that it suffices to be able to compute g in polynomial time to be able to also compute f in polynomial time.

Example. *Let us go back to Example 4.3.1. Naturally, we have:*

$$PreV(v_1) = \{v_1\}, \quad PreV(v_2) = \{v_1, v_2\}, \quad PreV(v_3) = \{v_1, v_2, v_3\}, \quad \ldots$$

We also note that $\mathcal{L}(2, d, v_2) = \{\{e, d\}\}$ *and that* $\mathcal{L}(2, d, v')$ *is empty for all other voters. Similarly,* $\mathcal{L}(2, c, v_3) = \mathcal{L}(2, c, v_4) = \{\{e, c\}, \{d, c\}\}$*, and* $\mathcal{L}(2, c, v')$ *is empty for all remaining voters* v'*. Thus for* $u = 2$ *it only makes sense to consider values of* g *for candidate* d *and voter* v_2 *or for candidate* c *and voters* v_3 *or* v_4*.*

Clearly, $g(2, d, v_2) = 1$ *as the only set* A' *to consider is* $\{e, d\}$*. In this case, voter* v_1 *gives his or her point to* e *and voter* v_2 *gives his or her point to* d*. Similarly, we have* $g(2, c, v_3) = 2$*. In this case we need to consider election among voters* v_1*,* v_2*, and* v_3 *either with added candidates* $\{e, c\}$ *or with added candidates* $\{d, c\}$*. In either case, there is a candidate (who is not* p*) that receives exactly two points. Finally, one can verify that* $g(2, c, v_4) = 2$ *(this time by considering the first four voters and the same two sets of candidates to add).*

We now compare values $f(2, d) = 4$ *and* $f(2, c) = 2$ *calculated in Example 4.3.1 with values that follow from Equation* (4.1)*. According to Equation* (4.1)*, we have*

that:

$$f(2,d) = \max\big(g(2,d,v_2),\ maxscore(C, V\ PreV(d))\big) = \max(1,4) = 4.$$

Calculating $f(2,c)$ requires a bit more effort, but still can be handled easily and efficiently:

$$\begin{aligned} f(2,c) &= \min\Big(\max\big(g(2,c,v_4),\ maxscore(C, V\ PreV(c))\big), \\ &\qquad \max\big(g(2,d,v_2),\ maxscore(C, V\ PreV(d))\big)\Big) \\ &= \min(\max(2,2), \max(1,4)) = 2. \end{aligned}$$

As expected, in both cases we get the same results as in Example 4.3.1.

We now describe a recursive method for computing g (we present this method as Algorithm 1, though the algorithm references the conditions we list below, so the reader might want to consider the algorithm and the description here in parallel). For each integer u, candidate $c \in A$, and voter $v \in V$, in order to compute $g(u,c,v)$ we scan through candidates $c' \in PreA(c)$, such that c scores a point from v in election $E = (C \cup \{c,c'\}, V)$, compute certain values (see below), and eventually return the smallest one of them. Let us fix $c \in A$ and $v \in V$. For each possible $c' \in PreA(c)$, such that c scores a point from v in $E = (C \cup \{c,c'\}, V)$ we consider the following cases:

Condition 1 ($last(c') < first(c)$). In this case the visibility intervals of c and c' do not overlap and we return the maximum of $g(u-1, c', last(c'))$ and $maxscore(C \cup \{c\}, PreV(c) \setminus PreV(c'))$.

Condition 2 ($first(c') > first(c)$). In this case the visibility interval of c' is within the visibility interval of c. Since V is single-crossing, we know that if both c and c' take part in the election, candidate c' will score no points. We skip this case for now and take it into account later.

Condition 3 ($last(c') \geq first(c)$). In this case the visibility intervals of c and c' overlap. We return the maximum of $g(u-1, c', t_{c',c})$ and $maxscore(C \cup \{c\}, PreV(c) \setminus PreV(t_{c',c}))$, where $t_{c',c}$ is such that all the voters $\hat{v}_1, \ldots, \hat{v}_{t_{c',c}}$ prefer c' to c, and the remaining ones prefer c to c' (recall Definition 2.1.5).

Going back to Condition 2, we know that for each c' that satisfies this condition, adding c' has no impact on the scores of the other candidates as long as c is taking

part in the election (the visibility interval of c covers the visibility interval of c'). Thus, whether we add such a candidate to the election or not, has no impact on the value of g, except that we might want to add such candidates to ensure that we add exactly u candidates and not any less. Algorithm 1 shows how to recursively compute g referring to all the possible cases described above. All border cases are covered and we take Condition 2 into account in the following way: We compute the number t of candidates that satisfy it and allow the recursive calls to g to use between $u-1$ and $u-1-t$ candidates (modeling the fact that we can add up to t candidates satisfying Condition 2). The algorithm returns ∞ if no subset $A' \subseteq A$ satisfying the definition of g exists.

Algorithm 1 Compute $g(u, c, v)$ recursively

if c does not score point from v in $E = (C \cup \{c\}, V)$ **then**
  return ∞
else if $u = 1$ **then**
  return $maxscore(C \cup \{c\}, PreV(v))$
end if
$result \leftarrow \infty$
$t \leftarrow$ the number of candidates $c' \in PreA(c) \setminus \{c\}$ that satisfy Condition 2
for $c' \in PreA(c) \setminus \{c\}$ **do**
  if c scores a point from v in $E = (C \cup \{c, c'\}, V)$ **then**
    $r' \leftarrow \infty$
    if Condition 1 is satisfied **then**
      $r' \leftarrow \min_{t' \in \{0,\ldots,t\}} \{\max(g(u-1-t', c', last(c'));$
        $maxscore(C \cup \{c\}, PreV(c) \setminus PreV(c'))\}$
    else if Condition 3 is satisfied **then**
      $r' \leftarrow \min_{t' \in \{0,\ldots,t\}} \{\max(g(u-1-t', c', t_{c',c});$
        $maxscore(C \cup \{c\}, PreV(c) \setminus PreV(t_{c',c}))\}$
    end if
    $result \leftarrow \min(result; r')$
  end if
end for
return $result$

Given the recursive formulation, expressed as Algorithm 1, we use standard dynamic programming techniques to compute g in polynomial time (strictly speaking, for each nonnegative integer $u < \|A\|$, each candidate $c \in A$ and each voter $v \in V$, $g(u, c, v)$ is computable in time $\mathrm{O}(n^2\|A\|^3)$). Using Equation (4.1), we compute $f(u, i)$ for each nonnegative $u \leq \|A\|$ and for each $c \in A$, in time $\mathrm{O}(n^2\|A\|^4)$. As per Observation 2, being able to compute f suffices to solve our problem. □

The next theorem demonstrates that we can use the previous algorithm and adapt it to the case where there exists alternatives that, when added, can reduce the score of the designated alternative.

Theorem 4.3.4. *A variant of Plurality-CCAC for the case of single-crossing elections, where it is required to add exactly a given number of candidates, is in* P.

Proof. We show a Turing reduction of our current problem to the one from Theorem 4.3.3 (in other words, we show how we can solve our current problem using the algorithm from Theorem 4.3.3 as a subroutine). Let $I = (C, A, V, p, k)$ be our input instance. Let $\hat{V} = (\hat{v}_1, ..., \hat{v}_n)$ be a single-crossing order of voters from V. For each set $H \subseteq C \cup A$ and each set $G \subseteq A$ such that $G \cap H = \emptyset$, let $J(H, G) = (H, G, V, p, k - \|H \cap A\|)$ be an instance of the problem in the format for Theorem 4.3.3 (though, to use Theorem 4.3.3, we would have to make sure the set of candidates G does not contain a candidate that would affect the score of p after being added to the election).

For each pair of candidates $a, b \in A$ (we allow $a = b$), let $G_{a,b}$ be a subset of A created by removing from set A candidates a, b and all candidates that can reduce the score of p (in the sense from Definition 4.3.1) in election $(C \cup \{a, b\}, V)$. Now we claim that for $k > 2$, I has a solution if and only if there exist $a, b \in A$, such that $J(C \cup \{a, b\}, G_{a,b})$ has a solution. The right-to-left implication is trivial, so let us focus on proving the other direction.

Let us assume that instance I has a solution and let A' be the set of candidates added in order to solve I. Let L be a subset of candidates from A' which are preferred to p by $\hat{v}_1$. We consider the case in which both L and $A' \setminus L$ are nonempty sets. We will later show how to generalize the following reasoning to the case where one of those sets is empty. Now let x be a candidate from L with maximum value of $t_{x,p}$ (recall Definition 2.1.5). If more than one candidate reaches the maximum value, let us select the one that is most preferred over all the others by the voter at index $t_{x,p}$ in $\hat{V}$. Similarly, let us select candidate y from $A' \setminus L$ with minimum value of $t_{p,y}$ (we break ties by selecting the one that is most preferred by the voter at index $t_{p,y}$ in $\hat{V}$). From Lemma 4.3.2, we know that the set of voters who rank p first in $(C \cup A', V)$ is a contiguous subsequence of $\hat{V}$. Clearly, this subsequence is located between indices $t_{x,p}$ and $t_{p,y}$. Thus it is easy to see that no candidate from $A' \setminus \{x, y\}$ can reduce the score of p in $(C \cup \{x, y\}, V)$. Therefore, clearly, set A' is a subset of $G_{x,y} \cup \{x, y\}$, and since $\|A' \setminus \{x, y\}\| = k - 2$, it is easy to see that by adding candidates from $A' \setminus \{x, y\}$ we can solve instance $J(C \cup \{x, y\}, G_{x,y})$. It

should be obvious now, that in case when either L or $A' \setminus L$ is empty, we can solve instances $J(C \cup \{x\}, G_{x,x})$ or $J(C \cup \{y\}, G_{y,y})$ respectively.

The case where $k \in \{0, 1\}$ is trivial since then we can solve our problem by brute-force. Therefore, in order to determine whether I has a solution we need to verify whether the solution exists for at least one out of fewer than $\|A\|^2$ instances of the problem from Theorem 4.3.3. Since the described reduction to the problem from Theorem 4.3.3 is polynomial-time computable and this problem is in P, we conclude our problem is in P as well. □

We now know that a variant of Plurality-CCAC where we ask if a fixed number of alternatives can be added in order to make p the winner, is polynomial-time solvable for single-crossing election. This turns out to be a very powerful result as it allows us to reduce to it Pluraily-CCAC and Plurality-CCDC.

Theorem 4.3.5. *Plurality-CCAC and Plurality-CCDC for single-crossing elections are both in* P*.*

Proof. We Turing-reduce Plurality-CCAC and Plurality-CCDC for single-crossing elections to the problem from Theorem 4.3.4.

Let $I_{CCAC} = (C, A, V, p, k)$ be an instance of Plurality-CCAC problem. That is, we are given a set of candidates C, a set of unregistered candidates A, a voter collection V, a designated candidate $p \in C$, and a nonnegative integer k—the upper bound for the number of unregistered candidates that can be added to the election. In addition, we know that $E = (C \cup A, V)$ is single-crossing. The reduction proceeds as follows. For each nonnegative integer s let $J_s = (C, A, V, p, s)$ be an instance of the problem from Theorem 4.3.4. It is easy to see that there exists a solution for I_{CCAC} if and only if for some $s \in \{0, ..., k\}$ there exists a solution for J_s. Clearly, the reduction runs in polynomial-time and, thus, Plurality-CCAC for single-crossing elections is in P.

Let $I_{CCDC} = (C, V, p, k)$ be an instance of Plurality-CCDC problem, where election $E = (C, V)$ is single-crossing. For each nonnegative integer s, let $J'_s = (\{p\}, C \setminus \{p\}, V, p, s)$ be an instance of the problem from Theorem 4.3.4. It is easy to see that solution for I_{CCDC} exists if and only if there exists a solution of J'_s for some $s \in \{m-1-k, ..., m-1\}$. (In other words, it exists in some instance J'_s, where we have to "bring back" all the candidate except up to k of them.) The reduction clearly runs in polynomial-time and, so, Plurality-CCDC for single-crossing election is in P. □

4.3.2 Destructive Control

We now move to the destructive control cases under plurality rule. Again, both adding and deleting candidate cases are in P under single-crossing electorates. To prove that, we could follow the same methodology as in the constructive control section and appropriately tweak previously shown algorithms. However, for the case of control by adding candidates we managed to find a much more elegant proof. As our dedicated proof is not only cleaner but also produces a much faster algorithm, we decided to present it. The algorithm in essence is that we try all subsets of unregistered alternatives of size not greater than three, and, if destructive control is possible, we expect the designated candidate not to win when at least one of such subsets is considered. The lemma below proves that testing subsets of size not greater than three suffices.

Lemma 4.3.6. *If destructive control by adding candidates is possible in single-crossing election under plurality, then it can be achieved by adding at most three candidates from the set of unregistered ones.*

Proof. We are given a set of candidates C, a set of unregistered candidates A, a collection of voters V, and a designated candidate p. We know that election $E = (C \cup A, V)$ is single-crossing.

Assume that there exists $A' \subseteq A$, $\|A'\| \geq 4$, such that in election $(C \cup A', V)$ candidate p is not a unique winner under plurality rule. Let $\hat{V} = (v_1, ..., v_n)$ be the single-crossing order of voters from V in election (C, V). From Lemma 4.3.2, we know that voters that rank p first in (C, V) represent a consecutive subsequence of $\hat{V}$ and we refer to that sequence as $\hat{V}^p$ from now on. Another observation is that if some candidate $a \in A'$ reduces the score of p in (C, V), then the set of voters that rank a first in $(C \cup \{a\}, V)$ but that rank p first in (C, V) is a consecutive subsequence of $\hat{V}^p$ and is either a prefix of $\hat{V}^p$ or a suffix of $\hat{V}^p$. This is so because, due to Lemma 4.3.2, the set of voters that rank p first cannot be split into two consecutive subsequences. A corollary of this is the following observation.

Observation 3. *For every three distinct candidates $a, b, c \in A'$ that each reduce the score of p in (C, V), there exists one candidate $x \in \{a, b, c\}$ that does not reduce the score of p in $(C \cup \{a, b, c\}, V)$.*

(For example, if both a and b reduced the score of p in $(C \cup \{a, b, c\}, V)$, then it would have to be the case that one of them were ranked first by some prefix of the voters in $\hat{V}^p$, whereas the other one were ranked first by some suffix of these voters.

However, if c also reduced the score of p in $(C \cup \{a, b, c\}, V)$ then either some prefix or some suffix of the voters from $\hat{V}^p$ would have to rank c first, which would be impossible.)

We now show how to reduce the size of the unregistered candidate set A' by removing a single candidate from it, without making p a unique winner. Let us select any four distinct candidates $a, b, c, d \in A'$ and consider the three following cases:

1. At least two candidates from $\{a, b, c, d\}$ do not reduce the score of p in (C, V). Let $x \in \{a, b, c, d\}$ be one of those candidates with the lowest score in $E = (C \cup A', V)$ (we break ties in an arbitrary fashion). We know that x is not a unique winner of $(C \cup A')$. Since x does not reduce the score of p in (C, V), the score of p will remain the same after we remove x from the election E (the scores of other candidates may increase). Therefore in election $(C \cup A \setminus \{x\})$ candidate p is not a unique winner.

2. Exactly three candidates from $\{a, b, c, d\}$ reduce the score of p in (C, V). Let us name those three candidates x, y, z and let us name the remaining 4th candidate r. Based on Observation 3, in group x, y, z, there is one candidate g that does not reduce the score of p in $(C \cup \{x, y, z\}, V)$. If g has lower score than r in $E = (C \cup A', V)$ we see that g can be safely removed from E without making p a unique winner. Otherwise we can remove r, retaining the same characteristics.

3. All candidates from $\{a, b, c, d\}$ reduce the score of p in (C, V). Based on Observation 3, we can now select two candidates $x, y \in \{a, b, c, d\}$ that do not reduce the score of p in $(C \cup \{a, b, c, d\}, V)$. Similarly as in the case above, we can remove the one with lower score in $(C \cup A, V)$, thereby not improving the score of candidate p and not removing the previous best-scoring candidate.

