Approval-Based Elections in the 1D-Euclidean Domain

Jan Sznajd

Supervisor: prof. dr hab. inż. Piotr Faliszewski

Institute of Computer Science
AGH University of Science and Technology

This dissertation is submitted for the degree of
Master of Science

July 2021

Acknowledgements

I would like to acknowledge and express my genuine gratitude to my supervisor, prof. dr hab. inż. Piotr Faliszewski for his patience and guidance. His countless substantive remarks and comprehensive supervision helped me a lot with the research.

Abstract

We study approval-based elections, that is, elections where each voter approves a subset of candidates, and disapproves the remaining ones. Our focus is on preference restrictions, in particular on a recently introduced model: the Voter/Candidate Range, defined in the work of Godziszewski et al. (2021). We explore multiple areas: First, we create a domain detection algorithm based on a reduction to integer linear programming, and compare it with existing solutions. Next, we use the said algorithm in an experiment, in order to establish the probability with which a random election falls within a certain domain. At last, we provide a polynomial-time algorithm for the approval-voting constructive control problem, for elections in the one-dimensional Voter/Candidate Range domain.

Contents

1 Introduction **1**

1.1 Preference Restrictions 2

1.2 Goals and Contributions 3

2 Preliminaries **4**

2.1 Basic Definitions 4

2.2 Preference Restrictions 6

2.2.1 Euclidean Preference Restrictions 7

2.3 Integer Linear Programming 9

3 Voter/Candidate Range Profile Detection **12**

3.1 Profile Structure Detection 12

3.2 ILP-Based Algorithm 13

3.3 Performance Comparison of ILP and SAT Based Solutions for COP Detection 15

3.4 Conclusions 16

4 Voter/Candidate Range Profiles **18**

4.1 Domain Analysis 18

4.1.1 Experiment 1: How Many Profiles Are TVCR or VCR? 19

4.1.2 Experiment 2: How Many Profiles Are TVCR, CR, VR or FCOP? . 19

4.2 Transforming a TVCR Profile into a CR or a VR One 26

4.2.1 Experiment 3: How Different Is a TVCR Profile from CR/VR Ones? 26

5 Control Problem **32**

5.1 Background 32

5.1.1 Constructive Control by Deleting Voters For VR Elections 33

5.1.2 Constructive Control by Deleting Voters For CR Elections 35

5.2 Constructive Control by Deleting Voters For VCR Elections 35

5.2.1 Special Case of Contained Candidates 37

5.2.2 Sketch Proof For CR Elections 39
5.2.3 Sketch Proof For VR Elections 40
5.2.4 Proof For the VCR Algorithm 41
5.2.5 Discussion 42

6 Summary **43**
6.1 Conclusions 43
6.2 Open Problems 43

Bibliography **45**

Chapter 1

Introduction

Elections are a ubiquitous element of our society in one form or another. For example, societies need to choose members of parliaments, referees have to pick winners of sport competitions, or citizens need to elect mayors of cities. However, elections are not only applicable to politics or competitions. In more day-to-day examples of elections, teams have to schedule meeting times, video streaming platforms need to provide lists of recommended movies to the users, or companies have to allocate yearly budgets. Each of these problems may require a different type of an election and a different strategy of selecting the winner (or winners). The differences come from the underlying properties and objectives of the problem. For example, in presidential elections we want to choose the candidate supported by the majority of the voters. It is not necessarily the case for the parliamentary elections, where the goal is to select members of the parliament in such a way that the opinions of the voters are represented proportionally. Another example would be the problem of scheduling a meeting, where we would try to pick the time that maximizes the number of participants who can attend, even though the selected time might not be optimal for anyone (e.g., early morning). In sport competitions, especially in the seasonal ones, i.e., the ones consisting of series of events, we need to come up with a strategy of calculating overall ranking at the end of the season, based on the aggregated results from the particular events. For example, in FIS Alpine Ski World Cup, or in Formula 1 World Championship, the overall winners achieve very high scores consistently throughout the season, but might not have necessarily achieved the highest scores (1st place) in the majority of the individual events.

We can classify election types based on the representation of voters' ballots (preferences). In this work we focus on approval elections, that is, on elections where each voter provides a subset of candidates that he or she approves of. Another notable type of elections is the ordinal model, where the voters order the candidates from the most to the least preferred

ones. For an extensive survey of the approval model and current state of the art regarding committee elections, we point the reader to a survey by Lackner and Skowron (2020).

Computational social choice theory has a wide variety of problems it tries to solve. Above-mentioned examples consider a winner determination problem. It is one of the core problems, however not the only one. Other ones include, e.g., control problems (see, e.g., the works of Bartholdi et al. (1992); Hemaspaandra et al. (2009)), which refer to certain ways of manipulating elections. In short, we try to make a given candidate win (or lose) by changing the structure of an election, that is, by adding or deleting voters or candidates. Other interesting problems, that we are not going to explore in this work, include manipulation (see, e.g., the works of Bartholdi et al. (1989); Faliszewski and Procaccia (2010), and the survey of Conitzer and Walsh (2016)), bribery (see, e.g., the works of Faliszewski et al. (2009), and the survey of Faliszewski and Rothe (2016)), justified representation (see, e.g., the works of Aziz et al. (2017)) and many others (see the handbook edited by Brandt et al. (2016)).

1.1 Preference Restrictions

Many of the above-mentioned social choice problems are known to be computationally hard. We can deal with this complexity using various methods, such as designing approximation algorithms (random or deterministic) (see the handbook by Vazirani (2001)) or parametrized algorithms (see the handbook by Cygan et al. (2015)). An alternative approach is to introduce restrictions on the preference domains. In other words, we restrict the structure of the voters' ballots. There is a successful and active line of research in this area. It begins with the notions of *single-peaked (SP)* (see the works of Black (1958); Arrow (1951)) and *single-crossing (SC)* (see the works of Mirrlees (1971); Roberts (1977)) elections. Much more recently, Barberà and Moreno (2011) introduced the concept of *top-monotonicity*, a generalization of the SP and SC restrictions. These models provided a basis for numerous and promising results in circumventing the computational complexity in a variety of problems. (see, e.g., the works of Walsh (2007); Faliszewski et al. (2011); Skowron et al. (2015); Brandt et al. (2015); Elkind et al. (2016)). Naturally, many other restricted domains were also studied.

In the context of this work, we emphasize a survey by Elkind and Lackner (2015). Unlike most of the above-mentioned examples, which considered the ordinal model of preferences, the survey explored well-known preference restrictions in the setting of approval voting, and introduced several new ones. This includes models such as *Voter Interval (VI)* and *Candidate Interval (CI)*, both based on the *Dichotomous Euclidean (DE)* restriction. Euclidean models are very intuitive, a simplified explanation of such models is that each candidate and each voter is assigned an ideal point in some Euclidean space $\mathbb{R}^d$, this point represents their

opinion on some d issues. Furthermore, each voter has an additional radius parameter. The points are assigned in such a way that voters approve the candidates that are within their radius, and prefer the closer ones. Godziszewski et al. (2021) extended this model, making it more versatile, by using a radius in addition to a point, to represent both the candidates and voters. They named the new model a *Voter/Candidate Range (VCR)* model. We note that the VI and CI models are special cases of the one dimensional VCR model. However, Godziszewski et al. (2021) use different terminology, and their names for the VI and CI are one-dimensional *Candidate Range (1D-CR)* and one-dimensional *Voter Range (1D-VR)* models, respectively.

1.2 Goals and Contributions

The goal of this work is to study the one-dimensional Voter/Candidate Range model (1D-VCR), and to compare it with the 1D-CR and 1D-VR models. Our main contributions are as follows:

1. In Chapter 3, we provide detection algorithm for the 1D-VCR property, based on a reduction to integer linear programming (ILP). Additionally, we apply it to detecting 1D-VR and 1D-CR profiles and, later, we do a performance comparison with a solution based on a 2-SAT reduction due to Fitzsimmons and Lackner (2020).

2. In Chapter 4, we analyze the 1D-VCR domain and compare it to other well known models. We use multiple random distributions to generate elections in the 1D-VCR model, then we check how many of them do not satisfy 1D-VR or 1D-CR properties. Furthermore, we inspect how different are the 1D-VCR elections from the 1D-VR and 1D-CR ones.

3. In Chapter 5, we propose a polynomial time algorithm for a certain control problem under the 1D-VCR model, expanding on the solutions for the 1D-CR and 1D-VR models, due to Magiera and Faliszewski (2017) and Faliszewski et al. (2011), respectively.

We do not discuss the related work right now but, instead we do it throughout this thesis, as the discussion will greatly benefit from the added context and definitions introduced in the subsequent chapters.

Chapter 2

Preliminaries

In this chapter we provide the necessary background for our work. First, in Section 2.1, we introduce basic definitions and core concepts, together with simple examples. Later, in Section 2.2, we discuss preference restrictions, and how we can leverage them to design better algorithms for many computational social choice problems. Finally, in Section 2.3, we explain some computational methods that we have used in this work.

2.1 Basic Definitions

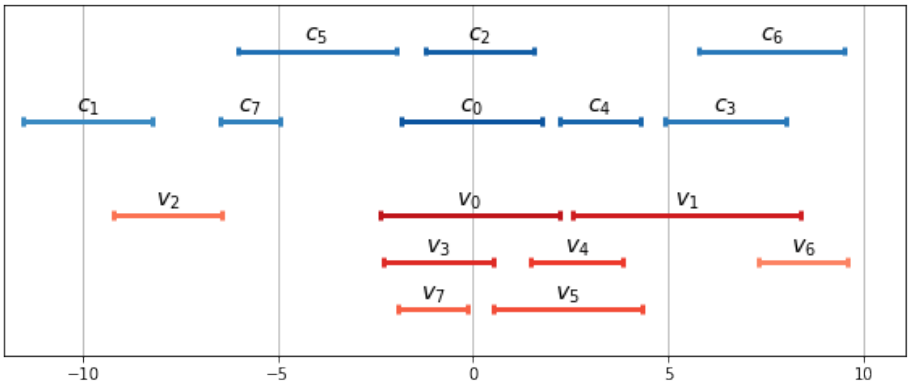
For an integer t, by $[t]$ we mean the set $\{1,...,t\}$. To describe a matrix $\mathbf{A}$ with r rows and c columns, we write $\mathbf{A}_{r\times c}$. By $\mathbf{A}[i,j]$ we refer to the entry in the i-th row and in the j-th column of matrix $\mathbf{A}$. We write $\mathbb{R}^{0+}$ to denote the set of real numbers greater than or equal to zero.

Approval Elections

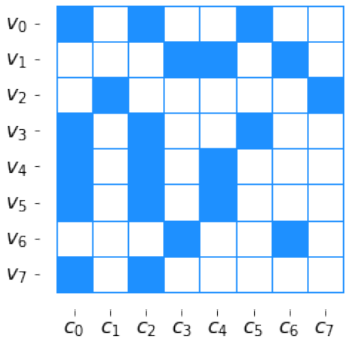
An approval-based election (in short, an *election*, or, interchangeably, a *profile*) is a pair $E=(C,V)$, where $C=\{c_1,...,c_M\}$ is a set of candidates and $V=\{v_1,...,v_N\}$ is a set of voters. We use M and N for the numbers of candidates and voters respectively. For each voter $v\in V$, his or her preferences are represented as a set of approved candidates $A(v)$, which we call an *approval set*.[1] Furthermore, for each candidate $c\in C$, we define $V(c)=\{v\in V\mid c\in A(v)\}$ to be the set of voters who approve candidate c.

In practice, we often use additional, simplified notation for representing an election. We redefine A as a binary matrix (a matrix where each entry is in $\{0,1\}$) with N rows and M

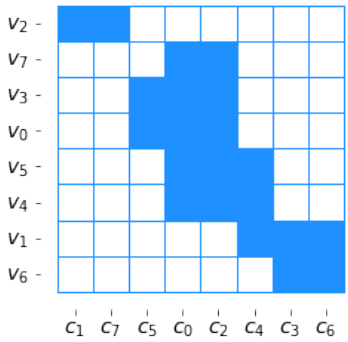
[1] Let us mention another widely studied type of elections, i.e., the *ordinal* elections. In this model the voters order the candidates from the most to the least preferred ones. Formally, each voter v provides a linear order $\succ_v$, which ranks all the candidates.



(a) VCR representation (see Restriction 2 in Section 2.2.1)



(b) Unordered approval matrix representation



(c) Ordered approval matrix representation

Figure 2.1 Three graphical representations of the same approval election from Example 1. The candidates are labeled with c_i and the voters with v_i. Figure 2.1a illustrates the agents in the VCR model, i.e., every agent is represented by the point x and radius r (see Restriction 2 in Section 2.2.1 for the explanation). Figures 2.1b and 2.1c visualize approval matrices of the election. Solid color corresponds to 1 and white to 0 in the cells of the matrix. The first one is presented in a natural order of the agents, and in the second the agents are ordered by their point x.

columns. For $i \in [N]$ and $j \in [M]$, entry $A[i, j]$ is 1 if voter v_i approves of candidate c_j, and it is 0 otherwise. In other words, each row of the matrix is a vector representation of a given voter's approval set. We refer to it as an *approval matrix* or a *preference matrix*. It will always be clear from the context which notation we are using.

Example 1. *Let us describe an approval election, which corresponds to the one in Figure 2.1. There are eight candidates* $C = \{c_0, \ldots, c_7\}$ *and there are eight voters* $V = \{v_0, \ldots, v_7\}$. *Their*

ballots (preferences) are:

$$A(v_0):\{c_0,c_2,c_5\} \quad A(v_1):\{c_3,c_4,c_6\} \quad A(v_2):\{c_1,c_7\} \quad A(v_3):\{c_0,c_2,c_5\}$$
$$A(v_4):\{c_0,c_2,c_4\} \quad A(v_5):\{c_0,c_2,c_4\} \quad A(v_6):\{c_3,c_6\} \quad A(v_7):\{c_0,c_2\}.$$

The reader may wonder who should be a winner given this approval profile. We answer this question in the subsequent section in Example 2.

