FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Structured Committee Election
Student:	Terezie Hrubanová
Supervisor:	doc. RNDr. Dušan Knop, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2023/2024

Instructions

The task is to survey committee elections and some possible approaches to their structural limitations. The focus should be on current literature and interval-based limitations. One of the most prominent and systematic approaches is based on the so-called OWA vectors. The student should find efficient algorithms for selected combinations of structural limit and a specific OWA vector.

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 2 February 2023 in Prague.

Bachelor's thesis

STRUCTURED COMMITTEE ELECTION

Terezie Hrubanová

Faculty of Information Technology
Department of Theoretical Computer Science
Supervisor: doc. RNDr. Dušan Knop, Ph.D.
May 9, 2023

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Hrubanová Terezie. *Structured committee election.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

List of Figures

List of Tables

List of Algorithms

First and foremost, I would like to thank my supervisor doc. RNDr. Dušan Knop, Ph.D. for his guidance and valuable advice. Next, I would like to express my gratitude to my friends who helped me with my thesis and supported me throughout my studies. I would like to name Matěj Doležálek, Ondřej Staníček and Petr Šejvl.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

In Prague on May 9, 2023 . . . .

Abstract

This thesis focuses on winner selection in committee election. We provide an overview of voting systems for single-winner and multi-winner election. Winner selection is often computationally hard, therefore we introduce election with structured preferences. In election with structured preferences, the votes are in some way restricted which may help to create more efficient winner selection algorithms. We describe some of the known structured preferences.

We provide an overview of the current literature on the topic of structured and nearly structured preferences. We also review current work on committee election with structured preferences and usage of ordered weighted average (OWA) in committee election.

We propose polynomial-time algorithms for finding a winning committee for approval election with OWA vector $(0, \dots, 0, 1)$ and interval preference restrictions. In such election, a voter approves a committee only if she approves all of its members. We use this property and show two approaches for finding a winning committee.

Keywords election, committee election, multi-winner election, OWA voting systems, structured preferences, voter interval, candidate interval

Abstrakt

Tato práce se zabývá hledáním vítěze ve volbách komisí. Poskytujeme přehled volebních systémů pro single-winner a multi-winner volby. Hledání vítěze je často výpočetně náročné, proto zavádíme volby se strukturovanými preferencemi. Ve volbách se strukturovanými preferencemi jsou hlasy voličů nějakým způsobem omezené, což může usnadnit vytváření efektivnějších algoritmů pro hledání vítěze. Popisujeme některé ze známých strukturovaných preferencí.

Poskytujeme přehled současné literatury týkající se strukturovaných a téměř strukturovaných preferencí. Zkoumáme současné práce týkající se volby komisí se strukturovanými preferencemi a použití ordered weighted average (OWA) ve volbách komisí.

Představujeme polynomiální algoritmy pro hledání vítězných komisí v approval volbách s OWA vektorem $(0, \dots, 0, 1)$ a intervalově omezenými voličskými preferencemi. V takových volbách, volič schvaluje komisi pouze pokud schvaluje všechny její členy. Tuto vlastnost používáme a ukazujeme dva přístupy pro hledání výherní komise.

Klíčová slova volby, volby komisí, multi-winner volby, OWA volební systémy, strukturované preference, interval voličů, interval kandidátů

List of abbreviations

CI	Candidate Interval
FPT	Fixed Parameter Tractable
OWA	Ordered Weighted Average
STV	Single Transferable Vote
SNTV	Single Non Transferable Vote
VI	Voter Interval

Chapter 1
Introduction

Elections are typically associated with politics. Citizens vote for their favourite candidates or political parties in order to choose their representation. However, there are many other situations where a group needs to come to a collective decision. It can be a group of friends picking a restaurant, deciding on sights for a family to visit on their holiday, or the process of selecting a talent show winner. We are surrounded by elections, so it is not surprising that there is a lot of research focused on their various types.

We distinguish two types of elections. These two types are elections where the winner is a single candidate and elections where we want to select a set of candidates, *a committee*, of a fixed size. Committee election is in some ways more challenging. Instead of selecting simply the *best* candidate (whatever the criteria for being the *best* is), we need to find some set of candidates to maximize the satisfaction of the voters. However, selecting the committee as the set of best individually ranked candidates is not a universal solution. Imagine, for example, the process of selecting a set of meals to serve at a picnic. Nearly everyone loves pizza, so serving only various kinds of pizzas would make a lot of people happy. Yet, there can be a minority of people who dislike or even cannot eat pizza. For this reason, there is a many different strategies for selecting the winner, i.e., *voting systems*. We focus mainly on the so-called *OWA based voting systems* which cover a whole range of committee voting systems.

The task of finding a winning committee is computationally hard in many cases. However, voters' preferences are not always random. There can be some sort of structure in their preferences and that may help us find a winning committee more efficiently. Imagine the process of selecting movies for a movie night with a group of friends. Most of the friends have a favourite genre of movies and they would, to a certain extent, enjoy watching any movie of this genre. None of the voters probably has a horror and a romantic comedy movie as their two most preferred choices. We describe some of the known ways, how preferences can be *structured*.

In this thesis, we aim to acquaint the reader with the topic of committee election. We introduce necessary terminology and review current literature with focus on OWA based election and structured preferences. Lastly, we aim to create algorithms for winning committee selection in OWA based election with certain OWA vectors and interval structured preferences.

After building the theoretical background in Chapter 2 and reviewing current literature in Chapter 3, we propose several original algorithms for winning committee selection in Chapter 4. We focus on specific voting systems and structured preferences and describe two approaches to selecting a winning committee.

Chapter 2

Definitions

This chapter contains an overview of the basic aspects of election. This thesis focuses on the problem of selecting a winner of an election. Therefore, we introduce a number of methods (*voting systems*) for winner selection. We describe some of their properties, however, for a deeper understanding of the properties of voting systems, we refer the reader to [1, 2].

Selecting a winner of an election is often computationally hard. For this reason, we introduce *structured preferences*, selecting a winner in an election with structured preferences can be easier than in the general case. We explain structured preferences more in detail in Section 2.3. In some situations, voters' preferences are structured naturally. In Section 2.3, besides the definitions, we provide examples of such situations when the preferences are structured. For a more in-depth overview of preference restrictions, we refer to [3].

Our definitions are mostly adopted from [2, 4].

2.1 Basic terminology

Notation We denote the interval $\{i \in \mathbf{N} \colon a, b \in \mathbf{N}, a \leq i \leq b\}$ as $[a, b]$ and integers a, b as the *lower* and the *upper* bound respectively.

We adopt the notation from [4]. For a positive integer r, we use $[r]$ to denote interval $[1, r]$. For a logical expression P, we use $[P]$ in the sense of the Iverson bracket where $[P] = 1$ if P is true and 0 otherwise.

We use *(non-strict) linear order* to refer to a binary relation $\leq$ on some set X, that is reflexive, transitive, antisymmetric and for all $a, b \in X, a \leq b$ or $b \leq a$. We use *strict linear order* to refer to a binary relation $<$ on some set X, that is irreflexive, transitive, asymmetric and for all $a, b \in X$, if $a \neq b$, then $a < b$ or $b < a$.

▶ **Definition 2.1** (Election)**.** *An* election $E = (C, V)$ *consists of a set of* m candidates $C = \{c_1, \ldots, c_m\}$ *and a collection of* n votes $V = \{v_1, \ldots, v_n\}$*. Each vote* v_i *is associated to a voter* $i \in [n]$ *and expresses* preference *(defined later in Definition 2.4 and 2.5) of the voter over candidates* C*.*

It is worth noting that in the Definition 2.1, V is not a set since multiple voters can have the same preferences.

When referring to voters and candidates, we use "he" and "she" (semi-)randomly. This approach is inspired by [1].