If our assumption regarding the existence of A' is true, then we can use the above-described technique to find a candidate $x \in A'$ such that in election $(C \cup A' \setminus \{x\}, V)$ candidate p is not a unique winner. This way, by removing candidates one by one, eventually we can produce a set A'' such that $\|A''\| = 3$ and p is not a winner of $(C \cup A'', V)$. This proves the lemma. □

Theorem 4.3.7. *Plurality-DCAC for single-crossing elections is in* P.

Proof. We are given a set of candidates C, a set of unregistered candidates A, a collection of voters V, a designated candidate p and an integer k, the upper bound

for the number of voters from A that can be added to the election. We are given that $E = (C \cup A, V)$ is single-crossing. From Lemma 4.3.6 follows that if destructive control by adding candidates is possible in E, then it can be achieved by adding at most three candidates. Therefore we can scan through all the subsets of A of size at most $\min(3, k)$ and check whether adding some given subset of candidates render that p is not a unique winner. Since there are no more than $\|A\|^3$ such subsets, we can find a solution of Plurality-DCAC in polynomial time. □

Unfortunately the above approach cannot be adapted to other variants of control problems. Thankfully the methodology we described in the constructive control section can be applied more broadly and below we describe how we apply it to solve the destructive control by deleting candidates variant (we need to make some modifications to compensate for the fact we use the unique-winner model and that the designated candidate cannot be deleted).

Theorem 4.3.8. *Plurality-DCDC for single-crossing elections is in* P.

Proof. We are given a set of candidates C, a collection of voters V such that $E = (C, V)$ is single-crossing, and a designated candidate p. We show a Turing reduction from the modified Plurality-CCDC problem for single-crossing elections, where (a) the designated candidate p wins when no other candidate has higher score, and (b) where there is another designated candidates, d, that cannot be deleted. The modified version of Plurality-CCDC can be easily proved to be in P by minor modifications in the proofs from Theorems 4.3.4 and 4.3.5. Let $I = (C, V, p, k)$ be an instance of Plurality-DCDC for single-crossing election. For each candidate $c \in C$, let $J_c = (C, V, c, p, k)$ be an instance of the modified Plurality-CCDC for single-crossing (p is the candidate that cannot be deleted). It is easy to see that I has a solution if and only if there exists a solution of J_c for some $c \in C \setminus \{p\}$. Thus the reduction is correct and runs in polynomial-time. □

4.4 Condorcet Elections

In this section we present our results for the Condorcet voting rule. Under this rule, constructive control by adding and deleting voters is hard in the unrestricted case, while the remaining control problems we discuss are polynomial-time solvable (see the results of Bartholdi [8, 47] and Table 4.1 for reference). However, when considering the single-crossing domain, we show that even Condorcet-CCAV and Condorcet-CCDV become computationally tractable.

Our proofs are based on the median voter theorem presented by Rothstein [75] and later discussed in the context of single-crossing electorates by Saporiti and Tohmé [76].

Lemma 4.4.1. *Let $E = (C, V)$ be a single-crossing election and let $\hat{V} = (\hat{v}_1, ..., \hat{v}_n)$ be a single-crossing order of the voters from V. Candidate $p \in C$ is a Condorcet winner if and only if it is the most preferred candidate by the median voter ($\hat{v}_{\lceil n/2 \rceil}$) or both median voters in case n is even (that is, both $\hat{v}_{n/2}$ and $\hat{v}_{n/2+1}$).*

Proof. Let c_P be the the most preferred candidate by the median voter or both median voters when n is even. Let $C_L = \{c \in C \mid c \succ_{\hat{v}_1} c_P\}$ and $C_R = \{c \in C \mid c_P \succ_{\hat{v}_1} c\}$. From the definition of single-crossingness, it is easy to see that for each candidate $c_R \in C_R$, c_P is preferred over c_R by all the voters from $\{\hat{v}_1, ..., \hat{v}_{\lceil n/2 \rceil}\}$. Similarly, for each candidate $c_L \in C_L$, c_P is preferred over c_L by all the voters from $\{\hat{v}_{\lceil n/2 \rceil}, ..., \hat{v}_n\}$. Since, clearly, $C_L \cup C_R = V \setminus \{c_P\}$ we see that c_P is preferred by more than half of the voters over each candidate from $V \setminus \{c_P\}$ and thus is a Condorcet winner, which ends the proof for the left-to-right direction.

Correctness of the right-to-left direction for the case when n is odd comes directly from the above as in that case median voter always exists. For the case when n is even, it is sufficient to note that when each of the two median voters prefers a different candidate then Condorcet winner does not exist at all. □

The results for both Condorcet-CCAV and Condorcet-CCDV under single-crossing follow a greedy approach that we prove to be correct using the above lemma.

Theorem 4.4.2. *Condorcet-CCDV and Condorcet-CCAV for single-crossing elections are in* P.

Proof. Proofs for the cases of adding and deleting voters are very similar. To solve CCDV and CCAV we simply try to delete/add voters in a correct place considering the single-crossing order of voters until the median voter ranks p first. Below we formalize those steps for both CCDV and CCAV.

For Condorcet-CCDV we are given a set of candidates C, a set of voters V of size n, a designated candidate $p \in C$, and an integer k, the upper bound on the number of voters that we can remove. By assumption, we know that $E = (C, V)$ is single-crossing and we are given a single-crossing order $\hat{V} = (\hat{v}_1, ..., \hat{v}_n)$ of the voters.

We define V_P to be the set of voters from $\hat{V}$ that rank p first. Note that voters from V_P form a contiguous block within $\hat{V}$. Let V_L be the set of voters that precede

the voters from V_P (under $\hat{V}$) and, similarly, let V_R be the set of voters that follow the voters from V_P. We now focus on finding a way for a voter from V_P to become the median voter (or a way for two voters from V_P to become the two median voters). If $V_P = \emptyset$ then p can never be a Condorcet winner, so let us assume that this is not the case. If we have $\|V_L\| < n/2$ and $\|V_R\| < n/2$ then no voter needs to be deleted and p is a Condorcet winner. In all the remaining cases, at most one of those two sets has more than $n/2$ voters. Let us assume without loss of generality that $\|V_L\| > \|V_R\|$. It suffices to keep on deleting voters from V_L until either p becomes a Condorcet winner or we delete more voters than we are allowed. It is easy to see that we can calculate sets V_P, V_L and V_R in polynomial time and, thus, Condorcet-CCDV is in P.

Let us now consider the adding voters case. As an additional part of the input, we are given a set W of size n' of unregistered voters. We know that election $(C, V \cup W)$ is single-crossing. Let $\hat{V} = \{\hat{v}_1, ..., \hat{v}_{n+n'}\}$ be a single-crossing order of the voters in this election. We use the same definition for sets V_P, V_L and V_R as in the deleting-voters case. In addition we define W_P, W_L and W_R as subsets of W in a similar manner: W_P is the set of voters from W that rank p first, W_L is the set of voters from W that precede those from $V_p \cup W_p$ (under $\hat{V}$), and W_R is the set of voters from W that follow those from $V_P \cup W_P$. Based on Lemma 4.4.1, we can express the Condorcet-CCAV problem as that of finding a set $W' \subseteq W$ of size at most k, such that the median voter from the set $V \cup W'$ (under $\hat{V}$) is from $V_P \cup (W_P \cap W')$ (or, that both the median voters in $V \cup W'$ are from $V_P \cup (W_P \cap W')$, if $\|V\| + \|W'\|$ is even). It is easy to see that we can always start by adding all the voters from W_P (at most k of them). So, without loss of generality, we may assume W_P to be empty (if it is not empty, we add at most k elements from W_P to V_P and reduce k by the number of voters we have added). With that assumption in mind, we reject if V_p is empty (it is impossible for p to become a Condorcet winner). Then we verify if the median voter (or, both median voters if the number of voters in the election is even) is from the set V_P. If so, then p already is a Condorcet winner and we don't need to add any voters. Otherwise we will be adding voters from either W_L or W_R. Without loss of generality, we assume that $\|V_L\| > \|V_R\|$. We keep on adding voters from V_R until either p becomes the Condorcet winner or we cannot add any more voters. Polynomial running time of the algorithm is straightforward to verify. □

4.5 Approval Voting

Originally, the notion of single-crossingness was defined for the ordinal preference model only. Therefore the definition of Mirrlees [62] and Roberts [74] cannot be directly applied to approval voting. However, Elkind and Lackner [28] provide a definition of a *voter-interval* restriction (VI) that in a way can be treated as an adoption of the concept of single-crossingness for the approval elections.

Definition 4.5.1 (Elkind and Lackner [28])**.** *Let $E = (A, N)$ be an approval election and let $\hat{N} = (\hat{v}_1, ..., \hat{v}_n)$ be an ordered sequence of the agents from N. We say that $\hat{N}$ is a* voter-interval *order of agents N in election E if for every candidate c it holds that the agents that approve of c form an interval with respect to $\hat{N}$ (i.e., there are numbers $f(c)$ and $e(c)$ such that exactly agents $\hat{v}_{f(c)}, \hat{v}_{f(c)+1}, \ldots, \hat{v}_{e(c)}$ approve of c).*

Definition 4.5.2. *An approval election $E = (A, N)$ has the* voter interval *property if there is a voter-interval order of N for E.*

Intuitively speaking, for an election with voter-interval property, the set of agents that approves of each alternative is a consecutive subsequence of a voter-interval order formed sequence of agents. It can also be illustrated in a similar way to how we explained single-crossing notion in Section 4.2. If we say that the voter-interval orders agents starting from a "left-wing" ones towards more "right-wing" ones, then the most left-wing alternative is going to be approved by the first agent in the order. When we start moving towards more right-wing agents at some point one such agent will disapprove this alternative. In such a case all further agents disapprove it as well.

Despite voter-interval definition capturing the natural notion of single-crossingness, it has its downsides too. For example, when considering an ordinal election that is single-crossing, we would expect a derived approval election (e.g., made by making each agent approve of some number of their top choices on their ordinal scale) to have the voter-interval property. Let us take a look at the following example to see that this is not always the case.

Example. *Consider an election $E = (A, N)$ with an alternatives set $A = \{a, b, c\}$ and three agents with the following preferences:*

$$a \succ_1 b \succ_1 c$$
$$a \succ_2 c \succ_2 b$$
$$c \succ_3 b \succ_3 a$$

Clearly, E is single-crossing with respect to the presented ordering of agents. Now, using votes from E we derive an approval election E' by making each alternative approve their top two choices. That is, agent 1 approves alternatives a and b, agent 2 approves a and c, and agent 3 approves c and b. The resulting approval election does not have the voter-interval property. This is true because each of these agents disapproves of a different alternative while it approves all the remaining ones. Therefore regardless of how we order the three agents the middle one will always disapprove an alternative that both surrounding agents approves.

The discussed limitation however does not diminish the value of voter-interval property from our perspective. It allows us to study the complexity of control problems under approval voting and compare the results with the unrestricted case which, in essence, is all we are looking for.

With the voter-interval property in hand we move on to the studied control problems results. We start with the destructive case as the main idea of this algorithm can be adapted to cover the constructive case as well.

Theorem 4.5.3. *Approval-CCDV for voter-interval elections is in* P.

Proof. Consider an instance of Approval-CCDV with candidate set C, voter set V, bound k on the number of voters that can be deleted, and preferred candidate p. We assume that election $E = (C, V)$ has the voter-interval property and we let $\hat{V} = (\hat{v_1}, \ldots, \hat{v_n})$ be a voter-interval order of the voters from V. We assume that p is approved of by at least one voter. Otherwise we can output the negative solution for our problem immediately.

Clearly, it never makes sense to delete voters that approve of p. For each candidate $c \in C \setminus \{p\}$, we define $\gamma(c)$ to be $score_E^A(c) - score_E^A(p)$, that is, the difference between the score of c and the score of p in the original election. Let $\widetilde{C}$ be the set of candidates $c \in C \setminus \{p\}$ for whom $\gamma(c) \geq 0$.

Let V' be the subset of voters from V who do not approve of p, and let $n' = \|V'\|$. Let $(v'_1, \ldots, v'_{n'})$ be a voter-interval order of the voters from V' (it is easy to see that this order exists). For each candidate $c \in \widetilde{C}$, let $f(c)$ and $e(c)$ be two numbers such that in V' exactly voters $v'_{f(c)}, v'_{f(c)+1}, \ldots, v'_{e(c)}$ approve of p. (Technically, these numbers may not be well-defined. However, this happens only for those candidates in $\widetilde{C}$ of whom no voter in V' approves. Such candidates are approved of exactly by the same voters as p and, thus, if they exist it is impossible to ensure that p is a unique winner of the election through deleting voters).

Our algorithm proceeds by executing the following (greedy) operation until either we ensure that p is a winner or exceed the number of voters that we can delete. First, we find a candidate $c \in \widetilde{C}$ for whom $e(c)$ is minimal (we break ties arbitrarily). We check if $e(c) - \gamma(c) + 1 \geq f(c)$. If so, we delete the $\gamma(c)$ voters $v'_{e(c)}, v'_{e(c)-1}, \ldots, v'_{e(c)-\gamma(c)+1}$. If not, we reject (it is impossible to delete sufficiently many voters that approve of c). Finally, we recalculate V', the values γ, and $\widetilde{C}$ (taking into account the fact that we deleted some voters). If $\widetilde{C}$ is not empty, then we repeat the above procedure. If it is, we check if we deleted at most k voters. If so, we accept. Otherwise, we reject.

Example. *We illustrate the algorithm on the following election with candidate set* $C = \{p, c_1, c_2, c_3, c_4, c_5\}$ *and with nine voters* $v_1, \ldots, v_9$ *(below for each voter we indicate with a '+' which candidates he or she approves of):*

	p	c_1	c_2	c_3	c_4	c_5
v_1						+
v_2			+			+
v_3		+	+			+
v_4		+	+			
v_5	+	+	+			
v_6	+	+	+		+	
v_7	+			+	+	
v_8				+	+	
v_9				+		

We see that the election has the voter-interval property for the natural ordering of the voters. The approval score of p *is three, whereas every other candidate has score three or higher. Since voters* v_5, v_6 *and* v_7 *approve* p*, we never delete either of them. For all the candidates (not counting* p*), candidate* c_5 *has the smallest* $e(\cdot)$ *value and so we first delete voter* v_3*. This ensures that* c_5 *has score two (lower than* p*). The next candidates to consider (in the order of the lowest* $e(\cdot)$ *value among the candidates whose score matches or exceeds that of* p*) are* c_1 *and* c_2*. We can choose either of them and we pick* c_1*. In effect, we delete voter* v_4*. Then we have to consider* c_2 *and delete voter* v_2*. Finally, due to candidate* c_4*, we need to delete voter* v_8 *(which also takes care of candidate* c_3*). All in all, we see that to ensure that* p *is the unique winner, we need to delete voters* v_1, v_2, v_3, v_4 *and* v_8*. One can verify that, indeed, this is an optimal solution.*

To see why this algorithm is correct, it suffices to show the correctness of the first greedy step performed in the algorithm. Let V', values γ, and $\widetilde{C}$ be as just before deleting the first voter, and let c be the candidate chosen by the algorithm in the first iteration. We have to delete $\gamma(c)$ voters that approve of c, that is, some $\gamma(c)$ voters among $v'_{f(c)}, \ldots, v'_{e(c)}$. For each two numbers i, j such that $f(c) \leq i < j \leq e(c)$ it holds that the set of candidates approved by v'_i is equal to or is a subset of the candidates approved by v'_j. (This is due to the fact that the election is voter-interval and due to the choice of c.) Thus deleting voters $v'_{e(c)}, v'_{e(c)-1}, \ldots, v'_{e(c)-\gamma(c)+1}$ is optimal. After executing one iteration of our algorithm we face a problem of the same type and we proceed iteratively. This completes the proof. □

The constructive control case can also be solved in polynomial-time. In fact, we reduce constructive control by adding voters to a variant of destructive control problem with a specified number of voters to be removed and that allows for only some certain voters to be deleted. We do that by first adding all the voters from W that approves of p, and then delete sufficiently many voters that we have just added from the election constructed that way. The greedy algorithm we provided for the destructive case can be easily adapted to solve this modified variant of the problem.