Single-winner Selection Rules

Voting rules are algorithms used to select the winner based on the voters preferences for a given election.[2]Each rule has different properties and must be selected according to the goal of the election. As this is not the main focus of this work, we only introduce the most basic voting rule for the approval model, simply called *Approval Voting (AV)*, and introduced by Brams and Fishburn (2007).

Definition 2.1.1 (Single-Winner Approval Voting (AV))**.** *For a given election $E=(C,V)$, AV rule selects the candidate with the highest approval score. AV-score of a given candidate c is the number of voters who approve of c. Formally, it is defined as*

$$score_{AV}(E,c)=|\{v\in V : c\in A(v)\}|.$$

Example 2. *Let us consider the approval election E from Example 1. To compute AV, we count the score of each candidate (i.e., how many voters approve of him or her):*

$$score_{AV}(E,c_0)=5 \quad score_{AV}(E,c_1)=1 \quad score_{AV}(E,c_2)=5 \quad score_{AV}(E,c_3)=2$$
$$score_{AV}(E,c_4)=3 \quad score_{AV}(E,c_5)=2 \quad score_{AV}(E,c_6)=2 \quad score_{AV}(E,c_7)=1.$$

AV returns the most approved candidate. In the example AV would return two tied candidates c_2 and c_0 (we break ties arbitrarily).

2.2 Preference Restrictions

Let us recall from Section 1.1 that many computational social choice problems, such as winner determination or control problems are known to be computationally hard. Preference

[2]Computational social choice theory also studies *multi-winner (committee)* elections, that is, where voting rules output a fixed size subset of the candidates (a committee). We point to the work of Faliszewski et al. (2017) for a detailed overview of multiwinner voting rules.

restrictions, i.e., restrictions on the structure of voters' ballots, are a very effective solution to circumvent the computational complexity of these problems.

2.2.1 Euclidean Preference Restrictions

Euclidean preference restrictions in the approval voting model are the main focus of this work. All the problems we solve in the subsequent chapters are structured around this concept. Unless stated otherwise, throughout this work we consider one-dimensional variants of the Euclidean models.

Let us introduce restrictions that we explore. We start with the most basic one, a predecessor of the other ones in this family of restrictions.

Restriction 1 (Dichotomous Euclidean (DE))**.** *[Elkind and Lackner (2015)] Given an election $E = (C,V)$, we say that E satisfies DE if there is a mapping ρ of voters and candidates into the real line such that for every voter $v \in V$ there exists a radius r_v, and it holds that $v = \{c : |\rho(v) - \rho(c)| \leq r_v\}$.*

The DE preference restriction is somewhat limited. Indeed, it permits for only one agent type, i.e., either the candidates or the voters, to have a radius parameter, while the other one is represented just by his or her point. We consider the DE model in one of the two variants:

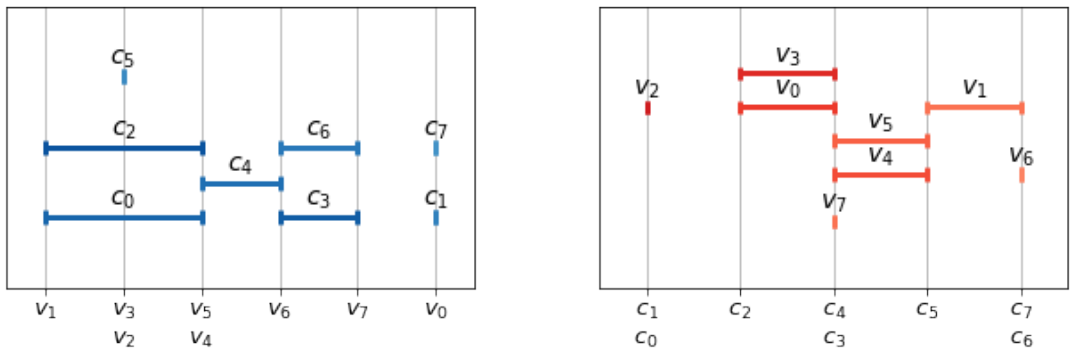
1. The voter-range model, where voters have the radius parameter. In other words, if a voter approves a candidate then he or she also approves all the closer ones.

2. The candidate-range model, where candidates have the radius parameter. We say that if a candidate is approved by a voter, then he or she is also approved by all the voters closer to him or her.

Typically, in literature these models are also known as the candidate interval and the voter interval ones, respectively. Below we provide their generalization which allows all agents to have a radius parameter, as well as their formal definitions.

Restriction 2 (Voter/Candidate Range (VCR))**.** *[Godziszewski et al. (2021)] Given an election $E = (C,V)$, we say that the voters have one-dimensional voter/candidate range preferences if for each agent $a \in C \cup V$ (i.e., for each candidate and each voter) there exists a point $x_a \in \mathbb{R}$ and nonnegative real value $r_a \in \mathbb{R}$ such that:*

$$c \in A(v) \iff |x_c - x_v| \leq r_c + r_v.$$



(a) CR election

(b) VR election

Figure 2.2 Example of elections with the CR and the VR preference restrictions. The candidates are labeled with c_i and the voters with v_i. We note that for these restrictions we replace the $\mathbb{R}$ points (as seen in, e.g., Figure 2.1a) on the x-axis, with the voters or candidates themselves (because of the CR/VR properties).

We can think of the VCR restriction as follows. For an agent $a \in C \cup V$ the point x_a describes a's ideal position in $\mathbb{R}$, representing his or hers opinion. For a candidate $c \in C$, radius r_c can be seen as c's charisma, or, in other words, their ability to reach a number of different voters (e.g., a centrist candidate possibly would have a larger reach, than an extremist one). For a voter $v \in V$, radius r_v corresponds to v's willingness to compromise, that is, how different (faraway) opinion they would still accept from their ideal one. Moreover, there are two special cases of the VCR model.

Restriction 3 (Candidate Range (CR))**.** *We say that a VCR election $E = (C,V)$, is CR if for each voter $v \in V$, their radius r_v is equal to zero.*

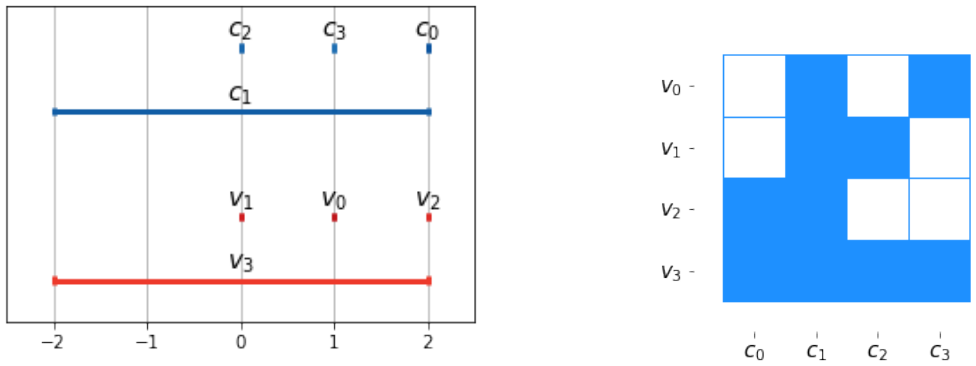
Restriction 4 (Voter Range (VR))**.** *We say that a VCR election $E = (C,V)$, is VR if for each candidate $c \in C$, their radius r_c is equal to zero.*

The CR and the VR models are equivalent to the voter interval (VI) and the candidate interval (CI) ones, both introduced by Elkind and Lackner (2015). Godziszewski et al. (2021) simply use a different terminology.

The VCR domain is larger than the CR and the VR domains combined. Below in Example 3, we discuss an election that is VCR, but neither CR nor VR[3]. Additionally, we present its visualization in Figure 2.3. We explore this topic further in Chapter 4.

Example 3. *Given an election E with the candidates $C = \{c_0, c_1, c_2, c_3\}$ and the voter's approval sets $\{c_1, c_3\}$, $\{c_1, c_2\}$, $\{c_0, c_1\}$, and $\{c_0, c_1, c_2, c_3\}$. We show that the election does*

[3]The example is based on the one from the work of Godziszewski et al. (2021).



(a) VCR representation

(b) Approval matrix

Figure 2.3 An election that satisfies the VCR property, but neither CR nor VR ones. Analyzed in Example 3.

not satisfy the VR property. If it did, then the candidates next to c_1 would have to be c_0, c_2, c_3, because they are all approved by at least one voter who also approves c_1. Clearly, it is impossible, because there can be only two candidates next to c_1 (one on each side).

Analogous reasoning works for the CR case, that is, where all the voters who approve of the same candidate have to form an interval. If this was the case, then the voters v_0, v_1, v_2 would have to be next to the v_3. Again, it is impossible as there can be only two voters next to v_3.

2.3 Integer Linear Programming

Integer Linear programming (ILP) is a common approach to solving NP-hard problems, furthermore it can be used successfully for other problems as well. To solve a problem by reduction to a linear program, we first need to define the given problem as a system of linear inequalities and an objective function. Formally, for $j \in [n]$, a single inequality consists of a set of decision variables x_j, a set of parameters a_j and a constraint b, together giving a formula: $a_1x_1 + a_2x_2 + ... + a_nx_n \leq b$ Usually, there are multiple inequalities, and we use more general notation to represent the i-th inequality:

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i.$$

Beside constraints, we define an objective function with parameters $c_1, ..., c_n$:

$$\sum_{j=1}^{n} c_j x_j.$$

Now, having defined the inequalities and the objective, the goal is to find a vector of values for the x_j variables that maximizes (or minimizes) the value of the objective function, while keeping all the inequalities satisfied. If the variables of a linear program can be fractional, then such programs can be solved in polynomial time. However, there is a special case called *(mixed) integer linear programming* where we require some of the variables to be integers. Solving integer linear programs is NP-hard. Despite this, there are available state-of-the-art linear program solvers that solve ILPs effectively. In this work we use the Gurobi solver. For a discussion of the theory of (integer) linear programming, we point to the book by Nemhauser and Wolsey (1988).

Not every problem can be directly defined as an integer linear program. However, we can apply basic techniques and tricks to help in such cases. For example, this is the case for our solution to the VCR profile detection problem described in Chapter 3. Below we present these methods.

Big-M Method. This technique is used to overcome a variety of problems, such as representing absolute or minimum/maximum values as linear inequalities. Another example of its application would be defining logical conditions (and/or) between the inequalities or using indicators (i.e., if-then logic). Such constructions are not linear in their nature, but by taking certain steps we can linearize them and, then, use them in a linear program.

Having a set of m linear inequalities with n variables represented as:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i \in [m], j \in [n], \tag{2.1}$$

the fundamental concept of the BigM method is to introduce in the inequalities an additional large constant coefficient M (hence Big in the name of the method), which is greater than the maximum value in the left-hand side of Equation 2.1. Below, in Example 4, we showcase the BigM technique to represent an *OR* condition between inequalities. To represent other non-linear constructs, we would apply the method in a slightly different way.

Example 4. *We have two inequalities and want to represent a logical OR condition between them.*

$$\sum_{j=1}^{n} a_{1j} x_j \leq b_1 \quad or \quad \sum_{j=1}^{n} a_{2j} x_j \leq b_2.$$

To do so, we introduce a large constant M and two binary variables $y_1, y_2 \in \{0,1\}$. *First we transform the inequalities:*

$$\sum_{j=1}^{n} a_{1j}x_j \leq b_1 + M(1-y_1),$$

$$\sum_{j=1}^{n} a_{2j}x_j \leq b_2 + M(1-y_2).$$

Then, we add one more constraint, binding the inequalities together:

$$y_1 + y_2 \geq 1.$$

This constraint enforces that at least one of the inequalities will be true.

The BigM method is a valuable and widely used tool. However, it has some downsides, mostly related to numerical issues. This is why most ILP solvers provide additional built-in mechanisms for applying it. For further reading, we point to the reference manual of Gurobi Optimization (2021). We use one of those mechanisms for our linear program in Chapter 3, namely an *indicator constraint*. It allows one to create constraints representing if-then logic. Formally, we have to introduce a binary variable $z \in \{0,1\}$, and then we can define an indicator constraint in the form of:

$$z = f \longrightarrow \sum_{j=1}^{n} a_{ij}x_j \leq b_i.$$

The constraint states that if variable z is equal to $f \in \{0,1\}$, then the linear constraint should hold. On the other hand, if $z = 1 - f$, then the constraint may be violated.

Chapter 3

Voter/Candidate Range Profile Detection

In this chapter we provide and evaluate an algorithm for recognizing profiles that satisfy the Voter/Candidate Range restriction. First, in Section 3.1, we discuss various approaches to preference recognition and we present selection of related work. Next, in Section 3.2, we describe our main contribution of this chapter, i.e., an algorithm based on a reduction to ILP for deciding if a given approval profile has the VCR structure. Finally, in Section 3.3 we experimentally compare the performance of our ILP based solution with the one based on a reduction to 2-SAT, for a task of recognizing the CR and VR preference restrictions.

3.1 Profile Structure Detection

To take advantage of the preference restrictions defined in Section 2.2, or any other ones for that matter, we need to design detection (recognition) algorithms. They can be used to check whether a given election has particular structural properties. Research in this area is active and successful. Provided solutions can be divided into two categories: In the first one, the recognition algorithm is based on a certain characterization of the domain. For example, Peters and Lackner (2020) show that a profile is *single-peaked on a circle* by defining a forbidden-substructure of profiles. For the *single-peaked single-crossing* domain, Elkind et al. (2020) characterize profiles by connecting them to another smaller domain. The second category of recognition algorithms is more vague and the solutions are more diverse. For example, Doignon and Falmagne (1994) provide detection algorithm for the *1D-Euclidean* domain using linear programming and Magiera and Faliszewski (2019) reduce the *top-monotonicity* recognition problem to the SAT-2CNF problem. Above-mentioned examples consider elections in the ordinal model, but, there are recognition algorithms for the approval-based elections as well. For a large group of preference restrictions, among others, the CR and the VR restrictions, the problem of recognition can be reduced to detecting

consecutive ones property (see Definition 3.1.1) in the approval matrix of a profile (Elkind and Lackner (2015)). However, for some restrictions designing a polynomial time domain recognition algorithm is not possible. Peters (2017) showed this to be true for the case of *d-Euclidean* ($d \geq 2$) restrictions in the ordinal election model.