We use a certain *voting rule* (*voting system*) to determine the winner of an election. The winner can be one candidate in *single-winner* election or a set of candidates (*committee)* in *multi-winner (committee) election.*

▶ **Definition 2.2** (Single-winner voting rule)**.** Single-winner voting rule (system) $\mathcal{R}$ *is a function that for every given election $E = (C, V)$ selects a (possibly empty) subset of C, which we call* winners*.*

▶ **Definition 2.3** (Committee voting rule)**.** Committee (multi-winner) voting rule (system) $\mathcal{R}$ *is a function that for every given election $E = (C, V)$ and a positive integer $k \leq m$ selects a (possibly empty) set of k-element subsets of C, which we call* winning committees*.*

Some voting systems guarantee a *unique winner* meaning that the winner of the election is always one candidate (resp. one committee). Such voting systems are called *resolute*. In the case of *non-unique* winner, the voting rule returns a set of candidates (resp. committees) that tie for winning. If we require to obtain only one winner, we can further apply some tie-breaking mechanisms (e.g., randomly picking one of the tied candidates as a winner).

There are also rules that for some election do not select a winner, i.e., the returned set is empty.

Different voting rules require a different form of expressing voters' preferences. The two most common are votes in the form of *preference order* and *approval preferences*.

▶ **Definition 2.4.** *In election $E = (C, V)$, $V = \{v_1, \dots, v_n\}$, a* preference order (preference list) *v_i is a strict linear order over C, where the candidates are ordered from the most preferred one by voter i.*

We write $a \succ_i b$, for $a, b \in C$, if a precedes b in the preference order v_i. We interpret $a \succ_i b$ so that voter i strictly prefers candidate a to candidate b.

In some cases, preference order is defined as non-strict linear order, i.e., voter is allowed to be indifferent in between candidates. It is referred to as *weak preference order*, however, we do not consider such preference order in this thesis. Preferences expressed as a preference order are also referred to as *ordinal preferences*.

▶ **Definition 2.5.** *In election $E = (C, V), V = \{v_1, \dots, v_n\}$, an* approval preference *v_i of a voter i is a partition of C in two sets of approved and unapproved candidates.*

We write $a \succ_i b$, for $a, b \in C$, if a is approved and b is unapproved.

We refer to the collection of votes V in the form of preference orders as *ordinal preference profile* and in the form of approval preference as *approval preference profile*. When the type or preference profile is clear from the context or both are applicable, we simply use *preference profile* or *profile*.

2.2 Voting systems

In this section, we introduce and categorize some of the basic single- and multi-winner voting rules. Unless stated otherwise, these rules do not guarantee a unique winner.

2.2.1 Single-winner voting rules

All of the voting rules in this section require the votes V to be submitted as an ordinal preference profile, i.e., a collection of preference orders.

2.2.1.1 Scoring rules

A scoring rule selects the winner as a candidate with the highest score. Let $\text{pos}_v(c)$ be a position of candidate c in the preference of voter v and $\gamma\colon [m] \to \mathbb{R}$ a scoring function. The score of each candidate is then $\sum_{v \in V} \gamma(\text{pos}_v(c))$.

The scoring function is required to be non-increasing, meaning that for candidates c_i and c_j with $\text{pos}_v(c_i) < \text{pos}_v(c_j)$ it holds that $\gamma(\text{pos}_v(c_i)) \geq \gamma(pos_v(c_ij)$.

Plurality Plurality is a scoring rule using a scoring function $p(i) = [i = 1]$. In other words, each voter gives a point only to the candidate on the top of their preference list.

Veto (antiplurality) Veto is a scoring rule with a scoring function $v(i) = [i < n]$, which means that voters give a point to everyone but their least preferred candidate.

***t*-approval** In t-approval, only the t most preferred candidates of each voter receive a point. We define the scoring function as $a_t(i) = [i \leq t]$ for a fixed $t \leq m$.

In some sense, we can perceive t-approval as a generalization of the previous two scoring rules because both of them can be defined as t-approval with $t = 1$ for plurality and $t = m - 1$ for veto.

Borda rule Borda is a scoring rule with a scoring function $b(i) = m - i$, which means that candidates receive points according to how many candidates are ranked behind them.

Table 2.1 shows, how to determine the winner in plurality, veto, 2-approval and Borda. Note, that for chosen preference profile, the winner(s) of the election are different for each of these scoring systems.

	Points															
	Plurality				**Veto**				**2-approval**				**Borda**			
Preference orders	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
$a \succ b \succ c \succ d$	1	0	0	0	1	1	1	0	1	1	0	0	3	2	1	0
$a \succ d \succ c \succ b$	1	0	0	0	1	0	1	1	1	0	0	1	3	0	1	2
$b \succ d \succ c \succ a$	0	1	0	0	0	1	1	1	0	1	0	1	0	3	1	2
$b \succ d \succ c \succ a$	0	1	0	0	0	1	1	1	0	1	0	1	0	3	1	2
Score	**2**	**2**	0	0	2	3	**4**	3	2	**3**	0	**3**	6	**8**	4	6

■ **Table 2.1** Example of an election with different winners for plurality, veto, 2-approval, and Borda scoring systems

2.2.1.2 Pairwise comparison systems

As the name suggests, pairwise comparison systems compare pairs of candidates in order to determine a winner. For our pairwise comparison systems, we use ordinal preferences.

In order to describe pairwise comparison more accurately, we define *pairwise reduced elections for election E and candidate c*, as a set of all elections where the candidates are reduced to a pair $C' = (c, j), \forall j \in C \setminus \{c\}$ and the votes are reduced accordingly.

An important notion for pairwise comparison voting systems is *Condorcet winner* introduced by Marquis de Condorcet [5].

▶ **Definition 2.6.** *In election $E = (C, V)$, a* Condorcet winner *(resp.* weak Condorcet winner*) is a candidate c who is preferred to the other candidate by a majority (resp. at least half) of voters in all of the pairwise reduced elections for E and c.*

Roughly speaking, we compare all candidates in head-to-head contests and the candidate who wins all of her contests is a Condorcet winner and the candidate who wins or ties all of her contests is a weak Condorcet winner.

It is not hard to see that if there exists a Condorcet winner, she is unique. Every Condorcet winner is also a weak Condorcet winner, however, there can be multiple weak Condorcet winners. In some elections, there is no Condorcet winner. We provide examples for election with Condorcet winner in Table 2.2, multiple weak Condorcet winners in Table 2.3, and no (weak) Condorcet winner in Table 2.4. In our example, the notation $a?b$ means a head-to-head contest of candidates a and b, i.e. election reduced to these two candidates.

	Pairwise comparison		
Preference orders	$a ? b$	$a ? c$	$b ? c$
$a \succ b \succ c$	a	a	b
$b \succ c \succ a$	b	c	b
$a \succ c \succ b$	a	a	c
Winner of comparison	a	a	b

Table 2.2 Election with a Condorcet winner, candidate a is prefered in all of her reduced elections

	Pairwise comparison		
Preference orders	$a ? b$	$a ? c$	$b ? c$
$a \succ b \succ c$	a	a	b
$c \succ a \succ b$	a	c	c
$c \succ b \succ a$	b	c	c
$a \succ c \succ b$	a	a	c
Winner of comparison	a	a, c	c

Table 2.3 Election with two weak Condorcet winners, candidates a and c both win or tie all of their reduced elections

	Pairwise comparison		
Preference orders	$a ? b$	$a ? c$	$b ? c$
$a \succ b \succ c$	a	a	b
$b \succ c \succ a$	b	c	b
$c \succ a \succ b$	a	c	c
Winner of comparison	a	c	b

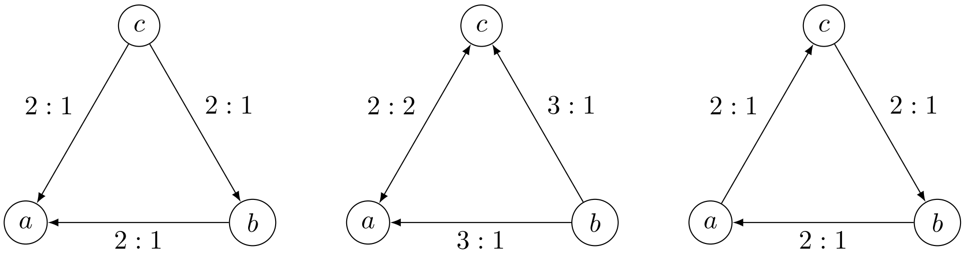
Table 2.4 Election with no Condorcet winner

Pairwise comparison can be also visualized as the so-called majority graph. Each candidate corresponds to a vertex and there is a directed edge from x to y, if y is preferred by majority of the voters to x and a bidirected edge if x and y tie. Edges are also labeled with comparison results, e.g. if 3 of 4 voters prefer x to y, the directed edge from y to x is labeled $3 : 1$. You can see such majority graphs, e.g., in Figure 2.1.