Theorem 4.5.4. *Approval-CCAV for voter-interval elections is in* P.

Proof. Consider an instance of Approval-CCAV with candidate set C, voter set V, set of unregistered voters W, bound k on the number of voters that we can add, and preferred candidate p. We assume that election $(C, V \cup W)$ has the voter-interval property. Further, without loss of generality, we assume that every voter in W approves of p (otherwise we could remove this voter from W because it is never beneficial to add a voter who does not approve of p).

We form election $E = (C, V \cup W)$. If p is not a unique winner in this election then, clearly, it is impossible to ensure his or her victory (this is because every voter in W approves of p). We compute number $k' = \max(\|W\| - k, 0)$. Intuitively, k' is the number of W-voters that we need to delete from E to make sure that it can be obtained from (C, V) by adding k voters from W (since every voter in W approves of p, we can focus on the case where we add exactly k voters).

After we delete k' of the W-voters from E, the score of p will decrease to $score^A_E(p) - k'$. Thus, we have to delete voters in such a way that afterward every candidate in $C \setminus \{p\}$ has lower score. For each candidate $c \in C \setminus \{p\}$ we define $\gamma(c) = score^A_E(c) - (score^A_E(p) - k') + 1$. Intuitively, for each $c \in C \setminus \{p\}$, $\gamma(c)$ is the number of W-voters that approve of c, that we need to delete.

At this point, our problem is, in essence, the same as in the case of control by deleting voters. We consider an election $E' = (C, W)$, which has the voter-interval property, we have to delete exactly k' voters, and we have to make sure that for each $c \in C \setminus \{p\}$ we delete at least $\gamma(c)$ voters that approve of c. Thus we use the same algorithm as in the proof of Theorem 4.5.3 above. (Note that, technically, the algorithm from the above theorem might delete fewer than k' voters to ensure the constraints regarding the γ values; in this case we pick further voters to delete arbitrarily).

The correctness of the algorithm and the running time follow directly from its construction. □

4.6 Conclusions

We presented polynomial-time algorithms for the election control problems under single-crossing profiles for Plurality, Condorcet and Approval rules. Our results align with the ones achieved by Faliszewski et al. [37] for the single-peaked domain. The results for both the single-peaked and single-crossing domains are in contrast to the unrestricted cases, where most of the control problems are intractable [8]. This further shows that the worst case scenarios on which unrestricted case NP-hardness results rely are not likely to appear in real life scenarios.

All of that raises the question of what type of constrains do voting profiles need to meet in order for the control problems to be intractable? A natural direction is to try to relax the restrictions put by either single-crossing or single-peaked domains. As shown by Faliszewski et al. [35] when considering nearly single-peaked preferences, for some election manipulation problems even having a single agent with a vote that does not fit into the single-peaked profile can render the whole problem NP-hard. It is tempting to try how results analogous to those for nearly single-peaked electorates would look like for the case of single-crossing ones. Finally, considering top-monotonic profiles (recall Chapter 3) could also yield very interesting results. Top-monotonicity is a larger class that captures both single-peaked and single-crossing profiles and therefore puts forward milder requirements on the electorate.

Chapter 5

Counting Variants of Control Problems

5.1 Introduction

In this chapter we discuss counting variants of various election control problems in the context of winner prediction. In winner prediction, our goal is to get a sense of what are the probabilities of each agent winning the election. Naturally, there are many factors that may have an impact on the election. For example, we can be unsure of which agents will cast their vote in the election, or what the final lineup of the alternatives will be (e.g., some candidates may choose to withdraw before the election). On top of that, we can think of the agents' preferences as being subject to change. Because of such uncertainties, different researchers approach the area of winner prediction from different angles.

One of the first works done in this field has been focused on modelling imperfect knowledge regarding voters' preferences (see the ideas of Konczak and Lang [56], further developed, e.g., by Xia et al. [88, 89], Betzler and Dorn [10], Bachrach and Betlzer [5], and by Chevaleyre et al. [19]). In their work, the knowledge we have about each agent's preferences is limited to a partial order over the set of alternatives. Then, given such agents' partial preferences, we ask if these preferences can be extended to total orders such that a designated alternative wins the election. It has been shown later by Bachrach, Betzler, and Faliszewski [5] that we can model the probability of each alternative's victory by considering counting variant of this problem. Namely, they asked how many different total-order extensions exists such that the designated alternative wins. That, with the assumption that each total extension

is equally likely to happen in the election, allowed them to compute the probability of each alternative's victory (see also the follow-up work of Kenig et al. [54] and of Kimelfeld et al. [55]). Another paper that studies uncertain preferences is that of Hazon et al. [46]. They consider that each agent has a number of possible preference orders and each order is assigned a likelihood of it being selected by the agent. Given that, they ask what is the probability that a designated alternative wins the election.

In the previously discussed models, we always have full knowledge about the alternatives and the agents taking part in the election, even though we are unsure what votes they are going to cast. An alternative approach is to assume we are certain about agents' preferences, but we are not sure about the exact set of agents or alternatives that will participate in the election. Wojtas and Faliszewski [86] first examined this direction. They considered election control problems as a way to model uncertainties about the sets of agents or alternatives joining the election. As an example, let us consider a set of alternatives C and two disjoint sets of agents V and W. Let us now form an election from the available sets of alternatives and agents. We assume that for the election that we consider, we can be sure that all the alternatives from C and all the agents from V take part in that election. For each of the remaining agents from W we do not know if they are going to vote. If we assume that exactly k agents from W would vote, then calculating probability of some alternative p winning the election can be reduced to a counting variant of the constructive control problem by adding voters (CCAV). In the counting variant of CCAV we are given an election $E = (C, V)$, a set of unregistered agents W with preferences over C, a designated alternative $p \in C$, and a number k, $k \leq \|W\|$. We ask how many subsets $W' \subseteq W$ of size k exist such that in election $E' = (C, V \cup W')$ alternative p is the winner. To calculate the probability of p winning in such a scenario, we need to divide the number of combinations in which it wins by the number of subsets of W of size k (that is, by $\binom{\|W\|}{k}$)—we denote that probability as $Q_p(k)$. As presented by Wojtas and Faliszewski [86], this model can be easily extended to a scenario in which we have a probability distribution P on the number of voters from W that will participate in the election. That is, for each k, $0 \leq k \leq \|W\|$, let $P(k)$ denote the probability that exactly k agents from W cast their votes in the election. We can then express the probability that p wins as:

$$Win(p) = P(0)Q_p(0) + P(1)Q_p(1) + \cdots P(\|W\|)Q_p(\|W\|).$$

Computing victory likelihoods using the above formula for each alternative allows, for example, to find the most probable winner. Analogously, we can achieve probability results for other variants of control problems, namely for both constructive and destructive cases of adding and deleting voters or candidates.

The results presented by Wojtas and Faliszewski [86] show that for most of the counting variants of the considered control problems under Plurality, k-Approval, Approval, Condorcet and Maximin voting rules there are no polynomial-time algorithms (see also Table 5.1 part (a) that contains a summary of their results). In this chapter we extend this line of work by focusing on restricted domains. With the single-peaked restriction put on the agents' preferences, we provide polynomial-time algorithms for counting variants of all of the mentioned control problems under Plurality, k-Approval and Condorcet rules. The summary of our results is presented in Table 5.1 part (b).

Acknowledgments

The results discussed in this chapter have been presented in the technical report *Possible Winners in Noisy Elections* co-authored by Krzysztof Wojtas, Krzysztof Magiera, Tomasz Miąsko and Piotr Faliszewski [87]. This report extends an earlier conference paper, under the same title but due to Krzysztof Wojtas and Piotr Faliszewski [86], which itself is based on the Master's thesis of Krzysztof Wojtas [85]. My contribution to the results in this study consists of the polynomial-time algorithms for single-peaked elections under Plurality, k-Approval, and Condorcet voting rules (presented in this chapter). The proofs for k-Approval voter control counting variants for single-peaked electorates have also been independently developed and extended to control with unary prices by Tomasz Miąsko in his Master's thesis [64]. The reason for this was that at some point it was not clear if Tomasz would provide these results[1] and I was asked to work on them. However, eventually, Tomasz delivered them and, thus, the complicated status of this chapter.

5.2 Preliminaries

In this chapter we discuss a group of computational problem called *counting problems* which we define as variants of corresponding decision problems in the following way: Let A be some computational problem where for each instance I we ask if there exists

[1] Piotr Faliszewski lost contact with him for some time.

(a) The Unrestricted Case

Problem	Plurality	k-Approval, $k \geq 2$	Approval	Condorcet	Maximin
#CCAC	#P-com.	#P-com.	–	–	#P-com.
#DCAC	#P-metric-com.	#P-metric-com.	FP	FP	#P-metric-com.
#CCDC	#P-com.	#P-com.	FP	FP	#P-Turing-com.
#DCDC	#P-metric-com.	#P-metric-com.	–	–	#P-Turing-com.
#CCAV	FP	#P-Turing-com.	#P-com.	#P-com.	#P-com.
#DCAV	FP	#P-Turing-com.	#P-metric-com.	#P-metric-com.	#P-metric-com.
#CCDV	FP	#P-metric-com.	#P-com.	#P-com.	#P-com.
#DCDV	FP	#P-metric-com.	#P-metric-com.	#P-metric-com.	#P-metric-com.

(b) The Single-Peaked Case

Problem	Plurality	k-Approval, $k \geq 2$	Condorcet-Consistent (e.g., Maximin)
#CCAC	FP	FP	–
#DCAC	FP	FP	FP
#CCDC	FP	FP	FP
#DCDC	FP	FP	–
#CCAV	FP	FP	FP
#DCAV	FP	FP	FP
#CCDV	FP	FP	FP
#DCDV	FP	FP	FP

Table 5.1: The complexity of counting variants of control problems for (a) the unrestricted case, and for (b) the single-peaked case. A dash in an entry means that the given system is *immune* to the type of control in question (i.e., it is impossible to achieve the desired effect by the action this control problem allows; technically this means the answer to the counting question is always 0). Immunity results were established by Bartholdi, Tovey, and Trick (1989) [8] for the constructive cases, and by Hemaspaandra, Hemaspaandra, and Rothe (2007) [47] for the destructive cases.

some mathematical object satisfying a given condition. In the counting variant of A, denoted $\#A$, for each instance I we ask for the number—denoted $\#A(I)$—of the objects that satisfy the condition. To analyze computational complexity of counting problems we need to use somewhat different notions than the usual P and NP classes. The class of counting variants of NP-problems is called #P and the class of functions computable in polynomial time, which includes counting problems that can be solved in polynomial time, is called FP. Analogously to the decision problems, we also define hardness classes for the counting variants, but in the case of counting problems different reducibility notions are needed.

Definition 5.2.1. *Let $\#A$ and $\#B$ be two counting problems.*

1. *We say that $\#A$ Turing reduces to $\#B$ if there exists an algorithm that solves $\#A$ in polynomial time given oracle access to $\#B$ (i.e., given the ability to solve $\#B$ instances in unit time).*

2. *We say that $\#A$ metric reduces to $\#B$ if there exist two polynomial-time computable functions, f and g, such that for each instance I of $\#A$ it holds that (1) $f(I)$ is an instance of $\#B$, and (2) $\#A(I) = g(I, \#B(f(I)))$.*

3. *We say that $\#A$ parsimoniously reduces to $\#B$ if $\#A$ metric reduces to $\#B$ via functions f and g such that for each instance I and each integer k, $g(I, k) = k$ (i.e., there is a polynomial-time computable function f such that for each instance I of $\#A$, we have $\#A(I) = \#B(f(I))$).*

For the listed reducibility notions we define completeness notions. For a given reducibility notion R, we say that a problem is #P-R-complete if it belongs to #P and every #P-problem R reduces to it. In Table 5.1 we write #P-complete to denote #P-parsimonious-completeness. An example of a #P-parsimonious-complete problem is #X3C [48].

Definition 5.2.2. *For a given pair of integers k and n, a set $B = \{b_1, \ldots, b_{3k}\}$ and a family of 3-element subsets of B, $\mathcal{S} = \{S_1, \ldots, S_n\}$. In #X3C we ask how many k-element subsets of $\mathcal{S}$ have B as their union.*

A good representative of #P-Turing-hard problems is #PerfectMatching [83]:

Definition 5.2.3. *In #PerfectMatching we are given a bipartite graph $G = (G(X), G(Y), G(E))$ with $\|G(X)\| = \|G(Y)\|$ and we ask how many perfect matchings does G have.*

In this chapter we present FP-membership results only, but we need the above definitions to compare our results to those for the unrestricted case, presented by Wojtas and Faliszewski [86].

5.3 Counting Variants of Election Control Problems

Below we formally define counting variants of the election control problems that we discuss in this chapter. Specifically, we are interested in control by adding candidates (AC), control by deleting candidates (DC), control by adding voters (AV), and control by deleting voters (DV). For each of these problems, we consider its constructive variant (CC) and its destructive variant (DC).

Definition 5.3.1 (Faliszewski and Wojtas [86])**.** *Let R be a voting system. In each of the counting variants of constructive control problems we are given a candidate set C, a voter collection V, a nonnegative integer k, and a designated candidate $p \in C$. In constructive control by adding voters we are additionally given a collection W of unregistered voters, and in constructive control by adding candidates we are additionally given a set A of unregistered candidates. In these problems we ask for the following quantities:*

1. *In control by adding voters (R-#CCAV), we ask how many sets W', $W' \subseteq W$, are there such that p is the unique winner of R-election $(C, V \cup W')$, where $\|W'\| \leq k$.*

2. *In control by deleting voters (R-#CCDV), we ask how many sets V', $V' \subseteq V$ are there such that p is the unique winner of R-election $(C, V - V')$, where $\|V'\| \leq k$.*

3. *In control by adding candidates (R-#CCAC), we ask how many sets A', $A' \subseteq A$, are there such that p is the unique winner of R-election $(C \cup A', V)$, where $\|A'\| \leq k$.*

4. *In control by deleting candidates (R-#CCDC), we ask how many sets C', $C' \subseteq C$, are there such that p is the unique winner of R-election $(C - C', V)$, where $\|C'\| \leq k$ and $p \notin C'$.*

Destructive variants are defined identically, except that we ask for the number of settings where the designated candidate is not the unique winner.

Different variants of election control problems are related to each other. This is best illustrated with the following two theorems originally presented by Wojtas and Faliszewski [86]. We rely on these connections to get the results for the complete spectrum of all our control problems. In short, the theorems below show links between the complexity of destructive and constructive variants of control problems, and links between the complexity of the "deleting" and "adding" variants.

Theorem 5.3.2 (Faliszewski and Wojtas [86])**.** *Let R be a voting system, let $\#\mathcal{C}$ be one of R-$\#$CCAC, R-$\#$CCDC, R-$\#$CCAV, R-$\#$CCDV, and let $\#\mathcal{D}$ be the destructive variant of $\#\mathcal{C}$. Then, $\#\mathcal{C}$ metric reduces to $\#\mathcal{D}$ and $\#\mathcal{D}$ metric reduces to $\#\mathcal{C}$.*

Theorem 5.3.3 (Faliszewski and Wojtas [86])**.** *Let R be a voting system, then R-$\#$CCDV Turing reduces to R-$\#$CCAV and R-$\#$CCDC Turing reduces to R-$\#$CCAC.*

The above two theorems are frequently used throughout this chapter as they allow us to directly obtain easiness results for destructive cases from the constructive ones (Theorem 5.3.2) and for the "deleting" cases from the "adding" ones (Theorem 5.3.3).

5.4 Plurality

Under Plurality rule, counting variants of both control by adding and by deleting voters are in FP even for the unrestricted case [87]. In consequence we obtain the following corollary for the single-peaked case.

Corollary 5.4.1. *For the case of single-peaked profiles Plurality-$\#$CCAV, Plurality-$\#$DCAV, Plurality-$\#$CCDV, and Plurality-$\#$DCDV are in* FP*.*

When considering the problems of control by adding and deleting candidates we rely on the results for k-Approval.