Definition 3.1.1. *A binary matrix (i.e., a matrix consisting of zeros and ones) has the consecutive ones property (COP) if its columns can be permuted in such a way that for each row the ones are consecutive, i.e., the 1s form an interval.*

In this work we refer to the term COP more broadly, using it interchangeably for an analogous scenario where we permute the rows and look for the consecutive 1*s* in every column in a given matrix. Booth and Lueker (1976) provided a linear-time COP detection algorithm based on the PQ-Tree data structure. Below, in Proposition 1, we give an example of the VR (the CR) recognition algorithm (due to Elkind and Lackner (2015)), which utilizes the COP detection algorithm.

Proposition 1 (Elkind and Lackner (2015))**.** *There is an algorithm that given an approval election with m candidates and n voters decides if it satisfies the VR (the CR) condition in* $O(mn)$ *time.*

Proof. Let us represent the election as an approval matrix $A_{n\times m}$. We can see that for the VR property, permuting the columns of matrix A so that ones form an interval in each row is equivalent to permuting candidates so that the set of candidates approved by each voter forms an interval. For the CR case, we use our extended definition of the COP (the transposed variant), where we permute the rows so that ones form an interval in each column, which is equivalent to permuting voters so that the set of voters who approve a certain candidate forms an interval. □

In this work, our focus lies on designing an algorithm for recognition of the one dimensional VCR property. That is, having an approval matrix as input we have to decide if for each agent (i.e., for each voter and each candidate) there is a point on the real line and a nonnegative real value representing the radius so that these parameters satisfy the VCR property.

3.2 ILP-Based Algorithm

An algorithm based on integer linear programming (ILP) is an excellent choice for the first attempt at designing a detection algorithm. It provides an efficient and relatively simple solution. Furthermore, it can be used as a fast benchmark for validating future solutions.

Below we provide an ILP for the VCR profile recognition problem, together with additional preliminary steps.

Transforming the Definition of VCR into Linear Constraints

Before formulating an ILP for detecting VCR profiles, we have to redefine the definition of VCR in order to make it appropriate for an ILP solver. As a short reminder from Section 2.2.1, for the candidate set C and voter set V, we say that the 1D-VCR property is satisfied if for each candidate $c \in C$ and each voter $v \in V$:

$$c \in A(v) \Leftrightarrow |x_c - x_v| \leq r_c + r_v,$$

where x_a and r_a represent agent's point on the real line and their radius, respectively. The definition is based on the Euclidean distance which cannot be directly represented in an ILP. Equation 3.1 shows transformation of the VCR definition into linear inequalities.

$$\begin{aligned} &|x_c - x_v| \leq r_c + r_v \iff \\ &-(r_c + r_v) \leq x_c - x_v \leq r_c + r_v \iff \\ &x_c - x_v \leq r_c + r_v \ \ \vee \ \ -(x_c - x_v) \leq r_c + r_v. \end{aligned} \tag{3.1}$$

The constraint obtained from Equation 3.1 is not yet suitable for an ILP formulation because it contains a logical *OR* condition, which again cannot be directly represented in an ILP. To transform it into linear form, we use a special type of constraints provided by the Gurobi solver, i.e., indicator constraints, which allow representing if-then logic. They are based on the BigM technique, which we described in Section 2.3.

Detection Algorithm

We present our final formulation of the ILP for recognising VCR elections in Figure 3.1. The program takes as input preference matrix A, and if a solution exists it outputs every agent's position x_a and radius r_a. Additionally, we use two binary variables p and q, required by the indicator constraints.

Variables :

$$x_a \in \mathbb{R} \qquad a \in C \cup V$$
$$r_a \in \mathbb{R}^{0+} \qquad a \in C \cup V$$
$$p_{c,v} \in \{0,1\} \qquad c \in C,\, v \in V$$
$$q_{c,v} \in \{0,1\} \qquad c \in C,\, v \in V$$

Constraints :

$$x_c - x_v \leq r_c + r_v \qquad \text{for } v \in V, \text{for } c \in A(v)$$
$$-(x_c - x_v) \leq r_c + r_v \qquad \text{for } v \in V, \text{for } c \in A(v)$$

$$p_{c,v} = 1 \rightarrow x_c - x_v \geq r_c + r_v + 1 \qquad \text{for } v \in V, \text{for } c \notin A(v)$$
$$q_{c,v} = 1 \rightarrow -(x_c - x_v) \geq r_c + r_v + 1 \qquad \text{for } v \in V, \text{for } c \notin A(v)$$
$$p_{c,v} + q_{c,v} \geq 1 \qquad \text{for } v \in V, \text{for } c \notin A(v)$$

Figure 3.1 An ILP for detecting the VCR property. Variables x_a and r_a represent agent's position and radius, and variables $p_{c,v}$ and $q_{c,v}$ are required to use indicator constraints.

3.3 Performance Comparison of ILP and SAT Based Solutions for COP Detection

In this section we propose a small extension of our ILP, so it can be used to detect the VR and the CR properties. Then, we compare it with an alternative solution, which is based on a reduction to the 2-SAT problem, due to Fitzsimmons and Lackner (2020).

Extending the VCR recognition ILP

We know that the VR and the CR properties are special cases of the VCR restriction. To be specific, they require either all the candidates or all the voters to have radius equal to zero, respectively. The extension is straightforward: We need to introduce additional constraints that require each candidate's or each voter's radius to be zero. For the VR case it is:

$$r_c = 0 \quad \text{for } c \in C,$$

and for the CR:

$$r_v = 0 \quad \text{for } v \in V.$$

Comparison Experiment

The 2-SAT based solution is an algorithm for the consecutive ones property detection in a matrix. From Proposition 1 we know it is equivalent to detecting the VR property in an approval matrix of an election. That said, we expect the ILP solution to be slower than the 2-SAT one, as the latter one is specialized in one task only, that is detecting the COP, and there are classic polynomial-time algorithms for 2-SAT (see the work of Aspvall et al. (1979)). On the other hand, the ILP solution is more general, as it outputs not only a VR certificate, but also a mapping of agents to points and radii. We design a simple experiment to find out how much faster is the 2-SAT solution.

We conduct the experiment in the following steps:

1. We consider three elections sizes. Having m candidates and n voters, election dimensions are: $(m,n) \in \{(15,15),(20,20),(40,40)\}$.

2. We generate 100 random VCR profiles for each election size (generation details are discussed in Section 4.1.2).

3. For both algorithms, we measure the time of model creation and optimization to detect the VR property. To minimize external factors on time measurement, we repeat this step three times, and pick the smallest result.

Results. Let us analyze the results shown in Figure 3.2 and Table 3.1. Each subplot presents results for a given election size. Algorithms are represented on the y-axis, and measured detection time on the x-axis in seconds. Furthermore, each point is colored based on the detection result. 2-SAT based algorithm is significantly faster and more consistent. It is anywhere between 10 and 100 times faster. These conclusions are reflected in the metrics presented in Table 3.1. Additionally, we have performed one-off measurements for some larger elections (i.e., $80C \times 80V$ and $100C \times 100V$), for which detection time differences were even bigger.

3.4 Conclusions

The key takeaway from this chapter is that recognition algorithms are crucial part of creating successful preference restrictions. We might create a preference restriction with extremely

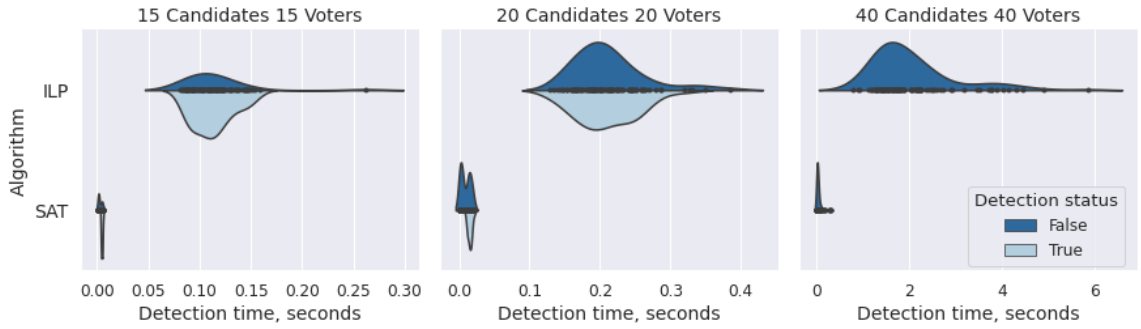


Figure 3.2 VR (COP) Detection time comparison between ILP and SAT. Results are divided into 3 subplots based on the election size. Categories, that is algorithms used (i.e., ILP or SAT) are described on y-axes. On x-axes detection time is shown in seconds. Additionally, we distinguish detection outcome (i.e., positive or negative) with different colors. Precise metrics are included in Table 3.1.

Algorithm	Election Size	mean	std
ILP	15 Candidates 15 Voters	0.113	0.024
	20 Candidates 20 Voters	0.211	0.047
	40 Candidates 40 Voters	2.055	0.924
SAT	15 Candidates 15 Voters	0.004	0.001
	20 Candidates 20 Voters	0.012	0.006
	40 Candidates 40 Voters	0.037	0.063

Table 3.1 Metrics for the VR detection time, grouped by algorithm type and election size.

desirable properties, but it would not be used in practice until the invention of an efficient recognition algorithm. Now, focusing on the VCR restriction and the 2-SAT approach to recognition problems, we showed how fast 2-SAT based recognition algorithm can be. However, it is unlikely that we will create one for the VCR restriction with our current knowledge of the domain. That is, until the VCR domain is not characterized by some combinatorial properties. Nonetheless, there might be a whole new approach to the VCR detection. We leave the problem open.

Chapter 4

Voter/Candidate Range Profiles

In this chapter we explore the Voter/Candidate Range domain, describe some of its properties, and compare it with the Voter Range and Candidate Range ones. We already know that the VCR domain is larger than the VR and CR ones combined (see Section 2, Example 3). However, we would like to know how much bigger it actually is, and if we can significantly benefit from further development of algorithms based on the VCR model.

4.1 Domain Analysis

We have performed several experiments, in each of them we check the probability with which a random election falls within a certain domain, we consider the VCR domain and its variants. For the first experiment we focus on the small size elections and only observe the overview of the properties. For the second one, we do a more thorough examination. We check larger profiles, and compare the properties in detail. In every experiment we use the detection algorithm described in Section 3.2.

For the sake of text clarity and more explicit naming, we introduce additional notations to describe some special cases of the VCR preference restriction.

Definition 4.1.1. *A profile P is **Truly VCR (TVCR)**, if P satisfies the VCR property and does not satisfy either of the VR and CR properties.*

Definition 4.1.2. *A profile P is **Fully COP (FCOP)**, if P satisfies both of the VR and CR properties (see Definition 3.1.1, for a reminder of the COP notion).*

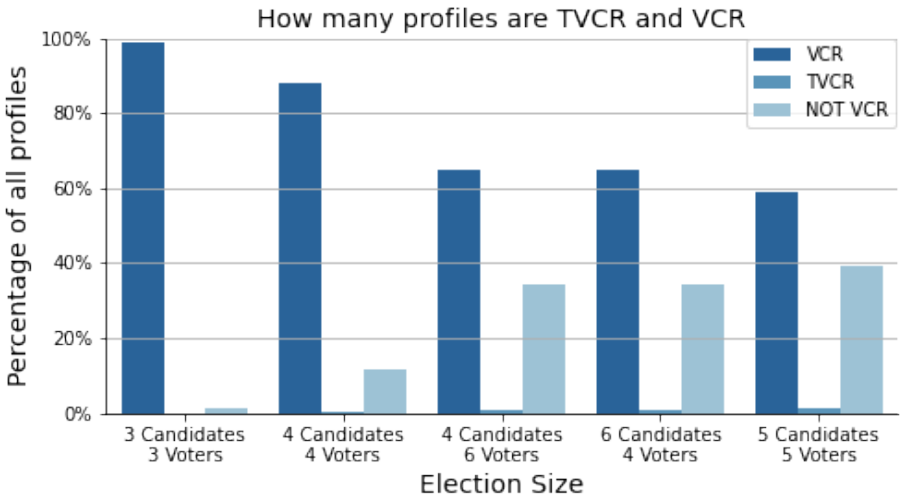


Figure 4.1 Relative number of the profiles with a given property (VCR and TVCR), out of all possible ones for a given election size, from Experiment 1

4.1.1 Experiment 1: How Many Profiles Are TVCR or VCR?

For this experiment we performed a complete search of the domain. That is, we generated every possible profile of a given size and checked if it has the VCR property, and, if so, we checked again if it were the TVCR variant. We focused on the small size of elections. Having elections of M candidates and N voters, we considered $(M,N) \in \{(3,3),(4,4),(5,5),(4,6),(6,4)\}$.

The results are presented in Figure 4.1. They show that the VCR domain is relatively large. However, for these small election sizes, only a limited number of profiles are Truly VCR (i.e., neither VR nor CR). For example, looking at the two smallest elections, for the 3 candidates, 3 voters case, there is no TVCR profiles at all. For the 4 by 4 case, there is only 96 of them, out of 2^{16} profiles. Another observation is a decrease in the number of the VCR profiles, in favor of the non-VCR ones, with the increase of the election size (e.g., for the 5 candidates, 5 voters case, only about 60% of profiles are VCR).