When a voting system *respects the Condorcet winner*, it means that it always selects a Condorcet winner if there is one. All of the voting systems in this section respect the Condorcet winner.

Condorcet voting Condorcet voting is a simple voting system based on selecting a Condorcet winner. If there exists a Condorcet winner, he is the winner of our election, otherwise, we have no winner. It follows that in this voting system we always have one or no winner.

Copeland voting system Unlike Condorcet voting, Copeland voting always selects a (non-unique) winner. Candidates are compared in pairs and receive 1 point if they are preferred by a majority of voters in pairwise comparison, α points with $0 \leq \alpha \leq 1$ if they tie, and 0, otherwise. Every candidate with maximal score is a winner.



(a) Election from Table 2.2 **(b)** Election from Table 2.3 **(c)** Election from Table 2.4

Figure 2.1 Majority graphs for elections in Tables 2.2, 2.3 and 2.4. We can notice that a (weak) Condorcet winner is a vertex, which has all adjacent edges directed toward itself. A candidate that loses every pairwise comparison and has no edges directed toward itself is referred to as a Condorcet loser.

We denote these systems as Copeland$^{\alpha}$, where α is the number of points received in the case of a tie. For example, in Copeland$^{\frac{1}{2}}$, candidates receive 1 point if they win pairwise comparison and $\frac{1}{2}$ point if they tie.

Copeland system is named after Arthur Copeland, who proposed a Copeland$^{\frac{1}{2}}$ system [6], however, there was a version of Copeland rule in the 13th century proposed by a philosopher Ramon Llull. The generalization with arbitrary α was brought later by Faliszewski et al. [7].

It is worth pointing out that there can never be a tie in an election with an odd number of voters and therefore, in that case, the Copeland system is equivalent for all α.

In Example 1 we show an example from [1].

▶ **Example 1.** Table 2.5 describes an election. We can express the score of candidates with arbitrary α as

$$\begin{aligned}\text{Copeland}^{\alpha}(a) &= 1 + 2\alpha,\\ \text{Copeland}^{\alpha}(b) &= \alpha,\\ \text{Copeland}^{\alpha}(c) &= 2,\\ \text{Copeland}^{\alpha}(d) &= 1 + \alpha.\end{aligned}$$

Now, depending on the chosen α, there are different winners. For $\alpha = 0$, the winner is candidate c with 2 points and for $\alpha = 1$, candidate a wins the election with 3 points. There are two winners, candidates a and c for $\alpha = \frac{1}{2}$, both with 2 points.

	Pairwise comparison					
Preference orders	$a\,?\,b$	$a\,?\,c$	$a\,?\,d$	$b\,?\,c$	$b\,?\,d$	$c\,?\,d$
$a \succ d \succ c \succ b$	a	a	a	c	d	d
$c \succ d \succ b \succ a$	b	c	d	c	d	c
$c \succ d \succ b \succ a$	b	c	d	c	d	c
$b \succ d \succ a \succ c$	b	a	d	b	b	d
$a \succ c \succ d \succ b$	a	a	a	c	d	c
$a \succ c \succ b \succ d$	a	a	a	c	b	c
Winner of comparison	a, b	a	a, d	c	d	c

Table 2.5 Election with different Copeland winners for different α

Dodgson system This system aims to select the winner to be as close to a Condorcet winner as possible. It assigns a score to each candidate as a number of swaps with adjacent candidates in voters' preference orders which this candidate needs to become a Condorcet winner. The winner is then the candidate with the lowest score. If there exists a Condorcet winner, she needs no swaps, therefore she wins the election.

Young system Similarly to Dodgson, Young system counts the number of changes needed to make a candidate a Condorcet winner. However, instead of altering the votes, it simply deletes them. Each candidate is assigned a score based on how many votes need to be deleted in order to make him a Condorcet winner and a candidate with the lowest score wins.

2.2.1.3 Other voting systems

In this section, we introduce some other important voting rules. Some of them build on the ideas of simpler already mentioned rules and apply them in stages and some of them do not require the voters to submit complete preference lists (we state this explicitly).

Some of the systems do not guarantee selecting a winner unless we define a tie-breaking mechanism. Tie-breaking mechanism is applied when the candidates are rated equally and we need to choose only one of them. Specifically, we presume that we have a tie-breaking mechanism for plurality with runoff, Single transferable vote (STV), and voting tree systems.

Plurality with runoff This voting system has two stages. The first stage is the same as plurality, the top candidate from the voters' preference lists receives a point. Then, instead of choosing the candidate with the highest score as the winner, we choose two candidates with the best score for the second round. After that, all other candidates are deleted, and the votes are altered so that they are now only preference lists of the two remaining candidates. The overall winner is the candidate who is preferred by a majority of the voters.

Single transferable vote (STV) In the same way as in plurality, all candidates on the top of voters' preference lists receive a point. If there exists a candidate who obtained more than half of the points, she is the winner. If not, we delete a candidate with the least points and alter the votes accordingly. After that, we check again, if any candidate has more than half of the points and we repeat this process until we find a winner.

Voting tree Voting tree is also known as the cup protocol because similar system is used to decide in sports tournaments in the playoff part. We provide a definition for *balanced binary tree* and *sibling vertices*. We use conventional definitions for basic terms such as *tree, binary tree, leaf, parent vertex, subtree, root*, the definitions can be found for example in [8].

▶ **Definition 2.7.** *A binary tree $T = (V, E)$ is* balanced *if for every vertex $u \in V$ it holds that $|L(u)| - |R(u)| \leq 1$, where $|L(u)|$ (resp. $|R(u)|$) is the number of vertices in the left (resp. right) subtree of u.*

▶ **Definition 2.8.** Sibling vertices *of a binary tree are vertices that have the same parent.*

In the voting tree voting system, we create a balanced binary tree of sufficient size, so that we can assign each candidate to a leaf vertex. The candidates in the sibling vertices are then compared and the one being preferred by a majority of voters moves up to the parent vertex. The candidates who moved up are then compared again to their siblings and this process is repeated until there is a winner in the root vertex.

The number of candidates does not necessarily have to be a power of two, if the candidate's sibling vertex is empty, the candidate just automatically moves up to his parent. However, the way how we assign candidates to the leaf vertices for the first round can completely change the outcome of the election. For example, for election from Table 2.4, we can see the different outcomes of the voting tree in Figure 2.2.

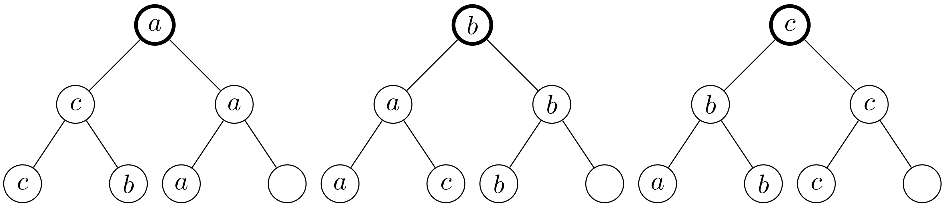


Figure 2.2 Voting trees for election from Table 2.4, where a different assignment of candidates to leaf vertices outputs different winners

Approval voting In approval voting, voters do not have to submit complete linear orderings of candidates, instead, their votes are a set of candidates that they approve. The winner of the election is the candidate who is approved by most of the voters.

It is important to notice the differences between approval voting and the previously mentioned t-approval scoring system. In t-approval, voters submit a preference list of candidates and approve a fixed number t of them, in approval voting, a voter can approve any number of candidates from 0 to m. Naturally, when all voters in approval voting decide to approve exactly t number of candidates the outcome will be equivalent to t-approval.

Range voting Range voting brings an additional option for voters to express preference between the approved voters. Candidates submit the set of approved candidates along with a number expressing their degree of preference, usually in some fixed range. These numbers are then interpreted as scores, they are summed up for each candidate and the candidate with the highest score wins.

Although we assign scores of candidates, this system cannot be interpreted as a scoring rule according to our definition, because the assigned number of points can differ between voters and therefore we would not be able to define a scoring function.