Theorem 5.4.2. *For the case of single-peaked profiles, Plurality-$\#$CCAC, Plurality-$\#$CCDC, Plurality-$\#$DCAC, and Plurality-$\#$DCDC are in* FP*.*

Proof. This result follows from Theorem 5.5.1 (see the next section) for the case of 1-Approval. □

5.5 k-Approval Voting

With k-Approval rule, all the variants of control problems are intractable in the unrestricted case [87]. Yet, if we restrict the profile domain to be single-peaked, it turns out that all of these problems can be solved in polynomial-time. Note that we rely on k being a constant and that our results do not generalize to the case where k is part of the input.

We start with the candidate control cases for which we give dynamic-programming-based algoriths. Our approach is inspired by the algorithm for the single-peaked variant of Plurality-CCAC of Faliszewski et al. [37]

Theorem 5.5.1. *For the case of single-peaked profiles, for each fixed positive k, k-Approval-#CCAC, k-Approval-#CCDC, k-Approval-#DCAC, and k-Approval-#DCDC are in* FP.

Proof. We give the proof for k-Approval-#CCAC only. The result for k-Approval-#CCDC follows by applying Theorem 5.3.3, and the destructive cases follow by applying Theorem 5.3.2.

We are given a candidate set C, a voter collection V, a designated candidate $p \in C$, a collection A of unregistered candidates, a positive integer ℓ, and a societal axis L over $C \cup A$. We show a polynomial time algorithm for determining the number of sets A', $A' \subseteq A$, where $\|A'\| \leq \ell$, such that p is a k-Approval winner in election $(C \cup A', V)$.

Our algorithm is based on dynamic programming. Intuitively, the idea is as follows: First, we fix two groups of k candidates, one group "to the left" of p with respect to L and the other group "to the right" of p (we try all reasonable choices of candidates for these groups and sum up the results; see details below). We will show that each such choice of the "neighborhood" of p fixes p's score. Thus, the choice of p's neighborhood reduces our task to counting the ways to add the candidates "to the left" of the neighborhood, so that the candidates to the left of p have scores smaller than p, and, independently, counting the candidates "to the right" of the neighborhood, so that the candidates "to the right" of p also have scores smaller than p. In the text below, function f will be responsible for the case of "adding candidates to the left," and function f' will regard the case of "adding candidates to the right." We now move on to the formal description of our algorithm.

For a set of candidates $X \subseteq C \cup A$ and a candidate $c \in X$, let $\text{rank}_X(c)$ be the rank of candidate c under L among the candidates from X, starting from 1 (i.e.,

$\text{rank}_X(c) = \|\{d \mid d \in X \wedge d \; L \; c\}\| + 1)$. For each $X, Y \subseteq C \cup A$, we define a partial order L' over $2^{C \cup A}$, such that $X \; L' \; Y$ if and only if $\forall_{x \in X} \forall_{y \in Y} x \; L \; y$. For a given set $X \subseteq C \cup A$, let U_X be the largest possible subset of $C \cup A$ such that $U_X \; L' \; X$. For a set $X \subseteq C \cup A$ and a candidate $c \in C \cup A$, let $V_{X,c}$ be the set of voters from V, such that for each $v \in V_{X,c}$ there are at least k candidates in X which v prefers over c. Let $\phi_X(c) = \|V_{X,c}\|$. Clearly, for each $A' \subseteq A$ and each candidate $c \in C \cup A'$, the number of points c scores in k-Approval election $(C \cup A', V)$ is $\|V\| - \phi_{C \cup A'}(c)$.

For a set $X \subseteq C \cup A$, $\|X\| \leq k$, an $\|X\|$-dimensional vector β of integers, an integer s, and a nonnegative integer z, we define the following function (intuitively, one can think of X here as the "left part of the neighborhood of p"):

$$f(X, \beta, z, s) = \begin{cases} \text{If } s \geq \|X \cap A\| \text{ and } \|X\| = k, \text{ then it is the number of sets } A', \\ A' \subseteq A \cap U_X, \text{ where } \|A'\| = s - \|X \cap A\|, \text{ such that in } k\text{-} \\ \text{Approval election } (((C \cup A') \cap U_X) \cup X, V), \text{ each candidate} \\ c \in (C \cup A') \cap U_X \text{ scores no more than } z \text{ points and each} \\ \text{candidate } c \in X \text{ scores no more than } \beta_{\text{rank}_X(c)} \text{ points.} \\ \\ \text{It is 1 if } s = \|X \cap A\|, \; \|X\| < k, \text{ and in } k\text{-Approval election} \\ (X, V) \text{ each candidate } c \in X \text{ scores no more than } \beta_{\text{rank}_X(c)} \\ \text{points.} \\ \\ \text{It is 0 otherwise.} \end{cases}$$

We repeat some of the steps from the above paragraph in order to define another function, f', a symmetric variant of f. For a given $X \subseteq C \cup A$, let U'_X be the largest subset of $C \cup A$ such that $X \; L' \; U'_X$. Now for a set $X \subseteq C \cup A$, $\|X\| = k$, a vector β of integers of size $\|X\|$, an integer s, and a nonnegative integer z we define (intuitively, one can think of X here as the "right part of the neighborhood of p"):

$$f'(X, \beta, z, s) = \begin{cases} \text{If } s \geq \|X \cap A\| \text{ and } \|X\| = k, \text{ then it is the number of sets } A', \\ A' \subseteq A \cap U'_X, \text{ where } \|A'\| = s - \|X \cap A\|, \text{ such that in } k\text{-} \\ \text{Approval election } (((C \cup A') \cap U'_X) \cup X, V), \text{ each candidate} \\ c \in (C \cup A') \cap U'_X \text{ scores no more than } z \text{ points and each} \\ \text{candidate } c \in X \text{ scores no more than } \beta_{\text{rank}_X(c)} \text{ points.} \\ \\ \text{It is 1 if } s = \|X \cap A\|, \; \|X\| < k, \text{ and in } k\text{-Approval election} \\ (X, V) \text{ each candidate } c \in X \text{ scores no more than } \beta_{\text{rank}_X(c)} \\ \text{points} \\ \\ \text{It is 0 otherwise.} \end{cases}$$

The next lemma is the basis of our idea of introducing two groups of candidates around p (the neighborhood of p) to fix p's score.

Lemma 5.5.2. *For each set $H \subseteq C \cup A$, $\|H\| = k$, each set $X \subseteq U_H$, and each candidate $c \in X$, if z is the score of candidate c in k-Approval election $(X \cup H, V)$, then for each $Y \subseteq U'_H$ in k-Approval election $(X \cup H \cup Y, V)$ the score of candidate c is z as well.*

Proof. For a given subset $Y \subseteq U'_H$ and candidate $c \in X$, suppose that score of candidate c in k-Approval election $(X \cup H \cup Y, V)$ is $z' \neq z$. Adding a candidate can never increase the score of any other candidate already present, so we assume that $z' < z$. It means that there exists a voter $v \in V$ from which c gains a point in election $E_1 = (X \cup H, V)$ but not in election $E_2 = (X \cup H \cup Y, V)$. Let $c' \in Y$ be a candidate that "steals" a point from c (i.e., in E_2 c' receives a point from V). Since c is among v's top k most preferred candidates across $X \cup H$ and $\|H\| = k$, there must exists a candidate $c'' \in H$ that is less preferred than c by v. We see that c' is preferred over c by v, so it is also perferred over c'' by v. Therefore both c and c' are preferred over c'' by v, but on the other hand we know that $c \; L \; c'' \; L \; c'$, because $c \in X$, $c'' \in H$ and $c' \in Y$. This is in contradiction with the single-peakedness of the voters' preferences. □

We are almost ready to describe our algorithm, but we need one more component. We define K_L to be a collection of subsets of $U_{\{p\}}$ with the following properties (intuitively, K_L is the family of sets of candidates that can form the "left part" of p's neighborhood):

1. if $Y \in K_L$, then $\|Y\| \leq k$, and if $\|Y\| < k$ then Y contains all elements from $U_{\{p\}} \cap C$.

2. if $Y \in K_L$, then for each candidate c from $U_{\{p\}} \cap C$ it holds that either $c \in Y$ or for each $c' \in Y$, $c \; L \; c'$.

We define K_R analogously, but "to the right of p". That is, K_R is a collection of subsets of U'_p with the following properties:

1. if $Y \in K_R$, then $\|Y\| \leq k$, and if $\|Y\| < k$ then Y contains all elements from $U'_{\{p\}} \cap C$.

2. if $Y \in K_R$, then for each candidate c from $U'_{\{p\}} \cap C$ it holds that either $c \in Y$ or for each $c' \in Y$, $c' \; L \; c$.

Thus, if we pick some set P from K_L and some set F from K_R, then $P \cup \{p\} \cup F$ is a set of up to $2k+1$ candidates that form the neighborhood of p (with set P directly preceding and set F following the candidate p). By the properties of K_L and K_R, for each candidate $c \in C - (P \cup \{p\} \cup F)$ it holds that either for each $d \in P \cup \{p\} \cup F$ we have $c \; L \; d$ or for each $d \in P \cup \{p\} \cup F$ we have $d \; L \; c$. By Lemma 5.5.2 we have that the score of p in election $(P \cup \{p\} \cup F, V)$ is the same as in every election that in addition contains some candidates "to the left" of P and "to the right" of F. Based on this observation, for each $P \in K_L$ and $F \in K_R$ we define $\rho_{P,F}$ to be the score of p in election $(P \cup \{p\} \cup F, V)$.

Let us now consider the situation "to the left" of p, after we have decided to fix some $P \in K_L$ and $F \in K_R$. Further, let a be some integer such that we are willing to add exactly a candidates from $A \cap U_A$ to our election. How many sets P_L, $P_L \subseteq A \cap U_P$ of cardinality exactly a are there such that in election $(C \cup P_L \cup P \cup \{p\} \cup F, V)$ the candidates "to the left" of p have scores smaller than $\rho_{P,F}$? For each pair of sets $X, Y \subseteq C \cup A$ and an integer z, we define vector $\lambda(z, X, Y)$ of size $\|X\|$, such that for each $c \in X$ we have $\lambda(z, X, Y)_{\mathrm{rank}_X(c)} = z + \phi_Y(c)$. Now, the answer to our question is:

$$f(P, \lambda(\rho_{P,F} - 1, P, F \cup \{p\}), \rho_{P,F} - 1, a).$$

The correctness of our claim follows by the definition of f, but there is one point that deserves further clarification: Why do we use vector $\lambda(\rho_{P,F} - 1, P, F \cup \{p\})$ as the bound for the scores of candidates in P? The reason is that function f is not "aware" that we are adding candidates from F, and so it counts the scores of candidates in P including those voters, who would give points to candidates from P provided candidates from F were not present; using vector $\lambda(\rho_{P,F} - 1, P, F \cup \{p\})$ allows f to count these points in, knowing that in the end they will not go to the candidates from P.

It is easy to see that we can also carry out a similar analysis as above for the case of candidates "to the right" of p.

To obtain the final result for our input instance of k-Approval-#CCAC, we need to try all combinations of sets $P \in K_L$, $F \in K_R$, and all partitions of the number ℓ of candidates between candidates added "to the left" of p and candidates added "to the right" of p. Since the results for the left part and the right part are independent, in the end we get the following formula:

$$\sum_{\substack{a,b\in\mathbb{N}\\ a+b\leq\ell}}\sum_{P\in K_L}\sum_{F\in K_R}\Big(f(P,\lambda(\rho_{P,F}-1,P,F\cup\{p\}),\rho_{P,F}-1,a)\\ f'(F,\lambda(\rho_{P,F}-1,F,P\cup\{p\}),\rho_{P,F}-1,b)\Big) \tag{5.1}$$

Using Eq. (5.1), we can compute the result of k-Approval-#CCAC problem in polynomial time, provided that the required values of f and f' can also be calculated in polynomial time. This follows from the fact that both $\|K_L\|$ and $\|K_R\|$ can be bound by $k\|C\cup A\|^k$ (and, thus, we can bound the number of components we need to sum up in Eq. (5.1) by $\ell^2k^2\|C\cup A\|^{2k}$).

Now we show how to compute the values of f and f' in polynomial time. We focus on function f (the case of f' is symmetrical). For a pair of sets $X,Y\subseteq C\cup A$ and a vector of integers β of size $\|X\|$, let $\delta(X,Y,\beta)=1$ when in k-Approval election $(X\cup Y,V)$ each candidate $c\in X$ scores no more than $\beta_{\mathrm{rank}_X(c)}$ points, and $\delta(X,Y,\beta)=0$ otherwise. Further, for each set X of candidates we define K_X to be a family of subsets of candidates defined in the same way as K_L, but where the "leftmost" (with respect to L) candidate of X takes the role of p. We have the following recursive formula for f:

$$f(X,\beta,z,s)=\begin{cases}0 & \text{if}\quad s<\|X\cap A\|\\ \delta(X,X,\beta) & \text{if}\quad \|X\|<k\\ \sum_{Y\in K_X}\delta(X,Y,\beta)f(Y,\lambda(z,Y,X),z,s-\|X\cap A\|) & \text{otherwise}\end{cases} \tag{5.2}$$

Eq. (5.2) shows a recursive formula for f. Note that we can bound the number of possible β vectors by $\|V\|^k$; when some entry of β is greater than $\|V\|$, we can replace it with $\|V\|$ because that is the highest possible score in k-Approval election with voter collection V. In the same way we can bound parameter z, so that we need to consider only $z\in\{0,...,\|V\|\}$. Finally, we can bound the number of possible sets X by $k\|C\cup A\|^k$. Using dynamic programming, we can, thus, compute values of f in polynomial time $\mathrm{O}\left(\|A\|\|V\|^{k+2}k^2\|C\cup A\|^{2k}\right)$. □

Next, we present a polynomial-time algorithm for k-Approval-#CCAV that also is based on on dynamic programming. The results for the remaining control problems follow by Theorems 5.3.2 and 5.3.3.

Theorem 5.5.3. *For the case of single-peaked profiles, for each fixed positive integer k, k-Approval-#CCAV, k-Approval-#CCDV, k-Approval-#DCAV and k-Approval-#DCDV are in* FP.

Proof. We give the proof for k-Approval-#CCAV. The result for k-Approval-#CCDV follows by Theorem 5.3.3 and the results for the destructive cases follow by Theorem 5.3.2.

We are given a candidate set C, a voter collection V, a designated candidate $p \in C$, a collection W of unregistered voters, a positive integer ℓ, and a societal axis L. We show a polynomial time algorithm for determining the number of sets W', $W' \subseteq W$, where $\|W'\| \leq \ell$, such that p is a k-Approval winner in election $(C, V \cup W')$.

For each voter $v \in V \cup W$, we define $\text{top}_k(v)$ to be the set of k most preferred candidates according to v. Let $\widetilde{L}'$ be a weak linear order over voter set $W \cup V$, such that $v_1 \ \widetilde{L}' \ v_2$, $v_1, v_2 \in V \cup W$, if and only if there exists a candidate $c \in \text{top}_k(v_2)$ such that for each $c' \in \text{top}_k(v_1)$ we have $c' \ L \ c$ or $c' = c$. Let L' be a strict order obtained from $\widetilde{L}'$ by breaking the ties in an arbitrary fashion. For each voter $v \in V \cup W$ let $X^v = \{w \mid w \in V \cup W \wedge w \ L' \ v\}$ be the subset of voters from $V \cup W$ that precede voter v under L'. We let v_l to be the last voter from V under L'. For a given voter $v \in V \cup W$, integer $z \in \mathbb{Z}$, and integer $s \in \mathbb{Z}$, we define $g(v, z, s)$ to be the number of sets $\widetilde{W} \subseteq X^v \cap W$, where $\|\widetilde{W}\| = s$, such that p is a k-Approval winner in election $(C, (X^v \cap V) \cup \widetilde{W} \cup \{v\})$ and in addition the score of candidate p in this election is exactly z. Equation (5.3) below gives the result that we should output on the given input instance of k-Approval-#CCAV problem:

$$\sum_{0 \leq s \leq \ell} \left(\sum_{0 \leq z \leq \|W \cup V\|} \left(g(v_l, z, s) + \sum_{\substack{w \in W \\ v_l L' w}} g(w, z, s) \right) \right) \tag{5.3}$$

Thus, it suffices to show how to compute $g(w, z, s)$ for $w \in V \cup W$ and $z, s \in \mathbb{Z}$ in polynomial time. Before we give an appropriate algorithm, we need to introduce some additional notation.