4.1.2 Experiment 2: How Many Profiles Are TVCR, CR, VR or FCOP?

In this experiment, we only consider the VCR profiles. We want to find out how their numbers are distributed across different variants of the VCR restriction. We generated a suitable number of random VCR profiles and checked how many of them are neither CR nor VR (TVCR). We grouped the results into three additional categories, besides the TVCR one, that is, VR and CR (FCOP), VR and not CR, CR and not VR.

We employ several strategies to generate a large number of profiles with distinct characteristics. We use various statistical cultures (distributions) to pick random points and radii representing the candidates and voters. Furthermore, we choose the random distribution independently for the position and radius parameters, and independently for the candidates and voters. We describe these categories in detail below. For each category pair (in total, 20 pairs) we generated 1000 elections with 20 candidates and 20 voters. Then we repeated the experiment for the elections consisting of 40 candidates and 40 voters.

Position Parameter

We consider four ways of generating the points, representing the positions of the candidates and the voters:

1. The *Uniform Candidate Positions, Uniform Voter Positions (UCP / UVP)* model, where the points for both, the candidates and the voters are selected uniformly at random from the $[-10, 10]$ interval.

2. The *Asymmetric Gauss Candidate Positions*, *Asymmetric Gauss Voter Positions (GCP / GVP)* model, where we take 70% of points from a Gaussian distribution with mean value -3 and standard deviation 1.8, and the remaining 30% come from a Gaussian with mean 3 and the same standard deviation. This model is inspired by the 2 dimensional model from the work of Godziszewski et al. (2021). The goal is to simulate a society with a large majority, and a minority with the opposite opinion.

3. The *Uniform Candidate Positions, Asymmetric Gauss Voter Positions (UCP/GVP)* model, it is a combination of the previous two models, where the candidates' points are selected uniformly at random, and for the voters we use the asymmetric Gaussian model. We use the same parameters for both of the distributions as above.

4. The *Asymmetric Gauss Candidate Positions, Uniform Voter Positions (GCP/UVP)* model, is a reversed version of the previous model, with the distributions swapped.

Radius Parameter

In this category we consider five ways of generating the radii for the candidates and the voters. We use the following models:

1. The *Gauss Candidate Radii, Gauss Voter Radii (GCR/GVR)* model, where we choose the radii from the Gaussian distribution with mean value 1.5 and standard deviation 0.5. We use the same distribution for the candidates and the voters.

2. The *Small Uniform Candidate Radii, Gauss Voter Radii (SUCR/GVR)* model. We pick the radii for the candidates uniformly at random from the interval $[0.7, 1.2]$, and for the voters we use the same Gaussian as in the previous model (with mean 1.5 and standard deviation 0.5).

3. The *Gauss Candidate Radii, Small Uniform Voter Radii (GCR/SUVR)* model. In this model we swap the distributions used in the previous one, that is, the radii for the candidates are picked from the Gaussian and the radii for the voters from the uniform distributions.

4. The *Small Uniform Candidate Radii, Small Uniform Voter Radii (SUCR/SUVR)* model. We pick the radii for both the candidates and the voters uniformly at random from the interval $[0.7, 1.2]$.

5. The *Large Uniform Candidate Radii, Large Uniform Voter Radii (LUCR/LUVR)* model. We pick the radii for both the candidates and the voters uniformly at random from the interval $[0, 3]$.

At first, it might be surprising why we do not include a constant radius as one of the distributions. We disregard this possibility because, then, all the profiles would be VR and CR (FCOP). Below in Proposition 2, we show why.

Proposition 2. *If every agent (candidate/voter) in a VCR profile has the same constant radius, then this profile satisfies both the VR and the CR properties (it is FCOP).*

Proof. Consider a VCR election $E = (C,V)$, where $C = \{c_1, ..., c_M\}$ and $V = \{v_1, ..., v_N\}$, and a constant radius R. An agent $a \in C \cup V$ is represented by a point x_a and a radius r_a, and in this special case every agent has the same radius $r_a = R$. We construct a CR election $E' = (C', V')$, where each voter $v' \in V'$ has radius $r_{v'} = 0$ and each candidate $c' \in C'$ has radius $r_{c'} = 2R$. We will show that these elections are identical. From the VCR definition, we know that for each candidate $c \in C$, if c is approved by a voter v_i, then

$$|x_c - x_{v_i}| \leq r_c + r_{v_i} \iff$$
$$|x_c - x_{v_i}| \leq R + R$$

Similarly, for each candidate $c' \in C'$ if c' is approved by a voter v'_i, then

$$|x_{c'} - x_{v'_i}| \leq 2R + 0$$

We showed that the elections E and E' are equivalent. Analogous reasoning works for the VR case, where each candidate has radius zero, completing the proof. □

Results

First, in Figure 4.2, we present the visualization of our example profiles approval matrices and how their visual structure is impacted by the generation parameters. Each subplot represents a different combination of the generation parameters pairs. Columns correspond to the radius candidate/voter models, and rows to the position candidate/voter models.

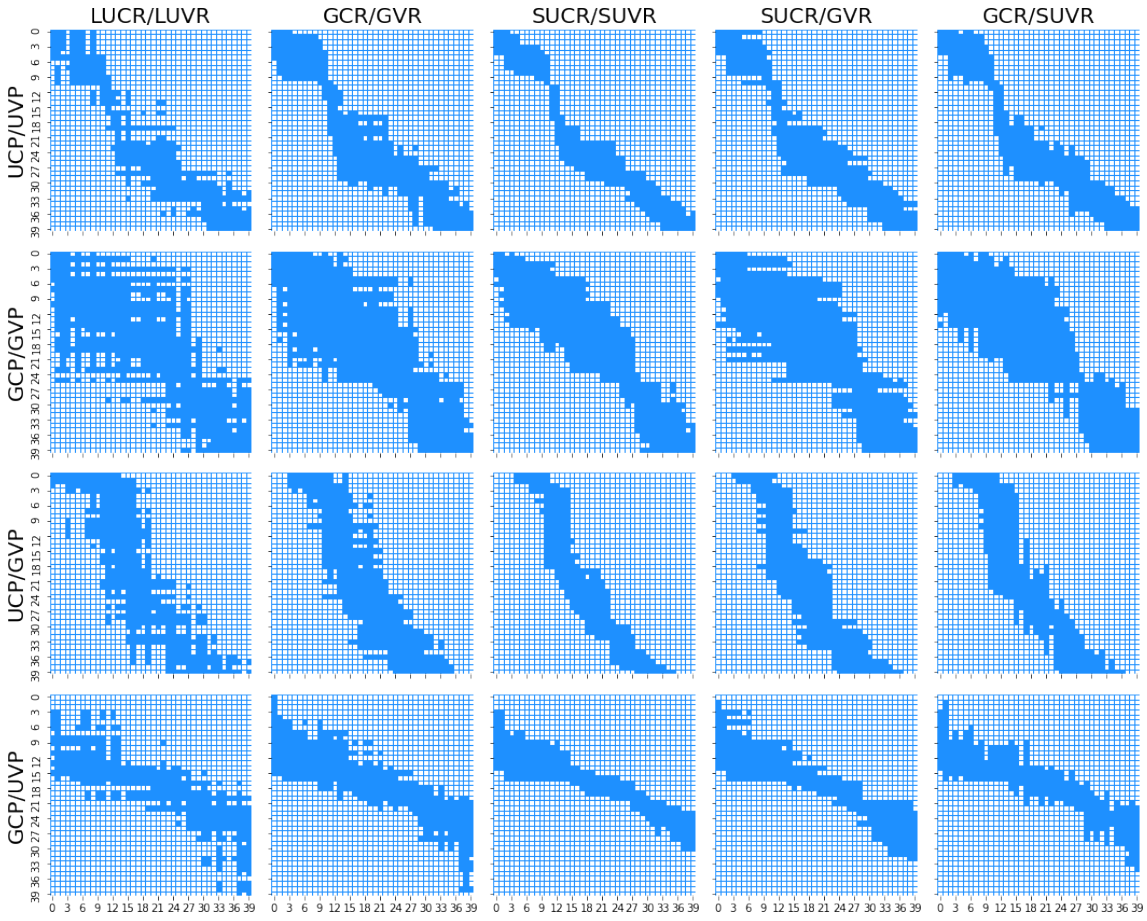


Figure 4.2 Examples of approval matrices for the elections of 40 candidates and 40 voters, from Experiment 2. Each subplot is a visual representation of an approval matrix of a selected profile. Solid color corresponds to 1 and white to 0 in the cells of the matrix. Both the candidates (i.e., columns) and the voters (i.e., rows) are ordered by their point in the VCR model. To create a grid, we use radius candidate/voter models for the columns, and position candidate/voter models for the rows, e.g., top-left subplot is visualization of a profile generated with the LUCR/LUVR radius model and the UCP/UVP position model.

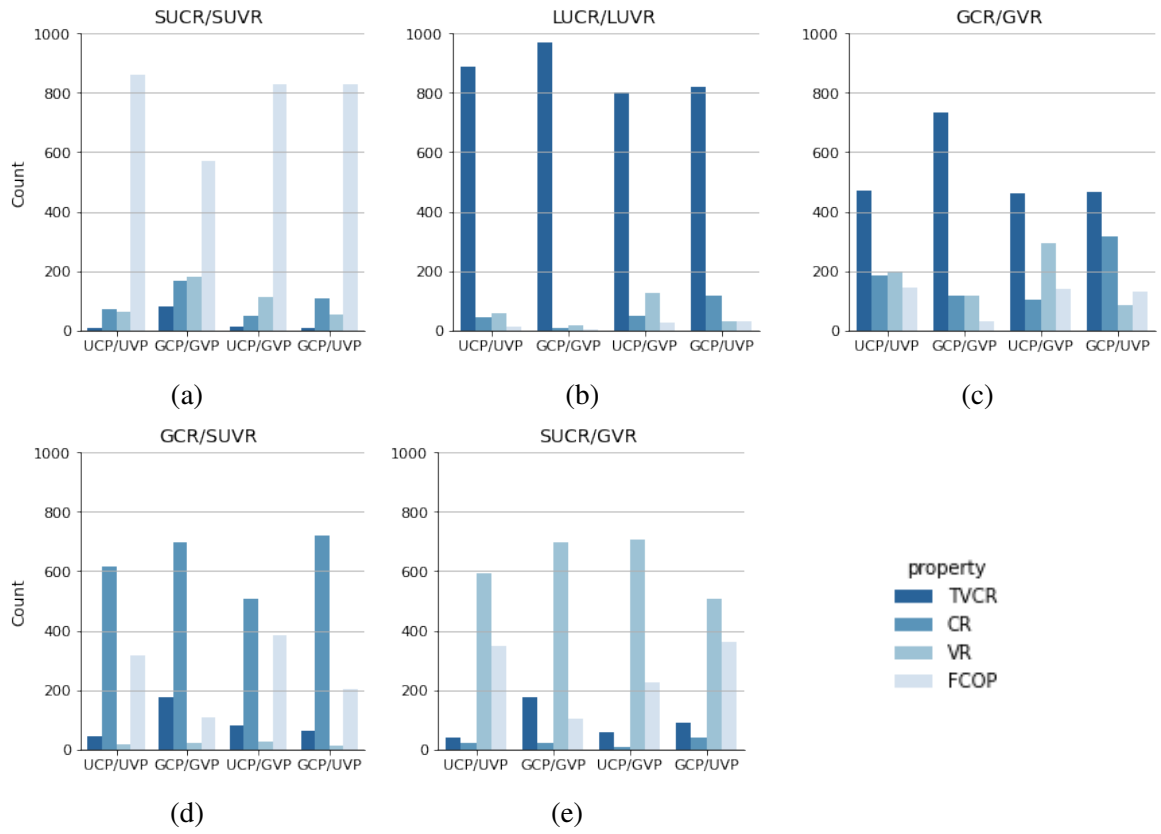


Figure 4.3 The numbers of profiles with a given property, for elections of 20 candidates and 20 voters, from Experiment 2. Results are grouped by the parameters used for the profile generation. Each subplot corresponds to the selected radius candidate/voter model. Position candidate/voter models are represented on the x-axis.