2.2.2 Committee voting systems

Selecting a committee is more difficult than selecting a single winner because the committee can represent voters' opinions in many ways. We have to make a choice, if we want to select a group of k individually best ranked candidates, create a somehow diverse committee, or represent all groups of voters.

Voting rules derived from single-winner voting systems that select k of the best candidates are called *best-k rules.*

▶ **Definition 2.9.** *Committee voting rule $\mathcal{R}$ is a* best-k rule *if there exists a single-winner voting rule $\mathcal{F}$, such that $\mathcal{F}$ returns an ordering over candidates and $\mathcal{R}$ selects the winning committee as the k candidates, that are ranked as the top k candidates in $\mathcal{F}$.*

Note that not all single-winner systems can be used for best-k rules. We can use only those systems that return an ordering over all the candidates (or at least the first k of them). Therefore, we would not be able to use for example Condorcet voting or plurality with runoff, unless we appropriately modify the definition.

The next section covers committee scoring rules, some of them being best-k rules as well.

2.2.2.1 Committee scoring rules

For a committee S, we define a *committee position in a voter's preference order* $\operatorname{pos}(S)$ as an increasing sequence of positions of candidates in voter's preference order, see Example 2.

▶ **Example 2.** For an election $E = (C, V)$ with candidates $C = \{a, b, c, d\}$, preference list $v_i = (b \succ c \succ d \succ a)$, and for a committee $S = \{c, a\}$; the position of S in the preference order of i is $(2, 4)$.

▶ **Definition 2.10.** *A committee voting rule $\mathcal{R}$ is a* committee scoring rule *if there exists a function for every number of candidates m and committee size k that assigns a score to a committee based on its position in voters' preference orders and $\mathcal{R}$ selects winning committee as the committee with the highest sum of these scores.*

SNTV SNTV (Single non transferable vote) is a best-k rule derived from plurality. Voters give a point to every committee, that includes their most preferred candidate.

Bloc This voting system is based on t-approval. We set $t = k$, so the number of approved candidates is the same as the number of committee members. We then choose a committee that has the greatest sum of approval votes.

k-Borda This is a best-k rule based on the Borda system for single-winner election. Voters assign candidates scores with the Borda scoring function and the committees calculate their score as a sum of Borda scores of its members.

Chamberlin-Courant and Monroe In this family of rules, we use a *satisfaction function* $\alpha \colon \mathbb{N} \to \mathbb{N}$ to measure how well each voter is represented. We assign each voter one of the committee members as their representative with an assignment function ϕ and calculate the voter's satisfaction based on the candidate's position in the voter's preference.

The overall satisfaction ℓ then may be computed for example by one of the following:

$$\ell_1(\phi) = \sum_{v \in V} \alpha(\mathrm{pos}_v(\phi(v))), \qquad \ell_{\min}(\phi) = \min_{v \in V} \alpha(\mathrm{pos}_v(\phi(v))).$$

Both Chamberlin-Courant [9] and Monroe [10] find an assignment function ϕ which maximizes the overall satisfaction $\ell(\phi)$. The winning committee is the committee with the highest overall satisfaction. The assignment function for the Monroe rule additionally has to satisfy the Monroe criterion: each candidate in the winning committee is assigned to either $\lceil \frac{n}{k} \rceil$ or $\lfloor \frac{n}{k} \rfloor$ voters.

Function ℓ_1 is a utilitarian measure, it calculates the overall satisfaction as a sum of satisfaction of all voters. Therefore, a possible outcome of an election with ℓ_1 is a small group of voters with high satisfaction and a lot of dissatisfied voters.

Egalitarian measure $\ell_{\min}$ prevents this situation, an election with $\ell_{\min}$ maximizes the satisfaction of the least satisfied voter. However, a different problem can occur when a single voter with diametrically different opinions can cause that the system does not elect a committee that would be preferred by all of the other voters.

Note that SNTV, Bloc, and k-Borda can be defined as a Chamberlin-Courant rule with ℓ_1 and satisfaction functions $\alpha_{\mathrm{SNTV}}(i) = [i = 1]$, $\alpha_{\mathrm{Bloc}}(i) = [i \leq k]$, and $\alpha_{k\text{-Borda}}(i) = i$.

OWA based systems Before defining OWA based system, we first define a *weighted ordered average (OWA) operator over k numbers.*

▶ **Definition 2.11.** *OWA operator f over k numbers is a function defined through OWA vector $\alpha^{(k)} = (\alpha_1, \ldots, \alpha_k)$ of k numbers. Let $x^{(k)}$ be a vector of k numbers and let $x^{\downarrow} = (x_1^{\downarrow}, \ldots, x_k^{\downarrow})$ be the nonincreasing rearrangment of $x^{(k)}$. Then we set*

$$f_{\alpha^{(k)}}(x) = \sum_{i=1}^{k} \alpha_i x_i^{\downarrow} \ .$$

Let W be a k-sized committee. Let $s_{v,W}$ be the vector of scores that voter v assigns to candidates in W. Then, the score of a committee W in election $E = (C, V)$ is

$$S(W) = \sum_{v \in V}^{n} f_{\alpha^{(k)}}(s_{v,W}) \ .$$

Informally, we describe calculating the score as follows. Let W be a k-sized committee. For a voter v, we have a vector $s_{v,W}$ of scores she assigns to candidates in W. We order this vector in nonincreasing order and denote it $s^{\downarrow}_{v,W}$. We then compute the dot product of $s^{\downarrow}_{v,W}$ and the OWA vector $\alpha^{(k)}$. This is the score of committee W from voter v. We repeat this for all voters and calculate the overall score of committee W as a sum of scores from particular voters. The winning committee is then a committee with the maximal score.

For obtaining the vector $s_{v,W}$, we need each voter to assign scores to candidates. Scores can be arbitrary but we distinguish two special cases. In the first one, voters assign scores to candidates based on some single-winner scoring function. Such OWA election is then defined by the OWA vector, committee size, and the single-winner scoring function.

The second case is approval election. Voters submit lists of approved candidates, these receive score 1 whereas disapproved candidates receive 0.

OWA based systems provide further generalization of scoring systems. We can define any best-k rule as an OWA rule with OWA vector $(1, 1, \ldots, 1)$ and with a single-winner scoring function for assigning score to candidates.

We explain the process of calculating the score in Example 3 on the election from Table 2.6.

	Borda score		
	a	b	c
v_1: $a \succ c \succ b$	2	0	1
v_2: $b \succ c \succ a$	0	2	1

Table 2.6 Election with assigned Borda scores

▶ **Example 3.** Consider election from Table 2.6 with committee size 2, OWA vector $(1, 1)$ and Borda scoring function. We show how to calculate the score of a committee $W = (b, c)$.

For v_1, the vector of assigned scores is $s_{v_1,W} = (0, 1)$. We order the elements in nonincreasing order to get a vector $s^{\downarrow}_{v_1,W} = (1, 0)$. We calculate dot product of $s^{\downarrow}_{v_1,W}$ and OWA vector $(1, 1)$ as $(1, 0) \cdot (1, 1) = 1 \cdot 1 + 0 \cdot 1 = 1$. The score of committee W obtained from voter v_1 is 1.

The score obtained from voter v_2 is calculated the same way. We have a vector $s_{v_2,W} = (2, 1)$, order it to get $s^{\downarrow}_{v_2,W} = (2, 1)$. The dot product of $s^{\downarrow}_{v_2,W}$ and the OWA vector $(1, 1)$ is 3.

The overall score of the committee W is then the sum of scores from v_1 and v_2, in this case, 4.

Now, we explain how particular OWA vectors affect the election outcome.

Imagine an example of creating a lunch menu for a restaurant. You need to select some k meals to offer on a particular day and you want to maximize the satisfaction of your customers, i.e., the score they would give to this list of k meals.

In this scenario, customers are voters and assign some score to every meal the restaurant is able to prepare. Since every customer eats only one lunch, he cares only about his most preferred meal on the list. This can be expressed in the election by choosing OWA vector $(1, 0, \ldots, 0)$.

However, if your customers also buy a different meal for dinner along with their lunch, you can simply alter the OWA vector to $(1, 1, 0, \ldots, 0)$. In such an election, the sum of scores of each customers' two favourite meals is maximized.