For each candidate $c \in C$, let $\text{rank}(c)$ be c's rank under L over all candidates from C (so, for example, if $C = \{a, b, c\}$ and $a \ L \ b \ L \ c$ then $\text{rank}(a) = 1$, $\text{rank}(b) = 2$ and $\text{rank}(c) = 3$). For each voter v, let $\text{rank}(v) = \max\{\text{rank}(c) \mid c \in \text{top}_k(v)\}$. For each $v \in V \cup W$, let $\mathcal{P}^v = \{w \mid w \in (V \cup W) \wedge w \ L' \ v \wedge (\nexists t \in V)[w \ L' \ t \ L' \ v]\}$. In other words, $\mathcal{P}^v$ consists of the closest voter $u \in V$ that precedes v under L', and of all the voters that are between u and v under L'. When v is the first element from V under L', then $\mathcal{P}^v$ contains all the voters from W preceeding v under L'.

For a j-element integer vector *scores* and an integer i, $1 \leq i \leq j$, by $scores_i$ we denote the i-th element of vector *scores* (i.e., $scores = (scores_1, \ldots, scores_j)$). Now, for a nonnegative integer r, $0 \leq r \leq j$, we let cutoff$(scores, r)$ denote a j-element vector $(scores_1, \ldots, scores_r, 0, \ldots, 0)$ (that is, we replace the last $j - r$ entries of vector *scores* with zeros). For each voter $w \in V \cup W$, we define approval(w) to be the $\|C\|$-dimensional 0/1 vector that for each candidate $c \in C$ has 1 at position rank(c) if and only if $c \in \text{top}_k(w)$. (In other words, approval(w) is the 0/1 vector that indicates which candidates receive points from voter w.) Note that, due to single-peakedness of the election, for each voter w, approval(w) contains exactly a single consecutive block of k ones.

We are now ready to proceed with our algorithm for computing function g. For a given voter $v \in V \cup W$, given integers $z, s \in \mathbb{Z}$, and a $\|C\|$-dimensional integer vector *scores*, we define $f(v, z, s, scores)$ to be the number of sets $\widetilde{W} \subseteq X^v \cap W$ such that $\|\widetilde{W}\| = s$ and in election $(C, (X^v \cap V) \cup \widetilde{W} \cup \{v\})$ the following holds: (1) p scores exactly z points and (2) each candidate $c \in C$ scores no more than $scores_{\text{rank}(c)}$ points. For a given integer $r \in \mathbb{Z}$, let Γ^r be the vector $(r, ..., r)$ of dimension $\|C\|$. Clearly, for a given voter $v \in V \cup W$ and given integers $z, s \in \mathbb{Z}$ we have:

$$g(v, z, s) = f(v, z, s, \Gamma^{z-1}) \tag{5.4}$$

We now show a recursive formula for f. For a given voter $v \in V \cup W$, a given vector *scores* of (nonnegative) integers of dimension $\|C\|$, and two integers $z, s \in \mathbb{Z}$, we have:

$$f(v, z, s, scores) = f(v, z, s, \text{cutoff}(scores, \text{rank}(v))) \tag{5.5}$$

This follows from the fact that in election $(C, X^v \cup \{v\})$ only candidates with ranks $j \leq \text{rank}(v)$ can score a point. It is easy to see that $f(v, z, s, scores) = 0$ when $z < 0$ or *scores* contains at least one negative entry, because the score is always a nonnegative integer. When $s = 0$ then $f(v, z, s, scores)$ is 1 if and only if in election $(C, (X^v \cap V) \cup \{v\})$ candidate p scores z points and each candidate $c \in C$ scores no more than $scores_{\text{rank}(c)}$ points; otherwise $f(v, z, s, scores)$ is 0. When $s > 0$, we note that each election consistent with the condition for $f(v, z, s, scores)$ contains at least one voter w from $\mathcal{P}^v$. Eq. (5.6) below gives formula for f in case when $s > 0$; for each voter $w \in \mathcal{P}^v$ we count all the sets $\widetilde{W} \subseteq X^v \cap W$ such that w directly precedes

v in $(X^v \cap V) \cup \widetilde{W} \cup \{v\}$ under L':

$$f(v, z, s, scores) = \sum_{w \in \mathcal{P}_v} f(w, z - z', s - s', scores - \text{approval}(w)), \tag{5.6}$$

where (1) $z' = 1$ if $p \in \text{top}_k(v)$ and $z' = 0$ otherwise, and (2) $s' = 1$ if $v \in W$ and $s' = 0$ otherwise. By combining Eq. (5.5) and (5.6) we get:

$$f(v, z, s, scores) = \sum_{w \in \mathcal{P}_v} f(w, z - z', s - s', \text{cutoff}(scores - \text{approval}(w), \text{rank}(w))). \tag{5.7}$$

We claim that function g can be computed through Eq. (5.4) in polynomial time, using standard dynamic programming techniques to compute function f. The reason is that to compute f for the arguments as in Eq. (5.4) using recursive formula (5.7), we need to obtain f's values for at most $\|C\|(\|V\| + \|W\|)^{k+3}$ different arguments. This is because starting from $scores = \Gamma^{z-1}$, the only allowed transformations of *scores* are given in Eq. (5.7) and ensure that whenever we need to compute f, the *scores* vector is of the form $(z-1, z-1, ..., z-1, e_1, e_2, ..., e_k, 0, 0, ..., 0)$, where $e = (e_1, e_2, ..., e_k)$ is some k-element vector of integers and for each i, $1 \leq i \leq k$, we have $0 \leq e_i \leq z - 1$. Clearly, there are no more than $\|C\|z^k$ vectors of this form. Thus, we can compute g in polynomial time and, through Eq. (5.3), we can solve k-Approval-#CCAV in polynomial time. □

5.6 Condorcet Voting

When it comes to candidate control, the results for the Condorcet rule in single-peaked domains are established by polynomial-time algorithms for the unrestricted case (with the exception for #CCAC and #DCDC for which it is impossible to achieve the desired effect by the actions allowed in these variants [8]). For the case of voter control in the unrestricted case, the counting variants are #P-hard [86]. Nonetheless, we have found polynomial-time algorithms for the case of single-peaked preferences. Just like in the previous section, we use dynamic programming.

Since under the single-peaked assumption, when the number of voters is odd, there always exists a Condorcet winner, in essence the results from this section apply to all Condorcet-consistent rules.

Theorem 5.6.1. *For the single-peaked case, Condorcet-#CCAV, Condorcet-#CCDV, Condorcet-#DCAV, and Condorcet-#DCDV are in* FP.

Proof. We focus on Condorcet-#CCAV. The remaining cases follow by applying Theorems 5.3.2 and 5.3.3.

We are given an election $E = (C, V)$ where C is a set of candidates and V is a set of voters, a designated candidate $p \in C$, a collection W of unregistered voters, a nonnegative integer k, and an order L, the societal axis, such that V and W are single-peaked with respect to L. We show a polynomial-time algorithm for determining the number of sets W', $W' \subseteq W$, where $\|W'\| \leq k$, such that p is a Condorcet winner in election $(C, V \cup W')$.

We split C into three sets, $C_a = \{c \mid c \in C \wedge c \; L \; p\}$, $C_b = \{c \mid c \in C \wedge p \; L \; c\}$ and $C_m = \{p\}$; C_a contains candidates that are before p on the societal axis and C_b contains candidates that are after p. For each voter v, by c_v we mean the candidate that v ranks first. We split V into three sets, $V_a = \{v \mid v \in V \wedge c_v \in C_a\}$, $V_b = \{v \mid v \in V \wedge c_v \in C_b\}$ and $V_m = \{v \mid v \in V \wedge c_v = p\}$. In other words, V_a and V_b consist of voters from V for which the most preferred candidate is, respectively, in C_a or C_b, and V_m contains those voters from V that rank p first. Similarly, we split W into three sets: $W_a = \{w \mid w \in W \wedge c_w \in C_a\}$, $W_b = \{w \mid w \in W \wedge c_w \in C_b\}$ and $W_m = \{w \mid w \in W \wedge c_w = p\}$.

For a given candidate $c \in C \setminus \{p\}$ and a subset of voters $\widetilde{V} \subseteq V \cup W$, let $\sigma(c, \widetilde{V})$ be the number of voters from $\widetilde{V}$ who prefer candidate c over p. Let $\delta(c, \widetilde{V}) = \sigma(c, \widetilde{V}) - (\|\widetilde{V}\| - \sigma(c, \widetilde{V}))$. That is, $\delta(c, \widetilde{V})$ is the difference between the number of voters from $\widetilde{V}$ who prefer c over p and the number of voters who prefer p over c. When for some candidate c we get $\delta(c, \widetilde{V}) > 0$, it means that more than half of the voters from $\widetilde{V}$ prefer p over c. Using this definition, p is a Condorcet winner of election $(C, \widetilde{V})$ if and only if $\delta(c, \widetilde{V}) > 0$ for each $c \in C \setminus \{p\}$.

For a given subset $\widetilde{C}$ of candidates, subset $\widetilde{W}$ of unregistered voters, integer ℓ, $0 \leq \ell \leq k$, and integer s, $-\|C\| \leq s \leq \|C\|$, let $g(\widetilde{C}, \widetilde{W}, \ell, s)$ be the number of ℓ-element sets $\widetilde{W'} \subseteq \widetilde{W}$, such that for each $c \in \widetilde{C} \setminus \{p\}$ we have $\delta(c, V \cup \widetilde{W'}) > s$.

We want to count the number of ℓ-element sets W', $W' \subseteq W$, such that p is a Condorcet winner in election $(C, V \cup W')$. We split ℓ into three components $\ell = \ell_a + \ell_b + \ell_m$, $0 \leq \ell_a, \ell_b, \ell_m$, each one corresponding to the number of elements from, respectively, W_a, W_b, and W_m, that are going to be used to build W'. Clearly, there are no more than ℓ^3 choices for ℓ_a, ℓ_b and ℓ_m. For some fixed triple ℓ_a, ℓ_b and ℓ_m, assume that we have some set W'_a, such that $W'_a \subseteq W_a$ and $\|W'_a\| = \ell_a$. For each

candidate $c \in C_a$, let $s_c = \delta(c, V \cup W'_a)$. Because of single-peakedness, all voters from $W_b \cup W_m$ prefer p to c, which means that $\delta(c, V \cup W') = s_c + \ell_b + \ell_m$. Since we want $\delta(c, V \cup W') > 0$ for each $c \in C \setminus \{p\}$, we need to ensure that $s_c + \ell_b + \ell_m > 0$ for each $c \in C_a$. Thus $g(C_a, W_a, \ell_a, -\ell_b - \ell_m)$ gives the number of subsets of ℓ_a-element voters from W_a which together with $\ell_b + \ell_m$ voters from $W_b \cup W_m$ we can add to election (C, V) in a way that p is a Condorcet winner. By symmetry, we can see that for given ℓ_a, ℓ_b and ℓ_m, the number of subsets $W' \subseteq W$ of size $\ell_a + \ell_b + \ell_m$ such that p is a Condorcet winner of election $(C, V \cup W')$ can be expressed by:

$$g(C_a, W_a, \ell_a, -\ell_b - \ell_m) g(C_b, W_b, \ell_b, -\ell_a - \ell_m) g(C_m, W_m, \ell_m, -\ell_a - \ell_b).$$

Since C_m contains only one element, the last factor is simply $\binom{\|W_m\|}{\ell_m}$. Below we give the final formula for #CCAV-Condorcet problem:

$$\sum_{\ell=0,\ldots,k} \sum_{\substack{\ell_a,\ell_b,\ell_m \in \mathbb{N} \\ \ell_a+\ell_b+\ell_m=\ell}} g(C_a, W_a, \ell_a, -\ell_b - \ell_m) g(C_b, W_b, \ell_b, -\ell_a - \ell_m) \binom{\|W_m\|}{\ell_m}. \quad (5.8)$$

Assuming we could compute g in polynomial time, we would be able to solve #CCAV-Condorcet in polynomial time using Equation (5.8). Below we show how to compute $g(C_a, W_a, \ell, s)$ for every $0 \leq \ell \leq k$ and $-\|C\| \leq s \leq \|C\|$.

For each voter $v \in W_a \cup V_a$, let $X_v \subseteq C_a$ be the subset of candidates that v ranks below p. Let $Y_v = C_a \setminus X_v$. Due to single-peakedness, we have $X_v \; L \; Y_v$. Clearly $Y_v \neq \emptyset$. For each voter $v \in W_a \cup V_a$, let c^*_v be the first element of L-ordered set Y_v, that is, let $c^*_v \; L \; c_y$ for every $c_y \in Y_v \setminus \{c^*_v\}$. Let us now define linear order L^a over the set of voters $W_a \cup V_a$. Let $v_1 \; L^a \; v_2$ for some $v_1, v_2 \in W_a \cup V_a$, iff $c^*_{v_1} \; L \; c^*_{v_2}$ (we break ties in L^a in some arbitrary fashion). Let $v^a_f, v^a_l \in W_a \cup V_a$ be, respectively, the first and the last element under L^a, which means that $v^a_f \; L^a \; v$ for every $v \in W_a \cup V_a \setminus \{v^a_f\}$ and $v \; L^a \; v^a_l$ for every $v \in W_a \cup V_a \setminus \{v^a_l\}$. For each $v \in W_a \cup V_a \setminus \{v^a_f\}$ we write $prev^a(v)$ to mean the voter directly preceding v under L^a. Let $c^a_l \in C_a$ be the last candidate from C_a under L, i.e., $c \; L \; c^a_l$ for every $c \in C_a \setminus \{c^a_l\}$. Let $U_v = \{w \mid w \in W_a \cup V_a \wedge wL^a v\}$ for some $v \in W_a \cup V_a$. For a given voter $v \in W_a \cup V_a$ and integer ℓ, let $\mathcal{P}_a(v, \ell)$ be the family of ℓ-element sets $U'_v \subseteq U_v \cup \{v\} \cap W$. For each integer s, $-\|C\| \leq s \leq \|C\|$, each integer ℓ, $0 \leq \ell \leq k$ and each voter $v \in W_a \cup V_a$, let $f_a(v, \ell, s)$ be the number of sets $U'_v \in \mathcal{P}_a(v, \ell)$ for which we have $\delta(c, V_a \cup U'_v) > s$ for every $c \in C_a$. It is easy to see that for a given

integer ℓ, $(0 \leq \ell \leq \|W_a\|)$ and integer s $(-\|C_a\| \leq s \leq \|C_a\|)$, it holds that:

$$g(C_a, W_a, \ell, s) = f_a(v_l^a, \ell, s - (\|V_b\| + \|V_m\|)). \tag{5.9}$$

In equation (5.9), every set $U'_v \in \mathcal{P}_a(v, \ell)$ that satisfies f_a corresponds to some set $\widetilde{W'} = U'_v$ that satifsies g. This is true because for each $c \in C_a$ and each $X \subseteq W_a$ we have $\delta(c, V \cup X) = \delta(c, V_a \cup X) + (\|V_b\| + \|V_m\|)$, which follows from the fact that every voter in $V_a \cup V_m$ prefers p to c.