Before we further describe the results of the experiment, we need to observe trends in the visualization and explain some basic ideas. Based on the visualizations, we know that the VCR profiles produce semi-structured approval matrices. That is, the ones in an approval matrix create roughly continuous blocks (intervals), they form a cluster. However, in certain circumstances, these blocks might have some irregularities (i.e., a 0 in a sequence of 1's). The perfect example of this phenomenon are profiles from the first column in Figure 4.2, that is, profiles generated using the LUCR/LUVR radius model. We can spot the "holes" in otherwise regular structure of 1's. Another important trend in these visualizations is the general direction in which the block structure is skewed. For profiles in the first and second row of Figure 4.2 (i.e., the UCP/UVP and the GCP/GVP models), the skew is on a diagonal from the top-left to the bottom-right corner. However, profiles from the last two rows (i.e., the UCP/GVP and the GCP/UVP models), while maintaining general diagonal direction,

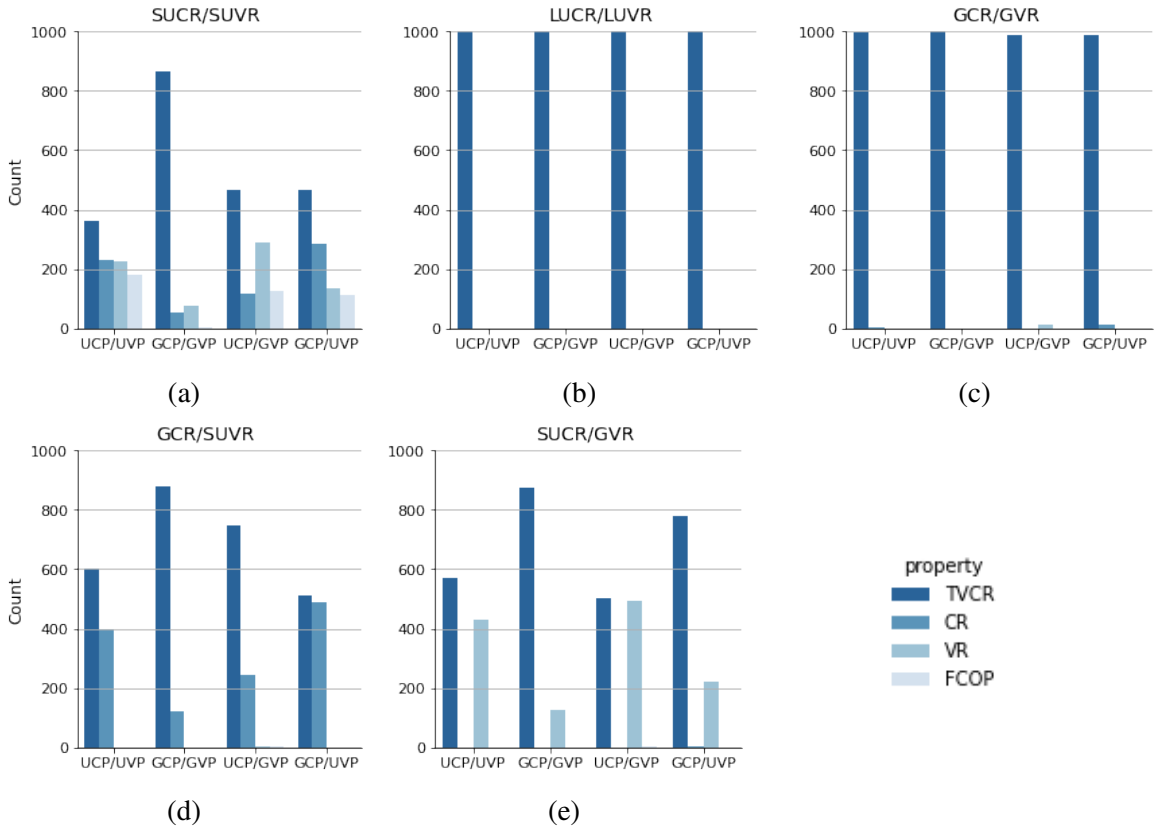


Figure 4.4 The numbers of profiles with a given property, for elections of 40 candidates and 40 voters, from Experiment 2. Results are grouped by the parameters used for the profile generation. Each subplot corresponds to the selected radius candidate/voter model. Position candidate/voter models are represented on the x-axis.

have a significant shift. The UCP/GVP ones are skewed vertically, opposite to the GCP/UVP ones, which are skewed horizontally.

In the following discussion we tie these observations with the aggregated results of the experiment visualized in Figures 4.3 and 4.4. In Figure 4.3 we show how many of the generated profiles satisfy a given preference restriction. To understand the results better, we split the collected data into groups. Each subplot represents a subset of results for the selected radius candidate/voter model, and position candidate/voter models are represented on x-axes. Figure 4.4 is almost identical, the only difference is the first one shows the results for the 20 candidates and 20 voters elections, whereas the second one describes the larger elections of 40 candidates and 40 voters.

Election Dimensions Our results for two election sizes have one crucial difference. Looking briefly at Figure 4.3 (20 candidates, 20 voters) and Figure 4.4 (40 candidates, 40

voters), we immediately see that in the larger elections the TVCR property is much more frequent. Election sizes in the experiment are relatively small when compared to real world scenarios. We can expect for this trend of increased number of TVCR profiles to continue for even larger elections. This result is very promising, as it strongly encourages further development and research related to the VCR domain.

In the subsequent discussion we will focus mostly on the results for smaller elections from Figure 4.3. We do so because there the differences between the parameters are more pronounced.

Position Choice. Let us analyze Figure 4.3c, which presents histograms for the GCR/GVR radius model and various candidate/voter position models. We consider this model to be our baseline scenario, because it has the most balanced distribution of the numbers of profiles between the selected preference restrictions. We compare the position models in pairs. Starting with the UCP/UVP and the GCP/GVP models, the latter one has noticeably more TVCR profiles. It can be easily explained by looking at the respective approval matrices in Figure 4.2 (2nd column, 1st and 2nd row). The profiles generated from the GCR/GVR model have higher count of approvals, which in turn increases the likelihood of any irregularities in the approvals block structure, which we directly link to the TVCR property.

Now, turning our attention to the second pair of the position models (still for the same radius model), the UCP/GVP and the GCP/UVP, we see that these are two perfectly symmetrical models. We only explain the first one, and keep in mind that the second one is simply a reversed scenario. Looking at the approval matrices in Figure 4.2 (3rd row), we notice that the blocks of approvals are skewed vertically, which is associated with the VR property, i.e., the approvals form intervals in each row of the matrix. On the contrary, in the second model (Figure 4.2, 4th row) the blocks are skewed horizontally, which links to the CR property.

Radius Choice. From Figure 4.3, it is clear that the choice of the radius candidate/voter model can significantly reinforce or subside trends in the distribution of the numbers of profiles with a selected property. Figure 4.3c was our baseline scenario when analyzing influence of the position candidate/voter models. Now, let us compare this subplot with the other ones. For example, the mixed radius models, that is, the GCR/SUVR (Figure 4.3d) and the SUCR/GVR (Figure 4.3e) models. Looking at the histograms for these models, we observe significant increase in the number of profiles with the CR and the VR properties, for the GCR/SUVR and the SUCR/GVR respectively. The core idea explaining this phenomenon is exactly the same as the one that we already described, when explaining the UCP/GVP and the GCP/UVP position models. Looking at approval matrices in Figure 4.2 (4th and

5th columns), the GCR/SUVR and the SUCR/GVR models output more consistent approval blocks (without 0's between the 1's), resulting in a large decrease of the number of the TVCR profiles. At the same time, these models work in synergy with the UCP/GVP and the GCP/UVP position models, and the created approval structures are skewed either horizontally or vertically, increasing the number of the VR or the CR profiles.

The striking difference between histograms for the LUCR/LUVR (Figure 4.3b) and the SUCR/SUVR (Figure 4.3a) radius models is again related to the inconsistencies in the approval blocks. They are easily observed in approval matrices visualized in Figure 4.2 (1st and 3rd column). Smaller radius range (the SUCR/SUVR) produces much more structured profiles, hence large number of the FCOP profiles. In contrast, large radius range (the LUCR/LUVR) outputs irregular approval blocks in the profiles, increasing the number of the TVCR profiles.

Summary. Most of the conclusions we draw are in line with the basic intuition. The results are very promising: We showed that the VCR domain can be significantly larger than the VR and the CR ones combined. We hope, this will encourage further development and research related to the VCR domain.

4.2 Transforming a TVCR Profile into a CR or a VR One

A natural question that arises from the previous experiments is: How different are the TVCR profiles from the VR or the CR ones? In other words, to what extent do we need to change a TVCR profile for it to become a VR or a CR one. To answer this question, we design yet another experiment.

4.2.1 Experiment 3: How Different Is a TVCR Profile from CR/VR Ones?

In this experiment we partially reuse the strategy for profile generation from the previous one. We narrow down our experiment parameters to only two radius generation methods, that is, the GCR/GVR and the LUCR/LUVR models. Nonetheless, we keep the position models unchanged and explore all four of them. Naturally, we only pick the TVCR profiles. To be exact, we use 400 of them for each category pair (in total, 8 pairs). Due to the limited computational resources, we only use profiles consisting of 15 candidates and 15 voters. Having a TVCR profile as input, we search for a minimal number of agents that we have to remove from this profile for it to become a VR or a CR one. To achieve more meaningful

results we categorize experiment parameters and describe them below.

Agent Deletion Parameter. We consider three methods to transform a TVCR profile into a VR or a CR one:

1. The *Candidate Deletion (CD)* method, where we search for a minimal number d, such that after deleting d candidates from the profile it becomes a VR or a CR profile.

2. The *Voter Deletion (VD)* method, which is similar to the CD method, but this time we delete d voters instead.

3. The *Candidate and Voter Deletion (CVD)* method, where we combine the two previous methods, and we simultaneously delete d_c candidates and d_v voters.

Additionally, we put an upper bound on the number of removed agents. For the CD and the VD methods the limit is 4, that is, we only try to remove 4 agents at most. If we fail to find a solution, we use a number 5 as a result, which we interpret as 5 or more agent deletions required to succesfully transform a profile. For the CVD method, the limit is 4 per each agent type, i.e, the maximum number of deleted agents is 4 candidates and 4 voters.

Transformation Result Parameter. Apart from choosing what type of agent (candidate or voter) to delete, for each profile we collect the transformation results in three groups. By the transformation result we mean the lowest possible cost of the transformation, i.e., the minimal number of agent deletions for a TVCR profile to become a VR or a CR one. We collect the following information:

1. The *Candidate Range (CR)*, where the cost is a number of deletions required to get a CR profile.

2. The *Voter Range (VR)*, where the cost is a number of deletions required to get a VR profile.

3. The *Candidate Range or Voter Range (CRVR)*, where the cost is a minimum from the previous two groups.

Results

We divide the observations in two groups based on the agent deletion parameter. First, we analyze the results for the CD and VD methods, and later we discuss the CVD method by itself. We use similar ideas and concepts as in previous experiments to explain the results.

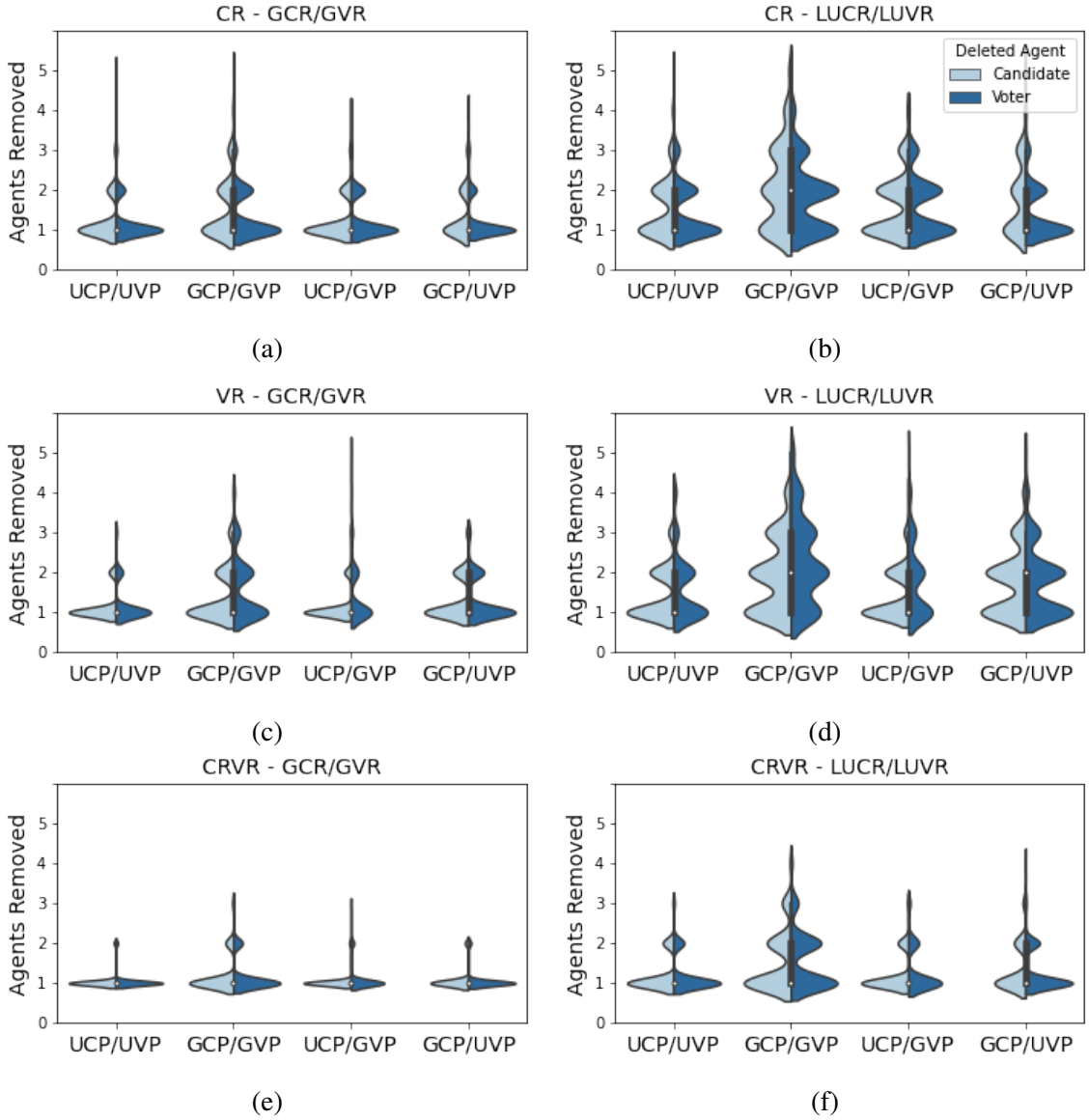


Figure 4.5 Required number of agents to remove, to transform a TVCR profile into a CR or a VR one, from Experiment 3. Each subplot consists of 4 violin plots. A violin plot shows the distribution of quantitative data, and uses kernel density estimation for the visualization. In our case, violins are additionally split in half, i.e., the left part (light color) shows the number of candidates deleted, and the right part (dark color) the number of voters deleted. Results are grouped by the experiment parameters. To create a subplot gird, we use radius candidate/voter models for the columns, and transformation result parameters for the rows. E.g., top-left subplot is a visualization of the results for the GCR/GVR radius model and the CR property. Furthermore, position candidate/voter models are represented on the x-axis.

We present the results for the first group in Figure 4.5. The figure is split into six subplots. Each one of them is a visualization for different subset of results, divided by two parameters.

The first category is the radius candidate/voter model used to generate the profiles, and the second one is based on the transformation result parameter, i.e, the CR, VR and CRVR types. A single subplot shows the distribution of a required number of agents to remove (a violin plot), to transform a TVCR profile into one with a given property.