In this imaginary scenario, your customers may have trouble making decisions and having too many good options bothers them. They can tolerate two good lunch offers on the menu but more than that makes them unhappy. You can express such situation for example by using OWA vector $(1, 1, -1, \ldots, -1)$.

There are no restrictions on OWA vectors, they do not have to be nonincreasing or binary. There can be an election where voters appreciate only the second item with vector $(0, 1, 0, \ldots, 0)$ or election with vector $(1, 2, \ldots, k)$ where their satisfaction increases with the growing number of favourite candidates.

We focus on one specific case of OWA election. In Chapter 4, we propose algorithms for finding a committee in an election with OWA vector $(0, \ldots, 0, 1)$ and approval scoring of candidates. This means that each voter scores the approved candidate with 1 point and disapproved with 0. Let us illustrate such an election with an example.

Imagine a group of friends creating a party playlist. Each of the friends (voters) approves some of the songs (candidates). However, if they hear a song they dislike at a party, it ruins their mood and they are dissatisfied with the party. Therefore, their overall ranking of the party is only as good as the worst song. This is expressed with the OWA vector $(0, \ldots, 0, 1)$. Since they rank the songs with approval votes, we can simply say that the friends (voters) like the party (committee) only if they like all of its songs (candidates). This simplification later helps us create algorithms for finding a winning committee in Chapter 4.

2.3 Preference restrictions

In certain situations, we can expect the voters' preferences to be somehow structured. For example, in a political election, there probably will not be many voters, whose two most preferred candidates are from opposite sides of the political spectrum. Instead, voters will approve candidates, that are in some sense similar. Another example could be roommates who try to decide on a time to set an alarm clock. Probably, each of them will have a certain time interval they find acceptable, it does not make much sense for someone to prefer either exactly 9 or 10 am but disapprove of the times in between.

In this section, we introduce *preference (domain) restrictions*, certain limitations of how the voters are allowed to vote. We refer to restricted voter preferences as *structured preferences.*

2.3.1 Preference list restrictions

Single-peaked preferences One of the basic domain restrictions is *single-peaked* preferences. An example of this domain could be a group of voters deciding on the temperature to set on a thermostat. Everyone has their ideal temperature and the further other temperatures are, the lower they are in the voter's preference order.

To define single-peakedness formally, we use $\operatorname{top}(v_i)$ to denote the most preferred candidate in the preference list v_i.

▶ **Definition 2.12.** *Let V be an ordinal preference profile over the set of candidates C and let $\lhd$ be a linear order over C. A preference list v_i over C is* single-peaked with respect to $\lhd$*, if for every pair of candidates $a, b \in C$ with $\operatorname{top}(v_i) \lhd b \lhd a$ or $a \lhd b \lhd \operatorname{top}(v_i)$ we have $b \succ_i a$.*

A preference profile V over C is single-peaked with respect to $\lhd$ *if every vote in V is single-peaked with respect to $\lhd$.*

A preference profile V over C is single-peaked *if there exists a linear order $\lhd$ over C such that V is single-peaked with respect to $\lhd$. We refer to $\lhd$ as* candidate axis *of the preference profile.*

Single-peaked preferences can be visualized on a graph, where the horizontal axis corresponds to the axis $\lhd$ (i.e., to the ordered voters) and the vertical axis corresponds to the position in voters' preference lists. For example, see Figure 2.3.

Single-crossing preferences Single-crossing domain restriction is similar to single-peakedness but instead of a linear ordering of candidates we have requirements on the voters' order.

Voter	Preference order
v_1	$d \succ c \succ b \succ e \succ a$
v_2	$b \succ c \succ a \succ d \succ e$
v_3	$d \succ c \succ b \succ e \succ a$

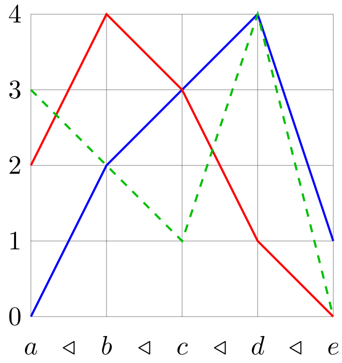


Figure 2.3 Election with voters v_1 and v_2 is single-peaked with respect to $\lhd$. The vote v_3 is not single-peaked with respect to $\lhd$ and therefore election with all three votes would not be single-peaked with respect to $\lhd$.

In the single-crossing domain, we have an ordering of the voters, such that when iterating over this ordering, the relative preference of any two candidates (i.e. their order in the preference lists) changes only once. It means that if the leftmost candidate prefers a over b and the rightmost prefers b over a, there will be only one voter $i \in [n]$, where $a \succ_i b$ and $b \succ_{i+1} a$, i.e., the preferences 'cross' only once.

Single-crossingness could appear, for example, in voting about taxation [11, 12]. Voters are ordered by their income, lower income voters prefer higher tax rates, because it brings them more benefits in the redistribution system, whereas voters with higher income prefer lower taxes.

▶ **Definition 2.13.** *Let V be a preference profile over the set of candidates C and let $\lhd$ be a linear order over V. We say that V is* single-crossing with respect to $\lhd$ *if when V is ordered by $\lhd$, then for every pair of candidates $a, b \in C$, both sets $i \in [n]: a \succ_i b$ and $i \in [n]: b \succ_i a$ are (possibly empty) intervals of $[n]$.*

Preference profile V over C is single crossing, *if there exists voter ordering $\lhd$ such that V is single-crossing with respect to $\lhd$. We refer to $\lhd$ as* voter axis *of the preference profile.*

Single-crossing preference allows great visualization, see Figure 2.4.

Voter	Preference order			
v_1	a	b	c	d
v_2	b	a	d	c
v_3	b	d	a	c

Figure 2.4 Election with single-crossing preferences. We draw a line connecting each candidate across the votes. Because the preferences are single-crossing, all pairs of lines cross only once.

Although single-peaked and single-crossing preferences are related (for more details we refer to [13]), we can find a profile that has any, none, or both of these properties, see examples inspired by [13] in Figures 2.5, 2.6 and 2.7.

2.3.2 Approval votes restrictions

We introduce a specific group of preference restrictions for election systems where voters approve or disapprove the candidates. This preference restriction applies to approval election as well as to scoring rule t-approval.

The two introduced restrictions are interval based, meaning that there exists in some sense an interval of (approved) candidates or (approving) voters. We visualize approval election as

Voter	Preference order		
v_1	a	b	c
v_2	c	a	b
v_3	c	b	a

Figure 2.5 Election with single-crossing but not single-peaked preferences. There is no candidate ordering so the preference would be single-peaked because there are three candidates that are rated in the last position in some preference order.

Voter	Preference order			
v_1	a	b	c	d
v_2	b	a	c	d
v_3	a	b	d	c
v_4	b	a	d	c

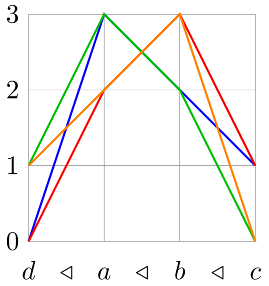


Figure 2.6 Election, that is single-peaked with respect to axis $d \lhd a \lhd b \lhd c$ but is not single-crossing. Certain pairs of voters need to be adjacent because of their same relative preference over a pair of candidates. Specifically, these are

- v_1 and v_2, because of c, d
- v_1 and v_3, because of a, b
- v_3 and v_4, because of c, d
- v_2 and v_4, because of a, b.

However, it is impossible to find such ordering.

a table, where rows correspond to voters and columns to candidates. There is 1 in row v and column c, if voter v approves candidate c and 0 otherwise. For example, see Figure 2.8.

Voter interval preferences The voter interval (VI) domain has a linear ordering of candidates, such that for every voter, the set of approved candidates is an interval. It is in some sense similar to the single-peaked domain if we use t-approval on the single-peaked domain, we can interpret the resulting votes as voter interval. However, there can be a voter interval election with t-approval votes that is not single-peaked (For example, an election from Figure 2.5 has voter interval domain for 1-approval but is not single-peaked.). The example from the beginning of Section 2.3 where roommates approve certain times to set an alarm clock could have a voter interval domain. For example, see Figure 2.8a.