To compute f_a, we note that every voter $v \in W_a \cup V_a$ prefers candidate c_l^a to p, which means that for each $H \subseteq W_a \cup V_a$, we have $\delta(c_l^a, H) = -\|H\|$. Additionally, for every $v \in W_a \cup V_a \setminus H$ we have:

$$\delta(c_l^a, H \cup \{v\}) = \delta(c_l^a, H) - 1. \tag{5.10}$$

Since $-\|H\|$ is the minimum possible value for $\delta(c_l^a, H)$, for every $c \in C_a$ we have $\delta(c_l^a, H) \leq \delta(c, H)$. Assume that $v_f^a \in V$. Then, we have $\delta(c_l^a, \{v_f^a\}) = -1$, so $f_a(v_f^a, \ell, s)$ is 1 for $s < -1$ and $\ell = 0$, and is 0 when $s \geq -1$. When $\ell > 0$ clearly $f_a(v_f^a, \ell, s) = 0$ for each possible s, since $\mathcal{P}_a(v_f^a, \ell)$ is empty. Now, assume that $v_f^a \in W$. We have that $f_a(v_f^a, \ell, s) = 1$ when $\ell = 0$ and $s < 0$ or when $\ell = 1$ and $s < -1$, since in that case $\mathcal{P}_a(v_f^a, 1)$ contains single set $\{v_f^a\}$. Similarily, for each other possible ℓ and s we have $f_a(v_f^a, \ell, s) = 0$. Assuming that $v \neq v_f^a$ and in addition $v \in V_a$, we have:

$$f_a(v, \ell, s) = f_a(prev^a(v), \ell, s + 1). \tag{5.11}$$

Equation (5.11) follows from Eq. (5.10) and the fact that if $v \in V_a$, then $\mathcal{P}_a(v, \ell) = \mathcal{P}_a(prev^a(v), \ell)$ for each nonnegative integer ℓ. In case where $v \in W_a$, we get:

$$f_a(v, \ell, s) = f_a(prev^a(v), \ell, s) + f_a(prev^a(v), \ell - 1, s - 1). \tag{5.12}$$

The first component of Eq. (5.12) gives us the number of subsets which v is not a part of. The second component corresponds to the number of subsets $U'_v \in \mathcal{P}_a(v, \ell)$ which contain v and satisfy definition of f_a. This follows from Eq. (5.10) and the observation that each of those subsets can be obtained from exactly one $(\ell - 1)$-element set from $\mathcal{P}_a(prev^a(v), \ell - 1)$. Eq. (5.13) shows a recursive formula for f_a

which combines all the considerations above:

$$f_a(v,\ell,s) = \begin{cases} f_a(prev^a(v),\ell,s+1) & \text{if} \quad v \in V_a \wedge v \neq v_f^a \\ f_a(prev^a(v),\ell,s) + f_a(prev^a(v),\ell-1,s+1) & \text{if} \quad v \in W_a \wedge v \neq v_f^a \\ 1 & \text{if} \quad v = v_f^a \wedge v \in V_a \wedge \\ & \qquad \ell = 0 \wedge s < -1 \\ 1 & \text{if} \quad v = v_f^a \wedge v \in W_a \wedge \\ & \qquad \ell = 0 \wedge s < 0 \\ 1 & \text{if} \quad v = v_f^a \wedge v \in W_a \wedge \\ & \qquad \ell = 1 \wedge s < -1 \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

With Eq. (5.13), function f_a can be computed in polynomial time using standard dynamic programming techinques. This means that we can also get the value of $g(C_a, W_a, \ell, s)$ in polynomial time (see Eq. (5.9)). Symmetrically, we can also compute $g(C_b, W_b, \ell, s)$ in polynomial time. In effect, we have that Condorcet-#CCAV is in FP. □

Finally, we note that the single-peaked easiness results translates to all Condorcet-consistent rules.

Corollary 5.6.2. *Let R be a Condrocet-consistent rule. For the single-peaked case, R-#CCDC, R-#DCAC, R-#CCAV, R-#CCDV, R-#DCAV, and R-#DCDV are in* FP.

The above corollary follows directly from the definition of Condorcet-consistent rules. Recall that a voting rule is Condorcet-consistent if it has the property that when there is a Condorcet winner then it is the winner under this rule. As in our work we focus on the unique winner model, we always require that only one winner exists. Therefore it is clear that if in a single-peaked election a Condorcet-consistent rule has a unique winner, it is also a Condorcet winner, and hence we can follow the same methodology as for Condorcet rule for all the counting variants of control problems.

for the case where there is an uncertainty regarding the exact set of agents or alternatives taking part in the election. Unlike with the decision variants of the election control problems, from a practical standpoint when considering counting variants we are more interested in efficient algorithms rather than worst-case complexity. Therefore, given that in the unrestricted case almost all of the counting variants of the problems that we consider are computationally hard [86], the algorithms we provide for the single-peaked case make the studied prediction model more viable for future research.

As efficient algorithms can be found for the presented model of winner prediction, it would be very interesting to see how this rather simple schema can be extended. For example, consider a model where instead of having a probability distribution over the number of agents participating in the election, for each agent v we have probability p_v of that agent casting their vote. Clearly, such a model is more complex than the one we considered (the case where probabilities for all the agents are the same reduces to our model), and so it seems unlikely that efficient algorithms can be found for the unrestricted case. Yet, we know that some election problems that are computationally hard under unrestricted domains can be solved in polynomial-time for the single-peaked case. Therefore, even when predicting the winners turns out to be hard for such a model, we may hope that efficient algorithms can be found for the single-peaked case.

Chapter 6

Control in Multiwinner Elections

So far in this thesis we only discussed election systems where the goal is to select a single winner. However, it is often the case that a single winner is not sufficient for our needs and we want to select a set of winners of a given size, commonly referred to as a *committee*. Parliamentary elections are the most notable example where such a system can be used, but it can also be applied in many other domains, e.g., when selecting representation among society members [57], for recommendation systems [66], or in automated collaborative planning [29]. The rules that output a winning set of a fixed size are called *multiwinner rules* or *committee voting rules*. Similarly to the single winner rules, they have been studied under the consideration of both ordinal voting (see, e.g., the works of Proccaccia et al. [73], Cornaz et al. [23], or Skowron et al. [80]) and dichotomous preferences (see, e.g., the work of Davis et al. [24], Aziz et al. [3] or Faliszewski et al. [41]). On top of that, many single winner voting rules have been adapted to the case of committee voting, starting from the approval rule with a corresponding *Approval Voting (AV)* multiwinner variant, *Condorcet winning set* [29] as an adaptation of Condorcet rule or a very popular STV rule, which can be considered in both single and multiwinner schemes. For a general overview of the topic of multiwinner voting, see the work of Faliszewski at al. [40].

Naturally, election control problems also can be considered in the context of committee elections. Yet, despite many similarities between single and multiwinner systems, this area has not been as broadly studied. Specifically, in the context of computational complexity there are only a handful of papers to date that tackle the issue of manipulating multiwinner elections. Some notable examples include the initial work of Meir et al. [61] who considered manipulation complexity for various multiwinner rules. Later, Aziz et al. [4] showed that control, manipulation and

bribery can be NP-hard for some approval-based multiwinner rules. More interesting results on control complexity was presented by Faliszewski et al. [41], who considered bribery as a measure of success and presented complexity results for adding, deleting and swapping approvals for various approval-based rules.[1]

In this chapter we continue this direction of research by considering both constructive and destructive variants of election control by deleting candidates and voters. We present complexity results for these control problems for *Approval Voting (AV)* and *Satisfaction Approval Voting (SAV)* rules. We stress that what we present here is only an initial set of results and studying election control in the multiwinner setting is an interesting research direction that can be further extended to considering restricted domains.

Acknowledgments

Results presented in this chapter have not been published in separate papers. However, as the process of finishing this thesis was delayed, they appeared independently in a paper of Yang [91]. We note that the proofs there are very different from ours.

6.1 Preliminaries

In an approval-based multiwinner election we are given a pair $E = (A, N)$, where A is a set of alternatives and N is a set of agents. For each agent x from N, let $App(x)$ be the set of alternatives that agent x approves. Multiwinner rules differ from single winner ones in that for a given election they output a winning set of some predefined size k, as opposed to a single winner (or no winner at all). Below we define rules that we use throughout this chapter.

Definition 6.1.1. *Under* Approval Voting (AV) *we are given an election $E = (A, N)$ and the size of the winning committee k. For each alternative $a \in A$ we define $score_E^{AV}(a)$ to be the number of agents that approve of a. The winning committee consist of k alternatives with the highest score.*

Definition 6.1.2. *Under* Satisfaction Approval Voting (SAV) *we are given an election $E = (A, N)$ and the size of the winning committee k. Each agent $x \in N$ contributes $\frac{1}{\|App(x)\|}$ points to each alternative it approves. For each alternative $a \in N$,*

[1] As the thesis faced some delays, currently there are many more relevant results, see, e.g., the works of Neveling et al. [65], Bredereck et el. [17], Gawron et al. [43], Misra et al. [63], or Yang [93]

let X_a be a set of agents who approve a. Then the score of alternative a is defined to be $score_E^{SAV}(a) = \sum_{x \in X_a} \frac{1}{\|App(x)\|}$. The winning committee consists of k alternatives with the highest score.

For the cases when more than k alternatives exist with a score that qualifies them to be included in the committee, a tie breaking rule needs to be adapted. For the purpose of this thesis we break ties based on the lexicographic order of alternatives and in addition exclude the designated candidate (later defined as p) from the winning committee in cases when it can be replaced by an alternative with an equal score.

Let us consider the following example to see how multiwinner rules work in practice.

Example. *We take a set of alternatives $A = \{a, b, c, d, e, f, g\}$ and set N of four agents, $N = \{1, 2, 3, 4\}$, that approve of the following alternatives:*

$$1\colon a, b, d, e \qquad 2\colon a, d, f \qquad 3\colon a, f, g \qquad 4\colon c$$

We now form a multiwinner election (A, N) with committee size $k = 3$. We see that under Approval Voting rule the winning committee consists of alternatives a, d and f. This is so because $score_E^{AV}(a) = 3$ and the scores of alternatives d and f are two. Each of the remaining alternatives is approved by one agent only and, therefore, its score is 1. However, if we consider the SAV rule, the winning committee consists of agents c, a and f. Alternative c scores highest under SAV rule as it is the only alternative approved by agent 4 and, hence, it has $score_E^{SAV}(c) = 1$. The second highest score belongs to alternative a who is approved by agents 2 and 3, along with two different alternatives, and by agent 1, among four alternatives. Therefore we have $score_E^{SAV}(a) = \frac{2}{3} + \frac{1}{4}$. Lastly, the score of the third highest-scoring alternative f is $\frac{2}{3}$. All the remaining alternatives score lower than f: We have alternative d with score of $\frac{1}{4} + \frac{1}{3}$, alternative g with score $\frac{1}{3}$, and both alternatives b and e which scores equal to $\frac{1}{4}$.

The election control problems for multiwinner rules are defined similarly to the single winner variants. The difference is that instead of asking if the designated candidate can become a winner (or not a winner, in the destructive variant) we ask if it can be made a part of the winning committee (or excluded from the committee, in the destructive variants).

Problem	CCDV	DCDV	CCDC	DCDC
Approval Voting (AV)	NP-hard	**NP-hard**	P	–
Satisfactory Approval Voting (SAV)	**NP-hard**	**NP-hard**	**NP-hard**	**NP-hard**

Table 6.1: The complexity of control problems for multiwinner rules. A dash in an entry means that the given system is *immune* to the type of control in question. Results marked using bold face are due to this thesis. The remaining results for Approval Voting are due to Meir et al. [61]. Note that these results were shown independently by Yang [91]

6.2 Results

In this section we present our results for the complexity of constructive and destructive variants of control by deleting voters and candidates. Table 6.1 shows a summary of the complexity results for Approval Voting and Satisfaction Approval Voting rules and highlights results that are due to this thesis.

We first show that destructive control by deleting agents is NP-hard for the Approval Voting rule. It is our only result for AV as the constructive control cases have been established by Meir et al. [61] and the rule is immune to destructive control by deleting candidates (this clearly follows from the immunity results for the single winner variant by Hemaspaandra et al. [47]).

Theorem 6.2.1. *Destructive Control by Deleting Voters for multiwinner approval voting is* NP*-hard.*

Proof. We show a reduction from X3C. Let $(B, \mathcal{S})$ be an instance of X3C, where $B = \{b_1, ..., b_m\}$, $m = 3k$, $k > 0$, $\mathcal{S} = \{S_1, ..., S_n\}$, and for each $i \in \{1, ..., n\}$, $S_i \subseteq B$ and $\|S_i\| = 3$. For each j, $1 \leq j \leq m$, we let $\ell_j = \|\{S_i \in \mathcal{S} \mid b_j \in S_i\}\|$. We construct election $E = (A, N)$ with the set of alternatives $A = B \cup \{p\}$, where p is the distinguished alternative, and with the agents set N constructed as follows:

1. For each $i \in \{1, ..., n\}$ we create an agent v_i that approves of p and all the alternatives from $B \setminus S_i$.

2. For each $j \in \{1, ..., m\}$ we create $\ell_j - 1$ agents $W_j = \{w_j^1, ..., w_j^{\ell_j - 1}\}$ that only approve of b_j.

Now, in the election (A, N), alternative p scores n points and each alternative from the set B scores $n - 1$ points (for each $j \in \{1, ..., m\}$, alternative b_j scores $n - \ell_j$ points from the first group of agents above, and $\ell_j - 1$ points from the second group). We set the size of the winning committee for the multiwinner approval

election (A, N) to be n. Clearly, the score of p is greater by one from all the remaining alternatives, therefore p is in the winning committee.

We need to prove that $\mathcal{S}$ contains an exact cover for B if and only if we can make p not to be included in the winning committee for (A, N) by deleting agents from N.

For the left to right direction, we assume that B has an exact cover on $\mathcal{S}$ of size k. We remove k agents from $v_1, ..., v_n$, that correspond to an exact cover of B. Since each of the deleted agents approves of p, the score of p decreases by k. For each $j \in \{1, ..., n\}$, we have that agent v_j does not approve alternatives from S_j. Therefore by deleting a set of such agents that correspond to an exact cover of B, for each $j \in \{1, ..., m\}$ we deleted exactly one agent that does not approve of b_j. Clearly, the score of each alternative from B will decrease by $k - 1$. This makes the score of all $n + 1$ alternatives equal, therefore if the committee size is n, alternative p will not be included in it (wrt. the tie breaking rule that prefers any alternative to p).

For the right to left direction, we assume that there exists a set of agents M of size k such that p is not in a winning commitee of the multiwinner approval election $(A, N \setminus M)$ with the committee size of n. Without loss of generality, we can assume that M does not contain agents that do not approve of p (deleting such agents will not decrease the score of p). Therefore M needs to consist only of agents from the set $\{v_1, ..., v_n\}$. We see that the score of p in $(A, N \setminus M)$ is $n - k$. This means that for p not to be included in the winning committee of size n, the score of each alternative from B has to be at least $n - k$. Since the initial score of each of the alternative from B was $n-1$, for each b_j, $j \in \{1, ..., m\}$, M needs to contain at least one agent that does not approve of b_j. It is easy thus to note that the selection of agents from $\{v_1, ..., v_n\}$ must correspond to a set cover for B. And since it has to be of size k, it must be an exact cover for B. □

We now consider the SAV rule. Because of the way how points are distributed among the alternatives in SAV, the results from Approval Voting or single winner approval do not directly translate to SAV. In particular, this means that the immunity results for Approval Voting DCDC do not hold anymore. However, in the case of DCDV we can adapt the reduction for Approval Voting presented earlier to also show hardness of that problem under the SAV rule.

Theorem 6.2.2. *Destructive Control by Deleting Voters for multiwinner SAV is* NP*-hard.*

Proof. The proof of this theorem is based on the reduction shown in the proof of Theorem 6.2.1. We change the set of alternatives to include an additional set of $n-3$ dummy alternatives D. We also update the sets of agents $W_1, ..., W_m$ so that for each $j \in \{1, ..., m\}$ the set W_j consists of $\ell - 1$ agents $w_j^1, ..., w_j^{\ell_j - 1}$, each of whom approves alternatives $D \cup \{b_j\}$. Because of that, each of the agents from $\{v_1, ..., v_n\}$ and $W_1 \cup \cdots \cup W_m$ approves exactly $n-2$ alternatives, and gives $\frac{1}{n-2}$ points to each of them. Finally, for each alternative $d \in D$, we add two agents that each approves only d. This improves d's score by two points. To sum up, p now scores $\frac{n}{n-2}$ points, each alternative from B scores $\frac{n-1}{n-2}$ points. In addition, all the alternatives from D score more than two points and are ahead of all the remaining alternatives in the ranking. We now set the size of the committee to be $2n-2$. The rest of the reasoning follows as in the proof of Theorem 6.2.1, with the restriction that alternatives D will always be in the winning committee (which is the reason the committee size has been increased by the size of D, as compared to the proof of Theorem 6.2.1). □

We also rely on X3C for showing the hardness of the constructive control variant.