Continuing, let us analyze Figure 4.5a and Figure 4.5c. They both show results for the GCR/GVR radius model, the first one corresponds to the CR and the second one to the VR properties. We notice that they are somewhat symmetric. For example, for the GCP/GVP position model, to obtain the CR profile it is preferable to delete voters. However, if we look at the VR case, it is better to delete candidates (by writing preferred or better, we mean, it requires fewer agent removals). Another interesting observation in these plots are the violins for the UCP/GVP and the GCP/UVP models. They are, in essence, swapped between the CR and the VR cases. To explain this, we go back to the respective histograms from Experiment 2 in Figure 4.3c. For the UCP/GVP model, the majority of the profiles are VR (not counting the TVCR ones). This suggests that the TVCR profiles from this model are more similar to the VR ones than to the CR ones. This idea is reflected in the corresponding violin plots (Figure 4.5c). For the GCP/UVP position model we see the reversed scenario. In the histograms in Figure 4.3c the majority of profiles are CR, and the violin plots (Figure 4.5a) show that it is better to delete voters to obtain the CR profiles. We observe the same phenomenon for the LUCR/LUVR radius model in Figure 4.5b (the CR) and in Figure 4.5d (the VR). There is nothing unexpected in the results for the CRVR property (Figure 4.5e and Figure 4.5f). Having said that, they give us a promising conclusion, that if we are indifferent whether we obtain the CR or the VR profile, we can greatly reduce the required number of agent deletions.

Let us now discuss the second part of the experiment, that is the results for the CVD method, in which we delete both the candidates and the voters simultaneously. We create plots in Figure 4.6 in the same way as in already described Figure 4.5. The only difference is conceptual. Previously the violins were split to distinguish between removed agent type, i.e, left half (light color) represented candidates and right half (dark color) showed voters. Now, the halves still represent the number of deleted agents, but the process of deletion is simultaneous for candidates and voters. We note that if all the values in the distribution are identical, then the violin plot is replaced with a single horizontal line. In most cases it is enough to delete just one candidate and one voter to obtain a CR or a VR profile. Besides that, we explain why it appears that we need to delete more voters than candidates. It comes merely from implementation nuances. To find a number of required removals, we iterate thorough pairs of all (d_c, d_v) element combinations. That is we start by deleting one candidate and one voter, if there is no solution, then we try to delete one candidate and two voters, and

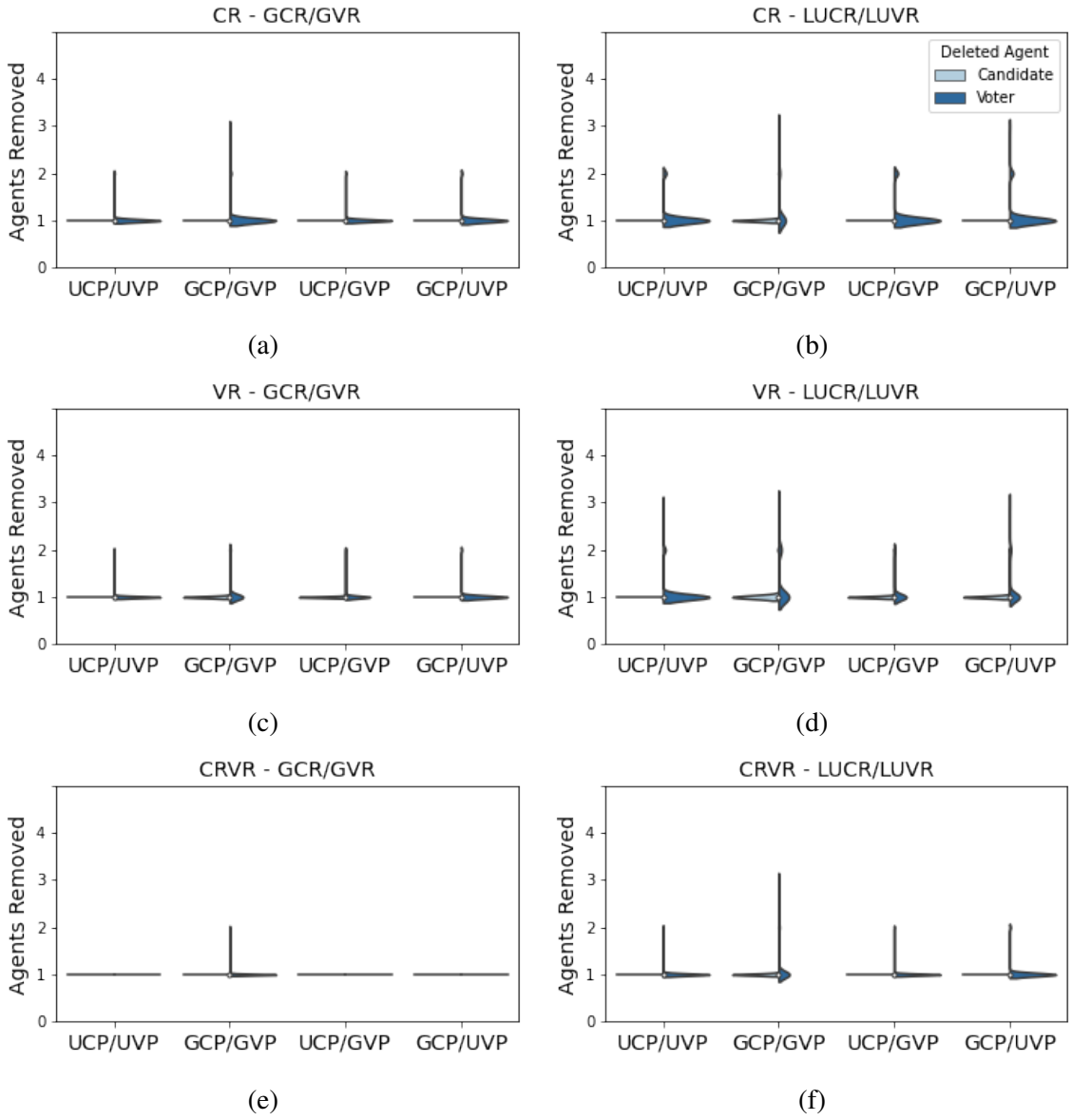


Figure 4.6 Required number of candidates and voters to remove, to transform a TVCR profile into a CR or a VR one, from Experiment 3. The plot is structured in the same way as in Figure 4.5, except one difference. Split in half violins no longer represent independent deletion of the candidates or the voters, but correspond to simultaneous deletion of both. If there is no violin (horizontal line), it means that all the values were identical.

only then if there is still no solution, we try two candidates and one voter, followed by 2 candidates and 2 voters, and so on.

In addition to the visual representation of the results, we provide statistical metrics of the results, i.e., the mean, standard deviation, minimum and maximum. Their values are grouped by the experiment parameters, in the same way as in the graphs, except the data is aggregated

for all the position models. We present them in Table 4.1 for the first part of the experiment, and in Table 4.2 for the second one.

Summary. First, we showed how results from Experiment 3 further confirm our conclusions from Experiment 2. Moreover, our results provide strong basis, to consider transforming a TVCR profile into a CR or a VR one, if it were required by other applications.

Radius Model	Property	Deleted Agent	mean	std	min	max
GCR/GVR	CR	Candidate	1.366	0.633	1	5
		Voter	1.258	0.506	1	4
	CRVR	Candidate	1.089	0.304	1	3
		Voter	1.092	0.297	1	3
	VR	Candidate	1.256	0.513	1	4
		Voter	1.373	0.623	1	5
LUCR/LUVR	CR	Candidate	1.788	0.903	1	5
		Voter	1.587	0.737	1	5
	CRVR	Candidate	1.341	0.593	1	4
		Voter	1.336	0.567	1	4
	VR	Candidate	1.656	0.792	1	5
		Voter	1.851	0.935	1	5

Table 4.1 Metrics for number of deleted agents, from Experiment 3.1

Radius Model	Property	Deleted Agent	mean	std	min	max
GCR/GVR	CR	Candidate	1.000	0.000	1	1
		Voter	1.021	0.146	1	3
	CRVR	Candidate	1.000	0.000	1	1
		Voter	1.001	0.025	1	2
	VR	Candidate	1.002	0.043	1	2
		Voter	1.022	0.148	1	2
LUCR/LUVR	CR	Candidate	1.002	0.043	1	2
		Voter	1.107	0.332	1	3
	CRVR	Candidate	1.001	0.035	1	2
		Voter	1.029	0.171	1	3
	VR	Candidate	1.008	0.090	1	2
		Voter	1.127	0.357	1	3

Table 4.2 Metrics for number of deleted agents, from Experiment 3.2

Chapter 5

Control Problem

In this chapter we demonstrate how a certain control problem that is known to be NP-hard for unrestricted preferences becomes polynomial-time solvable for the VCR elections. First, in Section 5.1, we discuss variants of the control problem. Additionally, we present two existing polynomial-time algorithms for a control problem, for elections in the VR and the CR domain. Later in Section 5.2, we describe our solution for the VCR elections, and we show that it unifies the two aforementioned algorithms for the VR and the CR cases.

5.1 Background

Control problems were first introduced by Bartholdi et al. (1992), and later extended by Hemaspaandra et al. (2009). Election control refers to an act of manipulating the structure of an election in a certain way. We assume that there is some person or authority who is in charge of organizing the election. We refer to them as the *election chair*. The chair has a goal to change the outcome of the election. Formally we call it the *control type*, and we distinguish two types. In the first one, *constructive* control (CC), the chair's goal is to ensure a victory of a selected candidate. In the second one, *destructive* control (DC), is opposite, that is, the chair's intention is to prevent a selected candidate from winning. Furthermore, control problems are grouped by the method used to change the structure of an election, which we call the *control action*. Usually in the literature, for each control type (i.e, CC or DC) we consider four possible actions, that is, adding candidates (AC), deleting candidates (DC), adding voters (AV) or deleting voters (DV). For further reading, we refer the reader to a chapter by Faliszewski and Rothe (2016), which provides a complete overview of control problems as well as closely related bribery problems.

The algorithms that we discuss in the following sections, consider constructive control by deleting voters (CC-DV) where for the winner determination, we use the approval voting rule (see Definition 2.1.1). Below, we give a formal definition of our problem.

Definition 5.1.1. *We are given a set of candidates C, a set of voters V, a nonnegative integer k, and a selected candidate $p \in C$. In the constructive control by deleting voters problem for approval voting (Approval-CC-DV), we ask whether there exists a set $V' \subseteq V$, such that p is the unique winner of the approval voting election $E = (C, V \setminus V')$, where $|V'| \leq k$.*

Now, we define some basic concepts and terminology that will help us to explain the algorithms in the following sections. First, we introduce an alternative representation of an agent in a VCR profile.

Definition 5.1.2. *In the VCR model, every agent a is represented by their point x_a and radius r_a. We define a's range beginning and range end as:*

$$b(a) = x_a - r_a,$$

$$e(a) = x_a + r_a.$$

Additionally, to represent the AV-score difference between two candidates, we use:

Definition 5.1.3. *Having an approval election E and two candidates a and b. Under the approval-voting scoring rule, we say that:*

$$scoreDelta_E(a,b) = score_{AV}(E,a) - score_{AV}(E,b).$$

However, we note that whenever election E is clear from the context, we use a more concise notation, that is, $scoreDelta(a,b)$.

5.1.1 Constructive Control by Deleting Voters For VR Elections

We now discuss an algorithm for Approval-CC-DV for VR elections due to Faliszewski et al. (2011) (we refer to it as the FHHR algorithm). It is the first paper to consider solving control problems under preference restrictions. The restriction that they consider is the single-peaked one. However, in the approval elections it is virtually identical to the voter range model (i.e., candidates' radii are zero). We only present a brief description of the algorithm that explains the core idea and some necessary concepts. For the detailed explanation, proofs, and examples we point to the original paper of Faliszewski et al. (2011).

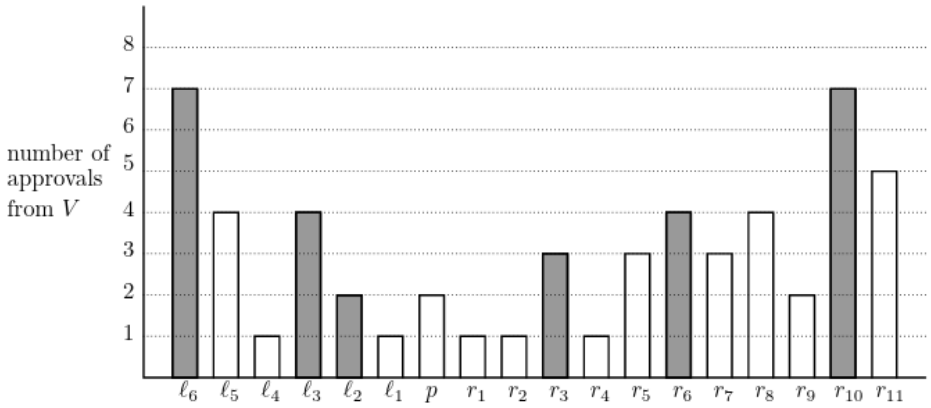


Figure 5.1 Illustration of the CC-DV for the VR elections from Faliszewski et al. (2011). Bar chart of approvals from V. The candidates are in the VR order, and each candidate is labeled with respect to whether he or she is left or right of the candidate p. Additionally, the "dangerous" candidates are marked with shaded bars.