▶ **Definition 2.14.** *Let $E = (C, V)$ be an election with approval votes and $\lhd$ a linear ordering*

Voter	Preference order		
v_1	a	b	c
v_2	b	a	c
v_3	b	c	a

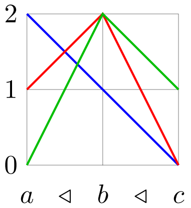


Figure 2.7 Election with single-peaked and single-crossing preference

of C. We say that V is voter interval restricted with respect to $\lhd$*, if when C is ordered by $\lhd$, for every voter v_i, the set $\{j \in C\colon v_i$ approves $j\}$ is an interval.*

We say that V is voter interval restricted *if there exists an ordering $\lhd$ over candidates, such that V is voter interval restricted with respect to $\lhd$.*

We refer to $\lhd$ as voter interval axis*.*

	a	b	c	d	e
v_1	0	1	1	1	0
v_2	1	1	0	0	0
v_3	0	1	1	1	1

(a) Voter interval preference

	a	b	c	d
v_1	0	0	1	1
v_2	1	0	1	1
v_3	0	1	1	0
v_4	0	1	0	0

(b) Candidate interval preference

Figure 2.8 Approval election with interval restricted domain

Candidate interval preferences Election with candidate interval (CI) preferences has a linear ordering of voters, such that for every candidate the set of voters who approve him is an interval. An example of candidate interval preferences could be a group of friends deciding what to eat. If we order the friends based on their tolerance of spice, friends standing close to each other would approve similar meals.

It is not difficult to see that this scenario could also work as a voter interval example if we order the meals by their spiciness. An example of a candidate interval election is in Figure 2.8b.

▶ **Definition 2.15.** *Let $E = (C, V)$ be an election with approval votes and $\lhd$ is a linear ordering of V. We say that V is* candidate interval restricted with respect to $\lhd$*, if when C is ordered by $\lhd$, for every candidate c the set $\{v_i \in V\colon v_i$ approves $c\}$ is an interval.*

We say that V is candidate interval restricted *if there exists a linear ordering of voters $\lhd$, such that V is candidate interval restricted with respect to $\lhd$.*

We refer to $\lhd$ as candidate interval axis*.*

Although voter and candidate interval restrictions are related, there can certainly be a domain that is voter interval restricted but not candidate interval restricted and vice versa. An example of such an election is given in Figure 2.9

	a	b	c	d
v_1	1	1	1	0
v_2	1	0	0	0
v_3	0	1	0	0
v_4	0	0	1	0

(a) Voter interval restricted election

	a	b	c	d
v_1	1	1	0	0
v_2	1	0	1	0
v_3	1	0	0	1
v_4	0	0	0	0

(b) Candidate interval restricted election

Figure 2.9 This is a simple example of preferences that are voter interval restricted but not candidate interval restricted and vice versa.

We explain only the reason why the preferences on Figure 2.9a cannot be candidate interval restricted, the other case works analogously (Figure 2.9b is just a transposed matrix of Figure 2.9a).

In order for the election from Figure 2.9a to be candidate interval restricted, we need to find an ordering of voters. However, voters v_2, v_3, v_4 all need to be adjacent to voter v_1 because they approve one of the same candidates v_1 approves (these are a, b, c respectively). Such an ordering is not possible and thus the domain is not candidate interval restricted.

Chapter 3

Related work

In this chapter, we provide an overview of the current research on election. We focus on literature studying preference restrictions, in particular on articles about structured preference recognition in Section 3.1. Since the voters' preferences are often not perfectly structured, we summarize some of the research on *nearly structured preferences* in Section 3.2.

Since the focus of this thesis is committee election, we dedicate Sections 3.3 and 3.4 to committee election in restricted domains and OWA based election respectively. We investigate the current research and describe different approaches to how ordered weighted average is used in different voting systems.

3.1 Structured preference recognition

Structure in voters' preferences can simplify a whole range of tasks from committee selection (see Section 3.3) to manipulation, control, and bribery (see, e.g., [14, 15]). In this thesis, we suppose, that the information about preference structure is given to us. However, in reality we would just obtain the votes and we would need to examine them for such structures. Therefore, there exist multiple articles on the topic of structured preference recognition.

Elkind and Lackner [16] investigate recognition of *dichotomous preferences*, or as we call them, approval preferences. In their article, they focus on two types of preference restrictions. The first type is restrictions based on an ordering of voters and /or candidates along an axis and requiring the votes to respect the order (this includes also voter and candidate interval). In the second type, they view approval votes as *weak orders* (i.e., non-strict linear orders) and ask if they could be refined to ordinal preference restrictions (e.g., single-peaked, single-crossing). They show that both voter and candidate interval can be recognized in polynomial time by solving the problem of *consecutive ones*.

▶ **Definition 3.1.** *Let M be a $n \times m$ matrix with $M(i, j) \in \{0, 1\}$. We say, that M has* consecutive ones property *if there is a permutation of columns of M, such that in every row, all* 1*-entries appear consecutively.*

It is quite easy to see how the voter (resp. candidate) interval recognition can be reduced to the problem of consecutive ones. Voters and candidates are assigned to rows or columns of a matrix and the matrix has 1 in cells corresponding to a voter approving a candidate. For more details, we refer to [16].

The problem of consecutive ones and its $\mathcal{O}(m^2n)$ time solution was first described by Fulkerson and Gross [17]. The consecutive ones algorithm was further improved in [18, 19, 20, 21].

Reduction to consecutive ones can also be used for recognition of single-peaked [22, 23, 24] or single-crossing [25] domain.

Many of the current articles focus on recognition of Euclidean preferences. In dD-Euclidean preference restriction voters and candidates are mapped to some points in an d dimensional Euclidean space. Voters' preferences then depend on the points' distance, nearer candidates are preferred to the farther ones. Voter and candidate interval, which we use in this thesis, can be described as Euclidean preference as well, as in Godziszewski et al. [26]. Godziszewski et al. define voter-range and candidate-range Euclidean model. In the voter-range model, if a voter approves a candidate, then she also approves all the closer ones and in the candidate-range model, if a candidate is approved by a voter, then she is approved also by all the closer ones. For 1D-Euclidean, these models are equivalent to candidate and voter interval.

There is a number of works proposing algorithms for recognizing 1D-Euclidean election. The first polynomial algorithm was proposed by Doignon and Falmagne [23], followed by algorithms using single-peaked [27] or single-crossing preferences [28] as their starting point. For dD-Euclidean preferences, where $d > 1$, the recognition is shown to be NP-hard by Peters [29].

More on the topic of domain restrictions and their recognition can be found in the works of Elkind, Lackner, and Peters [13, 30].

3.2 Nearly structured preferences

Structure in voters' preferences is often not so straightforward. Imagine the example of a political election in the left-right spectrum. Each voter has his favourite candidate, and the farther the other candidates are from this favourite candidate on the left-right spectrum the lower ranking they receive. This would make the domain single-peaked, however, there are a few voters, who choose their candidate differently. These so-called *maverick voters* can for example make their decision based on personal sympathy or the candidate's policy on a particular issue. Therefore, the preferences are not really single-peaked but are very close to it.

Similarly as structured preferences, also nearly structured preferences can help us reduce the complexity of some tasks, such as winner selection or manipulation. For that reason, research focuses on proposing metrics that express, how close the voters' preferences are to being structured. This is usually referred to as distance. We provide a general overview of current research on this topic, for more details, refer to the respective articles.

Faliszewski et al. [31] define multiple measures of distance to single-peaked preference. These include for example a number of voters not respecting the structure, the number of swaps in adjacent votes, or candidates needed to obtain single-peaked preferences. They also analyze the complexity of manipulation and control for a whole range of these nearly structured domains and show that some of them can be approximated or computed in polynomial time.

Erdélyi et al. [32] use voter and candidate swaps as well and they define other measures, such as the minimum of partitions of candidates we need so that each of these partitions is single-peaked. The authors show NP-completeness of measuring the distance to the single-peaked domain by all of their defined measures, except two of them. The first one, the number of candidates that need to be deleted to obtain single-peaked preference can be done in polynomial time and the second one, the previously mentioned candidate partition, the complexity of which remains open.