Theorem 6.2.3. *Constructive Control by Deleting Voters for multiwinner SAV is* NP*-hard.*

Proof. We show a reduction from X3C. Let $(B, \mathcal{S})$ be an instance of X3C, where $B = \{b_1, ..., b_m\}$, $m = 3k$, $k > 0$, $\mathcal{S} = \{S_1, ..., S_n\}$, and for each $i \in \{1, ..., n\}$, $S_i \subseteq B$ and $\|S_i\| = 3$. For each j, $1 \leq j \leq m$, we let $\ell_j = \|\{S_i \in \mathcal{S} \mid b_j \in S_i\}\|$. We make a set $F = \{f_{i,j} | i \in [m], j \in [3]\}$ of $3m$ dummy alternatives and another set $D = \{d_{i,j} | i \in [n], j \in [3]\}$ of $3n$ dummy alternatives. We construct election $E = (A, N)$ with the alternatives set $A = B \cup \mathcal{S} \cup \{p\} \cup F \cup D$ where p is the distinguished alternative, and with the agents set N constructed as follows:

1. We create a set of agents $V = \{v_1, ..., v_n\}$. For each $i \in \{1, ..., n\}$ agent v_i approves three alternatives from S_i (all three belong to B) and an alternative that correspond to a set S_i.

2. For each $j \in \{1, ..., m\}$ we create $n - \ell_j$ agents that approve alternatives b_j, $f_{j,1}$, $f_{j,2}$ and $f_{j,3}$.

3. Finally, for each $i \in \{1, ..., n\}$ we create $n-2$ agents that each approve alternatives S_i, $d_{i,1}$, $d_{i,2}$ and $d_{i,3}$ and one agents that approves S_i, p, $d_{i,1}$ and $d_{i,2}$.

In the election E, alternatives from B, $\mathcal{S}$ and alternative p score $\frac{n}{4}$ points each, while the score of each alternative from F and D is less than $\frac{n}{4}$. We set the size of the winning committee to be $n-k+1$ and break ties in the way that any alternative from B or $\mathcal{S}$ is preferred to p. Clearly, alternative p is not in the SAV winning committee.

We now demonstrate that the instance of X3C has a solution if and only if constructive control by deleting k voters under SAV is possible in E. The left-to-right part of the proof is straightforward. We note that if a set cover exists and we delete agents from V that correspond to the sets from the cover, then the score of each alternative from B will decrease by $\frac{1}{4}$. Also, the score of exactly k alternatives from $\mathcal{S}$ will decrease by $\frac{1}{4}$. As a result we will be left with $n-k$ alternatives from $\mathcal{S}$ with a score of $\frac{n}{4}$ and hence alternative p becomes a winner.

In order to prove the right-to-left direction we assume that the considered control variant is possible. In such a case we only allow $n-k$ alternatives to have greater or equal score to the score of alternative p. We start off by having n alternatives from $\mathcal{S}$ and m alternatives from B with a score equal to p's score, therefore we need to find $m+k$ alternatives from sets B and $\mathcal{S}$ whose score will decrease as a result of deleting at most k agents. We note that if control is possible then we can select these k agents along the agents from set V. This follows from the fact that when deleting agent from V, we decrease the score of four alternatives from sets B and $\mathcal{S}$, whereas deleting any other agent which is not in V will at best decrease the score of at most one such alternative. In addition, we note that there is no point to decrease some alternative by more than $\frac{1}{4}$, because p already has score $\frac{1}{4}$. We now further claim that the set K of k agents that we delete in order for p to be included in the winning committee corresponds to a set cover. To prove this claim, we note that regardless of the selection of K, agents from set K approve exactly k alternatives from $\mathcal{S}$. However, as stated earlier, we need agents from K to approve $m+k$ distinct alternatives from sets B and $\mathcal{S}$. Therefore the only possibility for that to be true is that agents from K approve of m distinct alternatives from B. We now see that because of the shape of the preferences of agents from V, the only possibility for k agents to cover m alternatives from B is for these agents to correspond to a set cover. □

We now move on to candidate control. Similarly to the case of voter control, both constructive and destructive variants of deleting candidates problems are hard for SAV rule.

Theorem 6.2.4. *Destructive Control by Deleting Candidates under SAV is* NP-*hard.*

Proof. We show a reduction from X3C. Let $(B, \mathcal{S})$ be an instance of X3C, where $B = \{b_1, ..., b_m\}$, $m = 3k$, $k > 0$, $\mathcal{S} = \{S_1, ..., S_n\}$, and for each $i \in \{1, ..., n\}$, $S_i \subseteq B$ and $\|S_i\| = 3$. For each j, $1 \leq j \leq m$, we let $\ell_j = \|\{S_i \in \mathcal{S} \mid b_j \in S_i\}\|$. Further we let $F = \{f_{i,j} | i \in [m], j \in [3]\}$ be a set of $3m$ dummy alternatives and D be a set of three dummy alternatives. We construct election $E = (A, N)$ with the set of alternatives $A = B \cup \mathcal{S} \cup \{p\} \cup F \cup D$, where p is the distinguished alternative, and with the agents set N, which consists of the following agents:

1. For each $i \in \{1, ..., n\}$, there are four agents who each approve four alternatives: three alternatives from S_i and the alternative that corresponds to the set S_i. Thus for each $j \in \{1, ..., m\}$ b_j gets $\frac{1}{4}\ell_j$ points from each of these agents, and for each $k \in \{1, ..., n\}$ the alternative that corresponds to S_k has exactly one point.

2. For each $j \in \{1, .., m\}$ we create $4(n - \ell_j)$ agents that approve alternatives b_j and dummy alternatives $f_{j,1}$, $f_{j,2}$ and $f_{j,3}$.

3. We add $n + k$ agents, each only approving of p, and one agent that approves p and all three alternatives from D.

4. For each alternative $f \in F$, we add $n + k + 1$ agents that approves of f only.

5. Finally, for each alternative from B we add k agents who approve that single alternative only. This way we improve the scores of all the alternatives from B by k points.

We set the committee size to be m. Now we note that the points are distributed as follows:

1. Each alternative from B has $k + n$ points.

2. Alternative p has $k + n + \frac{1}{4}$ points.

3. Alternatives from $\mathcal{S}$ score one point each.

4. Alternatives from F score at least $k + n + 1$ points each.

5. Alternatives from D score $\frac{1}{4}$ points each.

We set the size of the winning committee to be $4 \cdot m$ and note that p is a member of this committee under the SAV rule. We claim that $\mathcal{S}$ contains an exact cover for B if and only if p can be precluded from being a member of the winning committee under SAV by deleting candidates.

For the left-to-right direction, it is sufficient to delete k alternatives from $\mathcal{S}$ that corresponds to an exact cover of B. By that, we improve the score of all the alternatives from B by $\frac{1}{3}$ (they used to score $\frac{1}{4}$ of a point from each of the four agents that now contributes $\frac{1}{3}$ of a point each). Clearly, the score of all the remaining alternatives (including p) does not change. Therefore it is sufficient for all the alternatives from B to get a score of $n + k + \frac{1}{3}$, which is greater than the score of p, and therefore, along with alternatives from F, they form a committee of size $4 \cdot m$ that does not include p.

For the right to left direction, we assume that there exist k alternatives that can be deleted in order for p not to be in the winning committee of size $4 \cdot m$. First, we note that by deleting alternatives it is not possible to decrease the score of any of the remaining alternatives and, specifically, of alternative p. We also note that alternatives from D and $\mathcal{S}$ can never score more than $n + k$ points if we were to only delete k alternatives, because deleting an alternative under SAV in the best case scenario can improve the score by only $\frac{1}{2}$ a point. Therefore, it is clear that if p is not in the winning committee of size $4 \cdot m$ after deleting k candidates then the only option is that the winning committee consists of $3 \cdot m$ alternatives from F and m alternatives from B. The score of alternatives from F is already greater than the score of p, so in order for that to happen we need to improve the score of all the alternatives from B. Clearly, deleting alternatives from B and F makes no sense as then we would not have sufficiently many alternatives to fill the committee of size $4 \cdot m$. Also, deleting alternatives from D makes no sense as it will only improve the score of other alternatives from D and the score of p. Hence, the only possibility is to delete k alternatives from $\mathcal{S}$. Since we need the score of all the alternatives from B to improve, it is easy to note that this can only happen if we delete alternatives that correspond to a set cover (if we were to delete k alternatives that do not correspond to a set cover then we would have at least one alternative from B whose score would not improve). In such a scenario, the score of each alternative from B improves by $\frac{1}{3}$ and is now $n + k + \frac{1}{3}$ which is greater than the score of p. And therefore, since there are $3 \cdot m$ alternatives from F with a score of at least $n + k + 1$, the alternative p is not a part of the winning committee of size $4 \cdot m$ under SAV. □

Theorem 6.2.5. *Costructive Control by Deleting Candidates for multiwinner SAV is in* NP*-hard.*

Proof. Again, for this proof we show a reduction from X3C. Let $(B, \mathcal{S})$ be an instance of X3C, where $B = \{b_1, ..., b_m\}$, $m = 3k$, $k > 0$, $\mathcal{S} = \{S_1, ..., S_n\}$, and for each $i \in \{1, ..., n\}$, $S_i \subseteq B$ and $\|S_i\| = 3$. Without loss of generality, we assume that $n \geq k$ as otherwise it is easy to see that the solution of X3C does not exist. For each $j \in \{1, ..., m\}$ we let $\ell_j = \|\{S_i \in \mathcal{S} \mid b_j \in S_i\}\|$. Further, we let $F = \{f_{i,j} | i \in [m], j \in [4]\}$ be a set of $4m$ dummy alternatives and $D = \{d_{i,j} | i \in [m], j \in [19]\}$ be a set of $19m$ dummy alternatives. We construct an election E with the alternatives set $A = B \cup \mathcal{S} \cup \{p\} \cup F \cup D$, and with the agents set N, which consists of the following agents:

1. For each $i \in \{1, ..., n\}$ we create an agent that approves of the alternative which corresponds to the set S_i, alternative p and the three members of S_i.

2. For each $i \in \{1, ..., n\}$ we create $n \cdot k$ agents that approve of the alternative that corresponds to the set S_i.

3. For each $j \in \{1, ..., m\}$ we create $n - \ell_j$ agents that approve of alternative b_j and four alternatives $f_{j,1}$, $f_{j,2}$, $f_{j,3}$ and $f_{j,4}$

4. For each $j \in \{1, ..., m\}$ we create $k - 2$ agents that approve of alternative b_j and 19 alternatives from $d_{j,1}...d_{j,19}$.

As a result, the score of each alternative corresponding to an item from B is now $\frac{n}{5} + \frac{k-2}{20}$. This follows from the fact that for each $j \in \{1, ..., m\}$ alternative b_j scores $\frac{1}{5}$ for every set from $\mathcal{S}$ that it is a part of; that is $\frac{\ell_j}{5}$ points in total. Then, b_j also scores $\frac{1}{5}$ of a point from the $n - \ell_j$ agents that approve it along with the dummy alternatives from F. Finally, it scores $\frac{1}{20}$ of a point from each of $k - 2$ agents that approves it along with the dummy alternatives from D. We further see that the score of alternative p is $\frac{n}{5}$, the score of each alternative that corresponds to a set from $\mathcal{S}$ is at least $n + k$ and that the scores of the dummy alternatives from F are at most $\frac{n}{5}$, while the dummy alternatives from D score $\frac{k-2}{20}$ points each.

We now set the committee size of the SAV election (A, N) to be $n - k + 1$ and claim that the instance of X3C has a solution if and only if constructive control by deleting k candidates is possible in our election. In order to show the left-to-right direction of this statement, it is enough to show what will happen if a set cover exists and we delete k alternatives that correspond to the sets from this cover. For

each $b_j \in B$, the score of the alternative b_j will increase by $\frac{1}{20}$. This comes from the fact that each alternative corresponding to $b \in B$ has been previously approved by exactly one agent who also approved of an alternative corresponding to the set from the cover, along with two other alternatives from B and alternative p. Therefore such agent would previously contribute $\frac{1}{5}$ of a point to that alternative's score, while after deleting the alternative from the set cover it contributes $\frac{1}{4}$ of a point. Finally the score of alternative p improves by $\frac{k}{20}$. As a result, we are left with $n-k$ alternatives corresponding to the sets which are not in the cover and whose scores are greater than $n \cdot k$, the alternatives from B, each with score of $\frac{n}{5} + \frac{k-1}{20}$ (improved by $\frac{1}{20}$), and alternative p with score $\frac{n}{5} + \frac{k}{20}$ (improved by $\frac{k}{20}$); the remaining alternative's scores did not change (because no dummy alternative has ever been approved along with the alternatives that correspond to the sets from $\mathcal{S}$). As a consequence, p is a part of the winning committee of size $n - k + 1$.

In order to show the right-to-left direction of the claim, we first note that it is never beneficial to delete alternatives other than those which correspond to the sets from $\mathcal{S}$. This is true because if we delete k alternatives, the score of p can never improve beyond $n \cdot k$, which follows from the fact that deleting a candidate can improve the score by no more than $\frac{1}{2}$ a point per agent, and we only have n agents that approve of p. Also, if we were to delete fewer than k alternatives that correspond to the sets from $\mathcal{S}$, we would be left with at least $n - k + 1$ such alternatives. We know that the score of each such an alternative is at least $n \cdot k$ and, therefore, would not be possible for p to make it to the winning committee.

Knowing that if constructive control by deleting candidates is possible only if we delete k alternatives that correspond to the sets from $\mathcal{S}$, we now move on to showing that it is only possible if we pick alternatives that correspond to the sets from a set cover. Let us take any subset $\mathcal{S}' \subseteq \mathcal{S}$ of size k that is not a set cover. If we now were to delete alternatives that correspond to the sets from $\mathcal{S}'$, the score of p would become $\frac{n}{5} + \frac{k}{20}$. However, we note that if $\mathcal{S}'$ were not a set cover, then there would exist index $t \in \{1, ..., m\}$ such that some b_t belonged to at least two sets from $\mathcal{S}'$. Therefore, after deleting the alternatives that correspond to sets from $\mathcal{S}'$, the score of b_t would improve by at least $\frac{2}{20}$ and would be at least $\frac{n}{5} + \frac{k}{20}$ and hence greater or equal to that of p. Knowing that after deleting k alternatives we are still left with $n - k$ alternatives whose score is greater than $n \cdot k$, it is easy to note that under such circumstances alternative p would not be a part of the winning committee of size $n - k + 1$. □

6.3 Conclusions

We have presented hardness proofs for constructive and destructive control problems by deleting voters and candidates for the SAV rule and NP-hardness proof for DCDV problem under Approval Voting. Our results contribute new data points to the field of study of multiwinner election rules in the context of the complexity of manipulating their results. Table 6.1 contains all of our results. The fact that most of those manipuation problems are NP-hard resonates with the ongoing debate whether worst-case complexity is a good benchmark for single winner rules. As with this work, we have barely scratched the surface by considering the basic approval-based multiwinner rules. It would be surprising to see more complex rules such as Greedy Approval Voting [58] or Reweighted Approval Voting [14] to be susceptible to the considered control problems. Given that, considering restricted domains in the context of control problems for multiwinner election is very tempting. The concept of restricted profiles translates seamlessly into the field of multiwinner election. We would hope for that research to give us better understanding on whether worst-case complexity results can be applied to the real life situations.

Conclusions and Future Directions

In this thesis we have presented a number of computational complexity results for various election control problems under different criteria. We considered restricted domains as differentiating factor for the variety of control problems we analyzed. In Chapter 4 we discussed single-crossing domain in the context of classic election control problem, and in Chapter 5 we took on the single-peaked domain but considered counting variants of control problems. Finally, in Chapter 6 we presented results for multiwinner voting rules under unrestricted cases. On top of that, in Chapter 3 we showed recognition algorithms for aforementioned single-peaked and single-crossing domains, but also for the top-monotonic domain.