The algorithm deploys a greedy strategy. In the explanation we follow the example from Figure 5.1. Let us consider the CC-DV problem under the approval voting rule. Having a set of candidates C in the VR order, a set of voters V, a designated winner candidate p, and a maximum number of voter deletions k, to find a solution (if it exists) we take the following steps: First, we group the candidates based on their position relative to p, i.e., left or right from p. Then, we proceeded to selecting the dangerous candidates (opponents). The first opponent to the right of p is the leftmost candidate from the right group, that gets at least as many approvals as p (this is r_3 in Figure 5.1). Every next right dangerous candidate is the candidate that gets strictly more approvals than the previous dangerous candidate (these are r_6, and then r_{10} in Figure 5.1). We take analogous steps for the candidates on the left. From this point on, we try to defeat dangerous candidates. We process them one at a time in each iteration. We start with the closest right opponent (i.e., r_3), and call him c_R. Now we have to choose which and how many voters to delete. We consider only a subset of voters, such that each voters' range beginning is in the open-closed interval between p and c_R, i.e., $\widehat{V} = \{v \in V \,|\, b(v) \in (p, c_R]\}$. Next we try to delete $scoreDelta(c_R, p) + 1$ rightmost voters from $\widehat{V}$, that is, the ones with the highest range end (i.e., $e(v)$). If we fail, or we exceed the voter deletion limit k, the solution does not exist. We repeat these steps until candidate p becomes a unique winner (or we fail in the process). We recalculate the approvals and the dangerous candidates set after each iteration. Once we reached the point where all the dangerous candidates are in the left group, then we simply "flip the universe", and consider them to be the right ones.

5.1.2 Constructive Control by Deleting Voters For CR Elections

The next algorithm is due to Magiera and Faliszewski (2017) (we refer to it as the MF algorithm). They study multiple election types, but we discuss only the approval elections case. They consider control problems for the single-crossing preferences restriction, which, in the approval elections, is identical to the candidate range model (i.e., voters' radii are zero). We again provide only a brief explanation of the algorithm, and for specifics we refer to the original paper. This algorithm has many similarities to the previous one (i.e., the FHHR algorithm), mainly it utilizes a greedy strategy as well. Let us consider the CC-DV problem under the approval voting rule. Having a set of candidates C, a set of voters V in the CR order, a designated winner candidate p, and a maximum number of voter deletions k, for each candidate $c \in C$, we say that the voters who approve of c form an interval, $v_{s(c)}, v_{s(c)+1}, ..., v_{f(c)}$ (this follows from the CR property). Additionally, we define a set of dangerous candidates: $\widetilde{C} = \{c \in C \setminus \{p\} | \, scoreDelta(c,p) \geq 0\}$, and a subset of voters who do not approve of p: $V' = \{v \in V | \, p \notin A(v)\}$. The rest of the algorithm is straightforward. It executes the following operation until p is a winner, or we fail by reaching the limit of deletions k. First, we need to select a dangerous candidate that we will try to defeat. To do so, we find an opponent $c_N \in \widetilde{C}$, for whom his or hers voter's interval end is the smallest (min $f(c_N)$). Then, we try to delete $scoreDelta(c_N, p) + 1$ voters, starting with the voter at the end of interval (i.e., $v_{f(c_N)}$). If we fail, the solution does not exist. After each iteration of this step, we recalculate the sets $\widetilde{C}$ and V'.

5.2 Constructive Control by Deleting Voters For VCR Elections

In this section we describe our algorithm and consider all the caveats. Let us consider a problem of CC-DV under the approval voting rule. The input consists of a VCR election with a set of candidates C, a set of voters V, a designated winner candidate p, and a maximum number of voter deletions k. First, we discuss some concepts and observations, that will help us explain the algorithm. We use the representation of an agent in a VCR election by his or hers range beginning and end (see Definition 5.1.2). Next, for p to become a unique winner, we need to defeat every candidate that has at least as many votes as p. We call this subset of candidates a set of dangerous candidates (opponents), formally:

$$\widetilde{C} = \{c \in C \setminus \{p\} | \, scoreDelta(c,p) \geq 0\}.$$

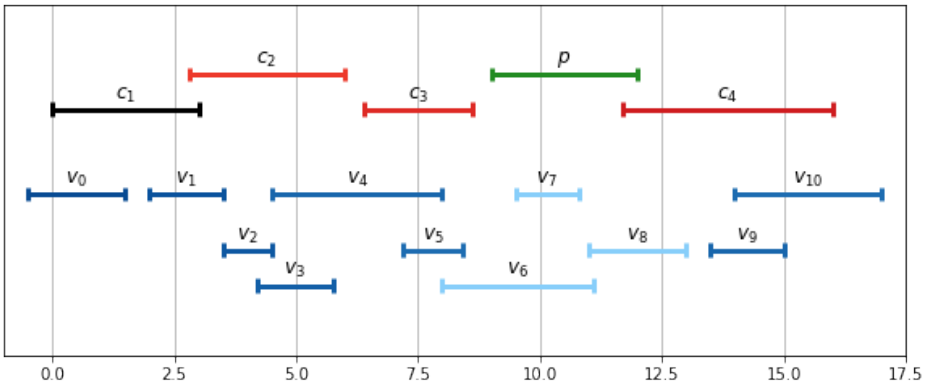


Figure 5.2 A VCR election for the CC-DV problem. The candidates are labeled with c_i (expect for the preferred candidate p), and the voters with v_i. We use colors to distinguish some key groups of agents. That is, p is in green, and voters that approve of him or her (we never delete them) are in light-blue color. Additionally, all the initially dangerous candidates are in red.

Another observation is that we never delete a voter who approves of p. Clearly, it would not bring us any closer to a solution. Thus, when deleting voters, we consider only the ones from the set:

$$V' = \{v \in V \,|\, p \notin A(v)\}.$$

Our algorithm is a greedy one. It repeats the following steps until p becomes a unique winner, or it fails by exceeding the maximum number of voter deletions k. First, we find a dangerous candidate that we will try to defeat. Then, to achieve this goal, we decide which voters to delete. After each iteration, we recompute approvals and the set of dangerous candidates $\widetilde{C}$. We describe these steps in detail below:

1. **Choosing opponents.** The only candidates that we consider are from the set of opponents $\widetilde{C}$. We pick the leftmost candidate, that is, the one with the minimal beginning of the range, $b(c)$ (we break ties arbitrarily). We refer to the selected candidate as c_L. This strategy together with recalculating $\widetilde{C}$ after each iteration, ensures that we perform only the necessary steps to find a solution.

2. **Deleting voters.** Having currently processed opponent c_L. We consider only the voters who approve of him or her, i.e., $\widehat{V} = \{v \in V' \,|\, c_L \in A(v)\}$. To defeat c_L, we need to delete $scoreDelta(c_L, p) + 1$ voters from $\widehat{V}$. If $|\widehat{V}| < scoreDelta(c_L, p) + 1$, then we fail (it is impossible to delete enough voters to defeat c_L). We proceed with deleting the rightmost voters, that is, the ones with the maximum end of the range $e(v)$ (we

break ties arbitrarily). We believe that this is an optimal strategy, because choosing the rightmost voter maximizes the number of other dangerous candidates who might be contained inside each voter's range. Furthermore, it is perfectly safe to do so for two reasons: The first one is our leftmost candidate picking strategy. At this point any dangerous candidate to the left of c_L would have already been defeated. The second reason is related to the special case of contained candidates that we discuss in detail in Section 5.2.1.

We illustrate the algorithm in Example 5, which describes an election from Figure 5.2.

Example 5. *Let us discuss an instance of the CC-DV problem. We consider the VCR election from Figure 5.2, the designated candidate P with 3 votes (i.e., $\{v_6, v_7, v_8\}$) and the set of dangerous candidates $\widetilde{C} = \{c_2, c_3, c_4\}$. We proceed to find the smallest number of voters that we need to delete for P to win the election. First, we choose the leftmost dangerous candidate c_2. We need to delete two voters ($scoreDelta(c_2, P) + 1$). We pick the rightmost voters v_4 and v_3. Now, an important observation is that by removing v_4 we defeated next dangerous candidate c_3. We finish this iteration, and recompute approvals and the set $\widetilde{C}$. The only dangerous candidate left is c_4, and we just need to delete one voter. We pick v_{10} (in this case it would not matter if we chose v_9). We succeed, p is the winner, and in total we removed three voters, i.e, v_4, v_3 and v_{10}.*

In the subsequent sections, first, we discuss a certain special case for which our voter picking strategy might not be obvious. Later, we provide informal sketch proofs that show how our algorithm is equivalent to the MF and the FHHR ones. Finally, we give a formal proof of correctness of the algorithm.

5.2.1 Special Case of Contained Candidates

In the VCR model it is possible that some candidates have their interval fully contained inside other candidate's interval (see Figure 5.3). If so, every voter who would approve the smaller candidate (i.e., the internal one) would also approve the larger one (i.e., the external one), but not necessarily the other way round. For this case, our algorithm's strategy of choosing rightmost voters to delete, might seem suboptimal at first glance. We define additional terminology and symbols to help us explain why we do not have to worry about this issue.

Definition 5.2.1. *Consider an instance of the CC-DV problem, with an approval election E, a set of candidates C, and the designated winner candidate p. We extend the representation*

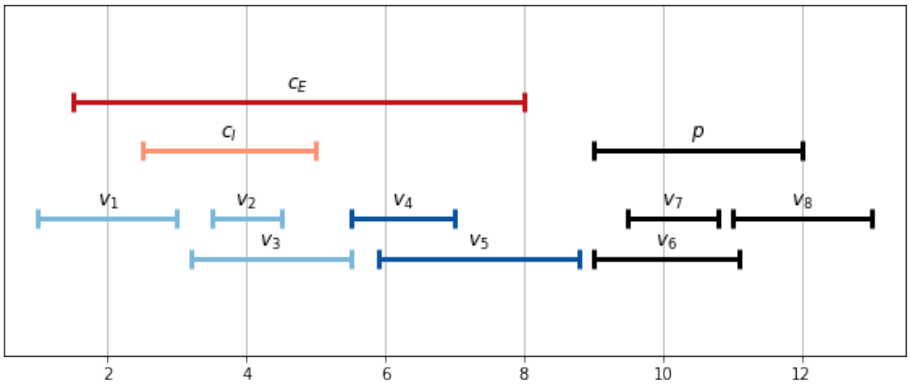


Figure 5.3 A VCR election with two candidates contained inside each other's ranges. The larger one c_E (external), and the smaller one c_I (internal). Furthermore, voters from the set $\widehat{V}$, are divided into two groups, i.e., $\widehat{V}_E = \{v_4, v_5\}$ a subset of voters who approve of c_E but not c_I, and a subset of remaining voters $\widehat{V}_I = \{v_1, v_2, v_3\}$. Because we discuss the CC-DV problem, there is also the preferred candidate p and his voters.

of the score difference between two candidates (see Definition 5.1.3). For each candidate $c \in C \setminus \{p\}$ we represent a number of voter deletions required for p to defeat c with:

$$\Delta(c) = score_{AV}(E, c) - score_{AV}(E, p) + 1.$$

Definition 5.2.2. *Consider an election E, where a candidate is contained inside another one. The two candidates are c_E and c_I, where the range of c_I is contained in that of c_E. The voters who approve of c_E or c_I are in the set $\widehat{V}$. We divide the voters into two groups. First, a subset of voters who approve of c_E but not c_I:*

$$\widehat{V}_E = \{v \in \widehat{V} \mid c_E \in A(v) \wedge c_I \notin A(V)\},$$

and the remaining ones:

$$\widehat{V}_I = \widehat{V} \setminus \widehat{V}_E.$$

We know the size of these subsets based on the approval scores of each candidate:

$$|\widehat{V}_E| = score_{AV}(E, c_E) - score_{AV}(E, c_I),$$

$$|\widehat{V}_I| = |\widehat{V}| - |\widehat{V}_E|.$$

Let us consider an instance of the CC-DV problem, with an election that has the special case. We keep in mind that the internal candidate c_I might not even be a dangerous candidate at all (i.e., $\Delta(c_I) \leq 0$). For p to defeat c_E we need to delete $\Delta(c_E)$ voters, and to defeat c_I (independently of c_E), we need to delete $\Delta(c_I)$ voters (see Definition 5.2.1). Now we show that in the process of defeating the larger candidate we always automatically defeat the smaller one. Using simple transformations in Equation 5.1 we show that:

$$\begin{aligned} \Delta(c_E) = score_{AV}(c_E) - score_{AV}(p) + 1 = \\ |\widehat{V}_E| + score_{AV}(c_I) - score_{AV}(p) + 1 = |\widehat{V}_E| + \Delta(c_I). \end{aligned} \tag{5.1}$$

It means that to defeat c_E we have no other choice but to delete at least $\Delta(c_I)$ voters from $\widehat{V}_I$. The remaining $|\widehat{V}_E|$ voters can be deleted from any of the subsets (i.e., either $\widehat{V}_E$ or $\widehat{V}_I$). This will result in defeating both c_E and c_I (automatically).

5.2.2 Sketch Proof For CR Elections

We show how our CC-DV algorithm for the VCR elections is equivalent to the one of Magiera and Faliszewski (2017) for the CR elections (the MF algorithm). Let us consider an instance of the CC-DV problem for a CR election (agents are still represented using the VCR model). In the explanation we follow the example from Figure 5.4.

Choosing opponents

In the MF algorithm, we choose dangerous candidates to process ordered by their voter's interval end (i.e., $v_{e(c)}$), ascending. We see that it is equivalent to picking the leftmost

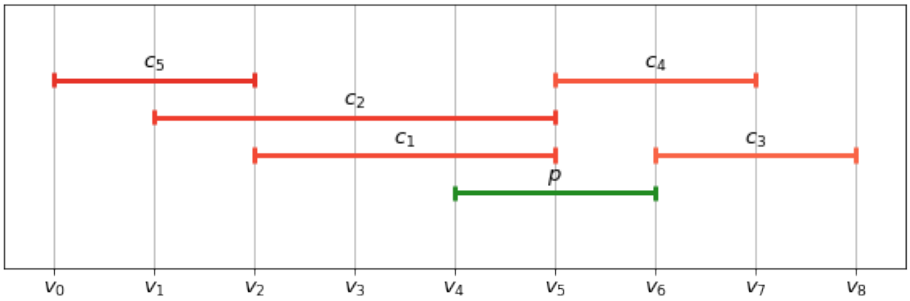


Figure 5.4 A CR election for the CC-DV problem, based on an example from Magiera and Faliszewski (2017). The candidates are labeled with c_i (expect for the preferred candidate p), and the voters with v_i. We use colors to distinguish some key groups of agents. That is, p is in dark-green, and all the initially dangerous candidates are in red.

candidate (i.e., min $b(c)$) from the set of opponents $\widehat{C}$ in the VCR algorithm. The selected candidate will also have the smallest $v_{e(c)}$ (candidate c_5 in the example). Of course the above statement is not true for the special case of the contained candidates. However, in Section 5.2.1, we already explained why it is irrelevant for the overall result.