Elkind et al. [33] and Cornaz et al. [34] focus on the case where there are some very similar candidates, *clones*, which all voters rate consecutively but in different order. They measure the distance to single-peakedness (and single-crossingness for [33]) by the number of groups of clones [33] or by the size of the largest one [34]. Elkind et al. further study axiomatic properties and recognition of such preferences. Cornaz et al. focus on proportional representation in committee election with nearly single-peaked preferences.

Current literature focuses mainly on defining nearly single-peaked preferences, however, most of the defined measures could possibly be applied to nearly single-crossing preferences as well. We feel that there are a lot of research opportunities for other families of nearly structured preferences such as preference restrictions in approval election, since to the best of our knowledge, there are no articles focusing solely on them.

3.3 Commitee election

Finding a winning committee is NP-hard for many voting rules. Therefore, current research often focuses on committee election in restricted domains.

That is the case of Godziszewski et al. [26], where voters and candidates are mapped to 2D-Euclidean space and the voters' preferences are dependent on the points' distances. The authors study specific NP-hard approval committee rules and show that all of them remain NP-hard even for 2D-Euclidean preferences. The authors also create visualizations of some approval rules and research their properties. The authors conclude that voters' preferences are represented inaccurately if the voters have to approve a fixed number of candidates.

Pierczyński and Skowron [35] investigate *committees in the core* in an election with restricted preferences.

▶ **Definition 3.2.** *In committee election $E = (C, V)$, for committee size k, we say that a committee W is* in the core*, if for each $S \subset N$ and each subset of candidates T with $|T| \leq k \cdot \frac{|S|}{n}$ there is a voter $v \in S$ preferring W to T or rating them the same.*

Intuitively, a committee in the core should represent the voters proportionally in some way.

Pierczyński and Skowron find that the core is non-empty for a number of domain restrictions, such as voter interval, candidate interval, single-peaked, and single-crossing preferences. They also show that a number of known voting rules does not guarantee selecting a committee in the core even when used in election with restricted preferences.

3.4 OWA based committee election

In this section, we provide an overview of what is known about the so-called ordered weighted average (OWA) based rules. There are multiple possibilities, for how to use OWA to estimate a score of a committee. Our usage of OWA (for definition, see Section 2.2.2.1) is based on Bredereck et al. [36]. The same approach is also used by Skowron et al. [37].

It is worth noting that the terminology of some authors differs from our terminology introduced in Chapter 2. They use terms *agents* and *items* to denote voters and candidates. Furthermore, instead of a score given from voter to a committee in an election, they use the notion of *utility* that the voter (agent) derives when the committee (set of items) is selected. Apart from the terminology, our understanding of the problem does not differ.

Skowron et al. [37] define a number of OWA vector families and prove that finding OWA winner is NP-hard for every family of OWA vectors that are non-constant and non-increasing. They also propose an approximation algorithm for non-increasing OWA and the relation of certain OWA families to known NP-hard problems. Authors also focus on rules with *non-finicky utilities*, in simple terms, rules where every voter gives sufficiently high utilities to sufficiently many candidates. Skowron et al. propose an approximation algorithm for this type of utility restriction and certain OWA vectors.

Bredereck et al. [36] focus on parametrized algorithms and propose an FPT algorithm for arbitrary utility and non-increasing OWA vectors parametrized by the number of agents and committee size. They also show an FPT algorithm for OWAs that are constant except the first or last π values parametrized by π and the number of voters. Furthermore, they propose FPT algorithms for approval utility and selected OWA vectors.

Bredereck et al. prove that if there is an approximation algorithm for approval election and a non-negative family of OWAs parametrized by the number of agents, then there also exists an approximation algorithm for this family of OWAs and certain non-finicky utilities.

We intuitively describe the usage of OWAs in articles with a different approach, for the formal definition we refer to the original articles.

Elkind and Ismaili [38] suggest extensions of Chamberlin-Courant rule based on OWA. They compute utility derived from a committee as an ordered weighted average of utilities derived by individual voters.

Elkind and Ismaili propose a polynomial-time algorithm for single-peaked preferences, with OWA vector, that maximizes utilities of all voters except some fixed number of the least happy ones. This is expressed with an OWA vector $w = (0, \ldots, 0, \frac{1}{n-d}, \ldots, \frac{1}{n-d})$, where 0 is in the first d places (in their notation, voters are ordered in decreasing order, while in our notation, we order candidate by their score increasingly).

They also propose a polynomial-time algorithm under some constraints for OWA election for single-peaked and single-crossing preferences.

Amanatidis et al. [39] study multiple referenda and approval committee election. Multiple referenda is a process of making a collective decision over multiple binary issues. In this case, we can view it similarly as an approval committee election without a fixed committee size.

Amanatidis et al. work with binary vectors, i.e., every vote and committee (resp. referenda proposition) are expressed as a vector of 1 for the approved candidate (resp. approved proposition) and 0 for disapproved candidates (resp. disapproved proposition). We then compute the score of a committee as an ordered weighted average of its Hamming distances to votes.

Amanditis et al. prove NP-hardness for certain classes of OWA vectors for both constrained and unconstrained committee size. In the case of unconstrained committee size, i.e., multiple referenda, they show that NP-hardness holds even for balanced elections, where all the candidates are approved by exactly half of the voters. They also find polynomial-time and approximation algorithms for certain OWA vectors.

Chapter 4
Algorithms

In this chapter, we propose original algorithms for selecting a winning committee. We focus on OWA based voting systems and interval restricted domains. Specifically, we describe algorithms that select a winning committee for approval election with OWA vector $(0, \dots, 0, 1)$ and voter and candidate interval restricted preferences. There can be multiple winning committees, our algorithms guarantee to select one of them.

We choose OWA vector $(0, \dots, 0, 1)$ because it offers a great simplification. A committee obtains a point from a voter only if the voter approves all members of the committee. This helps us greatly to create efficient algorithms for selecting a winning committee.

We focus on election with approval preferences. We do not restrict the number of candidates that a voter can approve, however, in some algorithms, we describe the time complexity for both restricted and unrestricted number of candidates. This restriction occurs when voters vote by t-approval and considering it separately from the general case gives us a better estimate of the time complexity.

In Sections 4.2.1 and 4.3, we describe two approaches to selecting a winning committee. We describe multiple algorithms that solve the same task in different time. We find that useful because some algorithms can be more fitting to processing certain inputs. Furthermore, the algorithms build on different ideas and these ideas can be possibly developed into some more efficient algorithms or expanded to election with different OWA vectors and/or preference restrictions.

4.1 Preliminaries

For our algorithms, we assume that the input is in the form of a matrix (two dimensional array), where candidates correspond to columns and voters to rows. In the candidate (resp. voter) interval, the candidates (voters) are ordered along the given axis $\lhd$ (e.g., in the candidate interval domain, the candidate who is the first on the axis corresponds to the column indexed by 0). Column i row j then contains information whether voter j approves candidate i, i.e., 1 when voter j approves candidate i and 0 otherwise.

In Section 4.3, we further suppose the input to contain a list of intervals associated to voters (resp. candidates) in voter (resp. candidate interval). These intervals are expressed as pairs of integers, denoting the lower and upper bound of an interval. Integers contained in the interval correspond to the indices of voters (resp. candidates) in the input matrix. In the context of election, we refer to the lower bound and upper bound of an interval as to the leftmost and rightmost voter or candidate, respectively. We give an example of expected input in Figure 4.1.

By the *distance* of two voters a, b (resp. candidates) in voter (resp. candidate) interval, we mean the size of the interval $[a, b] = b - a + 1$.

		Candidates			
		0	**1**	**2**	**3**
Voters	**0**	1	0	0	0
	1	1	1	0	1
	2	0	1	1	1
	3	0	1	1	0

(a) Input matrix

```
L = ((0, (0,1)),
     (1, (1,3)),
     (2, (2,3)),
     (3, (1,2)))
```

(b) List of candidate intervals with their associated candidates

Figure 4.1 Input example for CI election

In our algorithms, we work with *intersection of a multiset of intervals.*

▶ **Definition 4.1.** *An* intersection of a multiset of intervals $\mathcal{M}$ *is an interval* $\bigcap_{I\in\mathcal{M}} I$.