The results we have presented on the complexity of control problem variants (i.e., in Chapters 4 and 5) show that considering restricted domains in many popular cases renders the originally hard problems polynomial-time solvable. Despite this fact being previously noted for single-peaked domain, our research further confirms this occurrence for different control problem settings and other restricted domains. This contributes important data points to the discussion on whether computational complexity can be used as a barrier to election manipulation. Yet, knowing that election control can be solved in polynomial-time for a given voting rule under single-peaked and single-crossing domain, but is intractable in an unrestricted case, still leaves us with an unanswered question on what are the criteria that make a given control computationally hard. One of the things that could help us understand it better is top-monotonicity, and in our opinion this is the case for two main reasons. First, top-monotonicity is not just some synthetic notion but it has been designed for the profiles to meet some important criteria, namely the existence of (weak) Condorcet winners. It would be interesting to learn whether tractability of election control problems can be associated with such profile properties. Second, top-monotonic profiles sit in between well-researched profile spaces of single-peaked or single-crossing domains and the unrestricted cases. That is, it puts less strict criteria on the shapes of the voting profiles.

Lastly, in Chapter 6 we have shown complexity results for unrestricted control problems under multiwinner elections. We believe that the area of multiwinner elections is particularly interesting, as even the basic voting rules are computationally hard to manipulate. This opens up new areas where restricted domains can be applied to broaden the understanding about the real-life complexity of election control.[2]

A general conclusion from out work is that, indeed, all sorts of election problems become significantly easier under restricted domains. In particular, this phenomenon is not limited to single-peaked elections. Yet, our work also leads to many new research directions. Foremost, it is interesting what algorithmic consequences follow from assuming the voters' preferences to be top-monotonic. In particular, it is interesting to study control by adding/deleting alternatives, because it is known that a top-monotonic election may cease to be top-monotonic after deleting some candidates [7].

Another natural research direction is to consider different restricted domains. For example, one may look at group-separable [49, 50], value-restricted [78, 79], or Euclidean preferences [30, 31] (here it is known that various control problems are in P for 1D-Euclidean preferences — which are both single-peaked and single-crossing — but one may expect NP-hardness results for higher dimensions).

Finally, we have barely touched upon control in multiwinner voting, and this area deserves far more detailed studies (which, indeed are now pursued [17, 43, 63, 65, 92, 93]).

[2]And, indeed, during the course of this thesis being written, other authors studied multiwinner voting with restricted domains in depth [91].

Bibliography

[1] K. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, 1951. Revised editon, 1963.

[2] H. Aziz. Testing top monotonicity. Technical Report arXiv:1403.7625v5 [cs.GT], arXiv.org, June 2014.

[3] H. Aziz, M. Brill, V. Conitzer, E. Elkind, R. Freeman, and T. Walsh. Justified representation in approval-based committee voting. *Social Choice and Welfare*, 48(2):461–485, 2017.

[4] H. Aziz, S. Gaspers, J. Gudmundsson, S. Mackenzie, N. Mattei, and T. Walsh. Computational aspects of multi-winner approval voting. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 107–115. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[5] Y. Bachrach, N. Betzler, and P. Faliszewski. Probabilistic possible winner determination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 697–702. AAAI Press, July 2010.

[6] M. Ballester and G. Haeringer. A characterization of the single-peaked domain. *Social Choice and Welfare*, 36(2):305–322, 2011.

[7] S. Barberà and B. Moreno. Top monotonicity: A common root for single peakedness, single crossing and the median voter result. *Games and Economic Behavior*, 73(2):345–359, 2011.

[8] J. Bartholdi, III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical and Computer Modeling*, 16(8/9):27–40, 1992.

[9] J. Bartholdi, III and M. Trick. Stable matching with preferences derived from a psychological model. *Operations Research Letters*, 5(4):165–169, 1986.

[10] N. Betzler and B. Dorn. Towards a dichotomy of finding possible winners in elections based on scoring rules. *Journal of Computer and System Sciences*, 76(8):812–836, 2010.

[11] N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, 47(1):475–519, 2013.

[12] D. Black. On the rationale of group decision-making. *Journal of Political Economy*, 56(1):23–34, 1948.

[13] D. Black. *The Theory of Committees and Elections*. Cambridge University Press, 1958.

[14] S. Brams and D. M. Kilgour. Satisfaction approval voting. In R. Fara, D. Leech, and M. Salles, editors, *Voting Power and Procedures*, pages 323–346. Springer, 2014.

[15] F. Brandt, M. Brill, E. Hemaspaandra, and L. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 715–722. AAAI Press, July 2010.

[16] R. Bredereck, J. Chen, and G. Woeginger. A characterization of the single-crossing domain. *Social Choice and Welfare*, 41(4):989–998, October 2013.

[17] R. Bredereck, P. Faliszewski, A. Kaczmarczyk, R. Niedermeier, P. Skowron, and N. Talmon. Robustness among multiwinner voting rules. In *Proceedings of the 10th International Symposium on Algorithmic Game Theory*, pages 80–92. Springer, 2017.

[18] E. Brelsford, P. Faliszewski, E. Hemaspaandra, H. Schnoor, and I. Schnoor. Approximability of manipulating elections. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 44–49. AAAI Press, July 2008.

[19] Y. Chevaleyre, J. Lang, N. Maudet, and J. Monnot. Possible winners when new candidates are added: The case of scoring rules. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 762–767. AAAI Press, July 2010.

[20] T. Coleman and V. Teague. On the complexity of manipulating elections. In *The Thirteenth Computing: The Australasian Theory Symposium (CATS2007)*, pages 25–33. Australian Computer Society Inc., January/February 2007.

[21] V. Conitzer. Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research*, 35:161–191, 2009.

[22] V. Conitzer and T. Walsh. Barriers to manipulation in voting. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia, editors, *Handbook of computational social choice*, chapter 6, pages 127–145. Cambridge University Press, 2016.

[23] D. Cornaz, L. Galand, and O. Spanjaard. Bounded single-peaked width and proportional representation. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 270–275. IOS Press, August 2012.

[24] M. Davis, M. Orrison, and F. Su. Voting for committees in agreeable societies. In K.-D. Crisman and M. Jones, editors, *The Mathematics of Decisions,*

Elections, and Games (Contemporary Mathematics, Vol. 624), pages 147–157. American Mathematical Society, 2014.

[25] P. Dey and N. Misra. Elicitation for preferences single peaked on trees. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 215–221. AAAI Press, 2016.

[26] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference*, pages 613–622. ACM Press, March 2001.

[27] E. Elkind, P. Faliszewski, and A. Slinko. Clone structures in voters' preferences. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 496–513. ACM Press, June 2012.

[28] E. Elkind and M. Lackner. Structure in dichotomous preferences. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 2019–2025. AAAI Press, 2015.

[29] E. Elkind, J. Lang, and A. Saffidine. Condorcet winning sets. *Social Choice and Welfare*, 44(3):493–517, 2015.

[30] J. Enelow and M. Hinich. *The spatial theory of voting: An introduction*. Cambridge University Press, 1984.

[31] J. Enelow and M. Hinich. *Advances in the spatial theory of voting*. Cambridge University Press, 1990.

[32] E. Ephrati and J. Rosenschein. A heuristic technique for multi-agent planning. *Annals of Mathematics and Artificial Intelligence*, 20(1–4):13–67, 1997.

[33] G. Erdélyi, M. Fellows, J. Rothe, and L. Schend. Control complexity in bucklin and fallback voting: An experimental analysis. *Journal of Computer and System Sciences*, 81(4):661–670, 2015.

[34] B. Escoffier, J. Lang, and M. Öztürk. Single-peaked consistency and its complexity. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 366–370. IOS Press, July 2008.

[35] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. The complexity of manipulative attacks in nearly single-peaked electorates. *Artificial Intelligence*, 207:69–99, February 2014.

[36] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. A richer understanding of the complexity of election systems. In S. Ravi and S. Shukla, editors, *Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz*, pages 375–406. Springer, 2009.

[37] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. 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.

[38] P. Faliszewski and J. Rothe. Control and bribery in voting. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia, editors, *Handbook of computational social choice*, chapter 7, pages 146–168. Cambridge University Press, 2016.

[39] P. Faliszewski, J. Sawicki, R. Schaefer, and M. Smolka. Multiwinner voting in genetic algorithms. *IEEE Intelligent Systems*, 32(1):40–48, 2017.

[40] P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. Multiwinner voting: A new challenge for social choice theory. In U. Endriss, editor, *Trends in Computational Social Choice*. AI Access Foundation, 2017.

[41] P. Faliszewski, P. Skowron, and N. Talmon. Bribery as a measure of candidate success: Complexity results for approval-based multiwinner rules. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 6–14. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[42] P. Fishburn. Acyclic sets of linear orders. *Social Choice and Welfare*, 14(1):113–124, 1997.

[43] G. Gawron and P. Faliszewski. Robustness of approval-based multiwinner voting rules. In *Proceedings of the 6th International Conference on Algorithmic Decision Theory*, pages 17–31. Springer, 2019.

[44] S. Ghosh, M. Mundhe, K. Hernandez, and S. Sen. Voting for movies: The anatomy of recommender systems. In *Proceedings of the 3rd Annual Conference on Autonomous Agents*, pages 434–435. ACM Press, 1999.

[45] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41(4):587–601, 1973.

[46] N. Hazon, Y. Aumann, S. Kraus, and M. Wooldridge. On the evaluation of election outcomes under uncertainty. *Artificial Intelligence*, 189:1–18, 2012.

[47] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(5–6):255–285, 2007.

[48] H. Hunt, M. Marathe, V. Radhakrishnan, and R. Stearns. The complexity of planar counting problems. *SIAM Journal on Computing*, 27(4):1142–1167, 1998.

[49] K. Inada. A note on the simple majority decision rule. *Econometrica*, 32(32):525–531, 1964.

[50] K. Inada. The simple majority decision rule. *Econometrica*, 37(3):490–506, 1969.

[51] A. Karpov. On the number of group-separable preference profiles. *Group Decision and Negotiation*, 28(3):501–517, 2019.

[52] O. Keller, A. Hassidim, and N. Hazon. Approximating weighted and priced bribery in scoring rules. *Journal of Artificial Intelligence Research*, 66:1057–1098, 2019.

[53] O. Keller, A. Hassidim, and N. Hazon. New approximations for coalitional manipulation in scoring rules. *Journal of Artificial Intelligence Research*, 64:109–145, 2019.

[54] B. Kenig and B. Kimelfeld. Approximate inference of outcomes in probabilistic elections. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 2061–2068. AAAI Press, 2019.

[55] B. Kimelfeld, P. Kolaitis, and M. Tibi. Query evaluation in election databases. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 32–46. ACM Press, 2019.

[56] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proceedins of the Multidisciplinary IJCAI-05 Worshop on Advances in Preference Handling*, pages 124–129, July/August 2005.

[57] R. LeGrand, E. Markakis, and A. Mehta. Some results on approximating the minimax solution in approval voting. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1185–1187. International Foundation for Autonomous Agents and Multiagent Systems, 2007.

[58] T. Lu and C. Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 280–286, 2011.

[59] K. Magiera and P. Faliszewski. How hard is control in single-crossing elections? *Autonomous Agents and Multi-Agent Systems*, 31(3):606–627, May 2017.

[60] K. Magiera and P. Faliszewski. Recognizing top-monotonic preference profiles in polynomial time. *Journal of Artificial Intelligence Research*, 66:57–84, 2019.

[61] R. Meir, A. Procaccia, J. Rosenschein, and A. Zohar. The complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research*, 33:149–178, 2008.

[62] J. Mirrlees. An exploration in the theory of optimal income taxation. *Review of Economic Studies*, 38:175–208, 1971.

[63] N. Misra and C. Sonar. Robustness radius for chamberlin-courant on restricted domains. In *Proceedings of the 45th International Conference on Current Trends in Theory and Practice of Computer Science*, pages 341–353. Springer, 2019.

[64] T. Miąsko. Algorithms and complexity results for election control problems with prices. Master's thesis, AGH University of Science and Technology, Kraków, Poland, 2011.

[65] M. Neveling and J. Rothe. The complexity of cloning candidates in multiwinner elections. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, pages 922–930. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[66] S. Obraztsova, Y. Zick, and E. Elkind. On manipulation in multi-winner elections based on scoring rules. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 359–366. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[67] T. Persson and G. Tabellini. *Political Economics: Explaining Economic Policy*. MIT Press, 2000.

[68] D. Peters. Recognising multidimensional euclidean preferences. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 642–648. AAAI Press, 2017.

[69] D. Peters and E. Elkind. Preferences single-peaked on nice trees. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 594–600. AAAI Press, 2016.

[70] D. Peters and M. Lackner. Preferences single-peaked on a circle. *Journal of Artificial Intelligence Research*, 68:463–502, 2020.

[71] M. Pourghanbar, M. Kelarestaghi, and F. Eshghi. EVEBO: A new election inspired optimization algorithm. In *Proceedings of the 2015 IEEE Congress on Evolutionary Computation*, pages 916–924, 2015.

[72] A. Procaccia and J. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence Research*, 28:157–181, 2007.

[73] A. Procaccia, J. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, 2008.

[74] K. Roberts. Voting over income tax schedules. *Journal of Public Economics*, 8(3):329–340, 1977.

[75] P. Rothstein. Representative voter theorems. *Public Choice*, 72(2–3):193–212, December 1991.

[76] A. Saporiti and F. Tohmé. Single-crossing, strategic voting and the median choice rule. *Social Choice and Welfare*, 26(2):363–383, 2006.

[77] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

[78] A. Sen. A possibility theorem on majority decisions. *Econometrica*, 34(2):491–499, 1966.

[79] A. Sen and P. Pattanaik. Necessary and sufficient conditions for rational choice under majority decision. *Journal of Economic Theory*, 1(2):178–202, 1969.

[80] P. Skowron, P. Faliszewski, and A. Slinko. Achieving fully proportional representation: Approximability results. *Artificial Intelligence*, 222:67–103, 2015.

[81] P. Skowron, L. Yu, P. Faliszewski, and E. Elkind. The complexity of fully proportional representation for single-crossing electorates. *Theoretical Computer Science*, 569:43–57, 2015.

[82] J. Sliwinski and E. Elkind. Preferences single-peaked on a tree: Sampling and tree recognition. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 580–586. AAAI Press, 2019.

[83] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[84] T. Walsh. Uncertainty in preference elicitation and aggregation. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 3–8. AAAI Press, July 2007.

[85] K. Wojtas. The complexity of control problems in elections. Master's thesis, AGH University of Science and Technology, Kraków, Poland, 2011.

[86] K. Wojtas and P. Faliszewski. Possible winners in noisy elections. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1499–1505. AAAI Press, July 2012.

[87] K. Wojtas, K. Magiera, T. Miąsko, and P. Faliszewski. Possible Winners in Noisy Elections. Technical Report arXiv:1405.6630 [cs.GT], arXiv.org, May 2014.

[88] L. Xia and V. Conitzer. Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research*, 41:25–67, 2011.

[89] L. Xia, J. Lang, and J. Monnot. Possible winners when new alternatives join: New results coming up! In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 829–836. International Foundation for Autonomous Agents and Multiagent Systems, May 2011.

[90] Y. Yang. Complexity of controlling nearly single-peaked elections revisited. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, pages 2139–2141. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[91] Y. Yang. Complexity of manipulating and controlling approval-based multiwinner voting. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 637–643. AAAI Press, 2019.

[92] Y. Yang. Complexity of manipulating and controlling approval-based multiwinner voting. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 637–643, 2019.

[93] Y. Yang. On the complexity of destructive bribery in approval-based multiwinner voting. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, pages 1584–1592. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[94] Y. Yang and J. Guo. The control complexity of r-approval: From the single-peaked case to the general case. *Journal of Computer and System Sciences*, 89:432–449, 2017.

[95] Y. Yang and J. Guo. Parameterized complexity of voter control in multi-peaked elections. *Theoretical Computer Science*, 62(8):1798–1825, 2018.

[96] L. Yu, H. Chan, and E. Elkind. Multiwinner elections under preferences that are single-peaked on a tree. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 425–431. AAAI Press, 2013.