Voter deletion

Now, once we know how to pick an opponent c_L, we show that choosing voters is identical as well. The notion of the beginning/end of the voter's interval in the CR model is equivalent to the left/right end of the voter's range in the VCR setting. Both algorithms try to delete $scoreDelta(c_L, p) + 1$ voters, the MF one starts deleting voters from the end of interval, while the VCR one does virtually the same by deleting the rightmost voters first (i.e., max $e(v)$).

5.2.3 Sketch Proof For VR Elections

We show how our CC-DV algorithm for the VCR elections is equivalent to the one from Faliszewski et al. (2011) for the VR elections (the FHHR algorithm). Let us consider an instance of the CC-DV problem for a VR election (agents are still represented using the VCR model). In the explanation we follow the example from Figure 5.5. Strategy of the FHHR algorithm is significantly different from the VCR and the MF ones. However, if we introduce one simple assumption and an additional step to our VCR algorithm, they become identical. First, let us assume that the preferred candidate p is always the first candidate in the VR order (see election in Figure 5.5). This assumption is correct based on two facts:

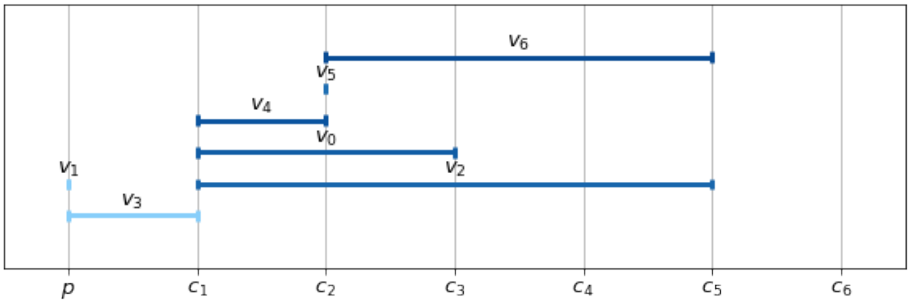


Figure 5.5 A VR election for the CC-DV problem. The candidates are labeled with c_i (expect for the preferred candidate p), and the voters with v_i. We use colors to distinguish some key groups of agents. That is, all the voters who approve of p (we never delete them) are in light-blue color.

1. Having candidates in order $[c_L, p, c_R]$, if there were a voter v who approved of c_L and c_R, then he or she would also have to approve of p (because of the VR property), so we could not delete v anyway.

2. Next, for the VCR algorithm (with the above assumption) to be complete, we need to extend it by using the "universe flip" method from the FHHR algorithm. That is, we process only the dangerous candidates to the right of candidate p, and once there is none left, we proceed with the remaining ones on the left side of p, treating them as if they were on the right side.

Choosing opponents

We can see that the VCR algorithm strategy (the extended one) of picking the leftmost dangerous candidates is identical to the FHHR algorithm choosing the first candidate on the right side of p. However, there is another difference between the algorithms, the VCR one considers every candidate with score equal or higher than p to be a dangerous candidate (i.e., $\{c_1, c_2, c_3, c_4, c_5\}$ in the example). Whereas the FHHR one limits this set, by picking only candidates that have strictly more votes than a previously picked dangerous candidate (i.e., $\{c_1, c_2\}$ in the example). Nonetheless, in the VCR algorithm we recompute the set of dangerous candidates after each iteration (we defeat $\{c_3, c_4, c_5\}$ as a side effect of defeating c_1), which ensures that we never process any unnecessary dangerous candidates.

Voter deletion

The reasoning here is very similar to the one in the voter deletion part of the CR sketch proof. Both algorithms try to delete $scoreDelta(c, p) + 1$ voters. The FHHR algorithm starts deleting voters from the end of interval (max $e(v)$). The VCR one does virtually the same by deleting the rightmost voter first.

5.2.4 Proof For the VCR Algorithm

Now, we provide a formal proof of correctness of our VCR algorithm. We work under the assumption that in the election there are no contained candidates (see Section 5.2.1).

Proposition 3. *The VCR CC-DV algorithm finds an optimal solution in polynomial time.*

Proof. Let S be the solution found by the algorithm and let S^* be an optimal solution to the problem, that is, a minimal-size set of removed voters. We consider every dangerous candidate $c \in \widetilde{C}$. They are ordered by their range's beginnings, ascending (i.e., left to right).

We choose the first candidate in this order, for whom there is a voter $v_r \in S - S^*$. When v_r was added to S, it was the rightmost voter (max $e(v_r)$). Since S^* is optimal, then there must be some voter $v_x \in S^* - S$, such that v_x approves the first candidate and $e(v_x) \leq e(v_r)$, because v_r is the rightmost voter. Let the set $S^{*\prime} = S^* - \{v_x\} \cup \{v_r\}$ be S^* but substituting v_r for v_x. We know that $S^{*\prime}$ is a correct solution, because the approval set of the voter v_r contains at least as many candidates as the approval set of the voter v_x. Since $|S^{*\prime}| = |S^*|$, and we can repeat the just-described process, we conclude that $S^{*\prime}$ is a minimal-size set of removed voters. □

5.2.5 Discussion

We successfully demonstrated a polynomial-time algorithm for Approval-CC-DV for VCR elections. Next step would be to consider other variants of control problem, that is, CC-AV, DC-DV and DC-AV (destructive control). We believe that our reasoning used in the CC-DV algorithm could be extended to the CC-AV one. Another interesting aspect would be to study higher dimensional VCR restrictions, for example, two-dimensional one (2D-VCR). However, Godziszewski et al. (2021) showed that many of committee selection problems remain NP-hard for 2D-VCR elections. Would the outcome be similar for control problems?

Chapter 6

Summary

6.1 Conclusions

We have studied various problems, all centered around the one-dimensional Voter/Candidate Range preferences restriction. First, in Chapter 3 we proposed the VCR detection algorithm based on a reduction to ILP. Our solution is perfectly suitable for basic analysis of the domain, however, it lacks in the terms of performance when it comes to recognizing larger profiles. Next, in Chapter 4 we performed an extensive analysis of the VCR domain. We showed that there is a significant number of profiles that satisfy the TVCR (i.e., VCR but neither VR nor CR) property. These results are very promising, and encourage further research and development related to the VCR domain. Finally, in Chapter 5 we demonstrated the advantages of introducing this preference restriction. That is, we gave a polynomial-time algorithm for the constructive control by deleting voters (CC-DV) problem for the VCR profiles, which is NP-hard for the unrestricted profiles. Furthermore, we showed that the VCR based algorithms can extend and replace the existing ones for the VR and CR restrictions.

6.2 Open Problems

The Concept of the VCR restriction was introduced only recently by Godziszewski et al. (2021). Naturally, there is a multitude of unanswered questions and possible research directions. Particularly because we have studied only the one-dimensional variant of the VCR. It would be interesting to see more solutions based on the VCR model for other computational social choice problems. For example, one might try to extend the minimax approval-voting (MAV) algorithms for the VR and the CR profiles (see the work of Liu and Guo (2016)). Moreover, we believe that improving our results for the VCR profile

recognition algorithms (see Chapter 3), would be greatly beneficial. One of the possible directions for this would be to study combinatorial properties of the VCR domain.

Bibliography

Arrow, K. (1951). *Social Choice and Individual Values*. Wiley: New York.

Aspvall, B., Plass, M. F., and Tarjan, R. E. (1979). A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Information Processing Letters*, 8(3):121–123.

Aziz, H., Brill, M., Conitzer, V., Elkind, E., Freeman, R., and Walsh, T. (2017). Justified Representation in Approval-Based Committee Voting. *Social Choice and Welfare*, 48(2):461–485.

Barberà, S. and Moreno, B. (2011). Top Monotonicity: A Common Root for Single Peakedness, Single Crossing and the Median Voter Result. *Games and Economic Behavior*, 73(2):345–359.

Bartholdi, J. J., Tovey, C. A., and Trick, M. A. (1989). The Computational Difficulty of Manipulating an Election. *Social Choice and Welfare*, 6(3):227–241.

Bartholdi, J. J., Tovey, C. A., and Trick, M. A. (1992). How Hard Is It to Control an Election? *Mathematical and Computer Modelling*, 16(8):27–40.

Black, D. (1958). *The Theory of Committees and Elections*. University Press, Cambridge.

Booth, K. S. and Lueker, G. S. (1976). Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms. *Journal of Computer and System Sciences*, 13(3):335–379.

Brams, S. J. and Fishburn, P. C. (2007). *Approval Voting, 2nd edition*. Springer.

Brandt, F., Brill, M., Hemaspaandra, E., and Hemaspaandra, L. A. (2015). Bypassing Combinatorial Protections: Polynomial-Time Algorithms for Single-Peaked Electorates. *Journal of Artificial Intelligence Research*, 53:439–496.

Brandt, F., Conitzer, V., Endriss, U., Lang, J., and Procaccia, A. D., editors (2016). *Handbook of Computational Social Choice*. Cambridge University Press.

Conitzer, V. and Walsh, T. (2016). Barriers to Manipulation in Voting. In Brandt, F., Conitzer, V., Endriss, U., Lang, J., and Procaccia, A. D., editors, *Handbook of Computational Social Choice*, pages 127–145. Cambridge University Press.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. (2015). *Parameterized Algorithms*. Springer.

Doignon, J. and Falmagne, J. (1994). A Polynomial Time Algorithm for Unidimensional Unfolding Representations. *Journal of Algorithms*, 16(2):218–233.

Elkind, E., Faliszewski, P., and Skowron, P. (2020). A Characterization of the Single-Peaked Single-Crossing Domain. *Social Choice and Welfare*, 54(1):167–181.

Elkind, E. and Lackner, M. (2015). Structure in Dichotomous Preferences. In Yang, Q. and Wooldridge, M. J., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2019–2025. AAAI Press.

Elkind, E., Lackner, M., and Peters, D. (2016). Preference restrictions in computational social choice: Recent progress. In Kambhampati, S., editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4062–4065. IJCAI/AAAI Press.

Faliszewski, P., Hemaspaandra, E., and Hemaspaandra, L. A. (2009). How Hard Is Bribery in Elections? *Journal of Artificial Intelligence Research*, 35:485–532.

Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L. A., and Rothe, J. (2011). The Shield that Never Was: Societies with Single-Peaked Preferences are More Open to Manipulation and Control. *Information and Computation*, 209(2):89–107.

Faliszewski, P. and Procaccia, A. D. (2010). AI's War on Manipulation: Are We Winning? *AI Magazine*, 31(4):53–64.

Faliszewski, P. and Rothe, J. (2016). Control and Bribery in Voting. In Brandt, F., Conitzer, V., Endriss, U., Lang, J., and Procaccia, A. D., editors, *Handbook of Computational Social Choice*, pages 146–168. Cambridge University Press.

Faliszewski, P., Skowron, P., Slinko, A., and Talmon, N. (2017). *Multiwinner Voting: A New Challenge for Social Choice Theory*. AI Access.

Fitzsimmons, Z. and Lackner, M. (2020). Incomplete Preferences in Single-Peaked Electorates. *Journal of Artificial Intelligence Research*, 67:797–833.

Godziszewski, M. T., Batko, P., Skowron, P., and Faliszewski, P. (2021). An Analysis of Approval-Based Committee Rules for 2D-Euclidean Elections. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 5448–5455. AAAI Press.

Gurobi Optimization, L. (2021). Gurobi Optimizer Reference Manual.

Hemaspaandra, E., Hemaspaandra, L. A., and Rothe, J. (2009). Hybrid Elections Broaden Complexity-Theoretic Resistance to Control. *Mathematical Logic Quarterly*, 55(4):397–424.

Lackner, M. and Skowron, P. (2020). Approval-Based Committee Voting: Axioms, Algorithms, and Applications. *CoRR*, abs/2007.01795.

Liu, H. and Guo, J. (2016). Parameterized Complexity of Winner Determination in Minimax Committee Elections. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '16, page 341–349, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Magiera, K. and Faliszewski, P. (2017). How Hard is Control in Single-Crossing Elections? *Autonomous Agents and Multiagent Systems*, 31(3):606–627.

Magiera, K. and Faliszewski, P. (2019). Recognizing Top-Monotonic Preference Profiles in Polynomial Time. *Journal of Artificial Intelligence Research*, 66:57–84.

Mirrlees, J. A. (1971). An Exploration in the Theory of Optimum Income Taxation. *Review of Economic Studies*, 38(2):175–208.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley interscience series in discrete mathematics and optimization. Wiley.

Peters, D. (2017). Recognising Multidimensional Euclidean Preferences. In Singh, S. P. and Markovitch, S., editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 642–648. AAAI Press.

Peters, D. and Lackner, M. (2020). Preferences Single-Peaked on a Circle. *Journal of Artificial Intelligence Research*, 68:463–502.

Roberts, K. W. (1977). Voting Over Income Tax Schedules. *Journal of Public Economics*, 8(3):329–340.

Skowron, P., Yu, L., Faliszewski, P., and Elkind, E. (2015). The Complexity of Fully Proportional Representation for Single-Crossing Electorates. *Theoretical Computer Science*, 569:43–57.

Vazirani, V. V. (2001). *Approximation algorithms*. Springer.

Walsh, T. (2007). Uncertainty in Preference Elicitation and Aggregation. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 3–8. AAAI Press.