It is worth noting that the intersection of a multiset of intervals is an interval including all elements that are contained in every interval of the multiset. It cannot be confused with an interval J such that $\forall I \in \mathcal{M}\colon J \subseteq I$. Interval J is not necessarily an intersection of $\mathcal{M}$ because there could be an element w such that $\forall I \in \mathcal{M}\colon w \in I$ and $w \notin J$. However, J is certainly a *common subset of a multiset of intervals.*

▶ **Definition 4.2.** *A* common subset of a multiset of intervals $\mathcal{M}$ *is a set J such that for all intervals $P \in \mathcal{M}$ it holds that $J \subseteq P$.*

From Definitions 4.1 and 4.2, we can deduce Observation 4.3.

▶ Observation 4.3. For all common subsets J of a multiset of intervals $\mathcal{M}$, J is a subset of the intersection of $\mathcal{M}$.

▶ **Definition 4.4.** Consecutive committee *in an election $E = (C, V)$ with voter interval restricted V with respect to $\lhd$ is a committee whose members form an interval on $\lhd$.*

▶ **Definition 4.5.** Consecutive voter group *in an election $E = (C, V)$ with candidate interval restriction with respect to $\lhd$ is a set of voters who form an interval on $\lhd$.*

4.2 Algorithms

We begin with some simple observations and a lemma.

▶ Observation 4.6. In approval election with OWA vector $(0, \ldots, 0, 1)$, a committee receives a point from a voter if and only if the voter approves all of its candidates.

▶ Observation 4.7. In voter interval domain with axis $\lhd$, for all candidates a, b, c and all voters v, if a voter v approves a, c and $a \lhd b \lhd c$, then v approves also b.

▶ Observation 4.8. In candidate interval domain with axis $\lhd$, for all voters v_1, v_2, v_3 and all candidates c, if c is approved by v_1, v_3 and $v_1 \lhd v_2 \lhd v_3$, then c is approved also by v_2.

▶ **Lemma 4.9.** *Let J be the set of all intervals that are intersections of any nonempty subset of intervals from a multiset of intervals P, i.e.,*

$$J := \left\{ \bigcap_{A\in P'} A \mid \emptyset \neq P' \subseteq P \right\} .$$

I is a set $\{[lb, ub] : [lb, x] \in P$ and $[y, ub] \in P$ for some $x, y\}$. Then $J \subseteq I$.

Informally, Lemma 4.9 states that for a list of intervals P, we can generate intersection intervals of all subsets of P, if we combine all lower bounds of intervals from P with all upper bounds. Or in other words, all intersections of all possible subsets of P are intervals $[lb, ub]$ where lb is the lower bound of some interval from P and ub is the upper bound of some interval from P.

Proof of Lemma 4.9. For contradiction, we suppose that there is an interval $[a, b] \in J$, where a is not a lower bound of an interval from P (case 1) or b is not an upper bound of an interval from P (case 2).

Case 1) If no interval starts in a, then interval $[a-1, b]$ intersects the same subset as $[a, b]$ and therefore it was not the longest intersection and we obtain a contradiction.

Case 2) Analogously for b and $[a, b+1]$. ◀

In the following, we introduce two types of algorithms for finding a winning committee that can both be specified for voter interval and candidate interval election.

We first intuitively describe the main ideas of these algorithms. First of them builds on the idea that in given restricted domains there will always be a winning committee that is consecutive (voter interval) or is elected by a consecutive voter group (candidate interval). The algorithm then iterates over all consecutive committees (resp. voter groups) and estimates their score.

The second algorithm works directly with intervals and their intersections. In voter interval, we find an intersection of length at least k that is an intersection of maximum voter intervals. In candidate interval, we find a longest intersection that intersects at least k candidate intervals.

Now, we look in more detail at both of these algorithms.

4.2.1 Consecutive group algorithms

First, we focus on voter interval domain where the algorithm is more straightforward than for candidate interval domain.

▶ **Theorem 4.10.** *For every committee approval election $E = (C, V)$ with committee size k, voter interval preferences with respect to $\lhd$, and OWA vector $(0, \dots, 0, 1)$, there exists a consecutive committee.*

Proof. To obtain a contradiction we suppose that in the given election all of the winning committees are not consecutive.

It implies that for every winning committee and its leftmost member c_ℓ and rightmost member c_r, there exists a candidate c who is not a member of the winning committee and $c_\ell \lhd c \lhd c_r$.

From the winning committees, we choose a committee O where the distance in $\lhd$ of the leftmost member c_ℓ and rightmost member c_r is minimal. We then create a new committee O' with a smaller distance and show that its score is higher or equal to the score of O, which is a contradiction.

We create O' from O by swapping the candidate c_ℓ with the candidate c. By Observation 4.7, we know that every voter who approves c_ℓ and c_r approves also c. Therefore, every voter who approved all members of O also approves all members of O' and by Observation 4.6 it follows that every voter who gives a point to O also gives a point to O'. Voters who do not approve c_ℓ and therefore do not give a point to O can possibly give 0 or 1 points to O'. It follows that O' has the same or greater score than O. If it has the same score, it is a winning committee with a smaller distance of the leftmost and rightmost member, which is a contradiction. If O' has a higher score, O was not a winning committee, which is a contradiction. Thus, the theorem follows. ◀

▶ **Corollary 4.11.** *Let $E = (C, V)$ be a committee approval election with committee size k, n voters, m candidates, and voter interval restricted preferences. In OWA based election with OWA vector $(0, \dots, 0, 1)$ we are able to find a winning committee in $\mathcal{O}((m-k)kn)$ time.*

Proof. We can find a winning committee by iterating over all consecutive committees and choosing one with the highest score. In the given election, there is $m - k$ possible consecutive committees. To calculate a score of a committee, we iterate over all of its members for every voter which can be done in $\mathcal{O}(kn)$ time. Therefore, the overall time complexity of finding a winning committee is $\mathcal{O}((m-k)kn)$. We provide pseudocode in Algorithm 1. ◀

Algorithm 1 Winning committee with consecutive voter group

```
Input: VI election, committee size $k$
Output: Winning committee
set max to $-\infty$, set $r$ to empty set
for all consecutive committees $p$ of length $k$ do
    set $s$ to score of committee $p$
    if $s >$ max then
        set max to $s$, $r$ to $p$
    end if
end for
return $r$
```

Correctness and time complexity of Algorithm 1 follows from Theorem 4.10 and Corollary 4.11.

Now, we propose a similar algorithm for candidate interval election. In this algorithm, we work with a consecutive group of voters. Naturally, we do not know how many points the winning committee obtains and therefore how big the consecutive group of voters should be. Therefore in this algorithm we iterate over possible voter group sizes and stop once we reach a voter group which elects a committee of size at least k.

First, we prove that for a fixed integer s there is always a committee with a maximal score at most s which gets all its points from a consecutive voter group.

▶ **Theorem 4.12.** *Let $E = (C, V)$ be an approval election with candidate interval preferences, committee size k, and fixed $s \in \mathbb{N}$. Then, there exists a consecutive voter group $Q \subseteq V, |Q| = s$, and a committee W of size k such that the score of W in election $E = (C, Q)$ is greater or equal to the score of any committee of size k in election $E = (C, V')$ for $V' \subseteq V$ with $|V'| = s$.*

This means that if we fix the maximal number of voters who can give points to a committee, there is always a consecutive group of voters selecting a committee with the maximal score.

Proof of Theorem 4.12. Suppose towards a contradiction that for all committees selected by a group of size s with a maximal score, there is no committee, which would be elected by a consecutive group of voters. We take one such committee D and a set of voters A that select this committee such that the distance between the leftmost voter v_ℓ and rightmost voter v_r in A is minimal.

Then, there is a voter $v \in V, v \notin A$ between the v_ℓ and v_r. By Observation 4.8, every candidate approved by v_ℓ and v_r is approved by v as well. Therefore, there exists a voter set $A' = A - \{\ell\} + \{v\}$, such that the score of D in election $E = (C, A')$ is the same or greater than in an election $E = (C, A)$.

If the score is greater, we have a contradiction that the score of D in $E = (C, A)$ was maximal. If the score is the same, we have a contradiction with the minimality of distance of v_ℓ and v_r. Thus, the theorem follows. ◀

However, when we are finding a committee in an election, we do not know how big the consecutive voter group of the winning committee is (i.e., how many points the committee receives). Therefore, we propose an algorithm that iterates over the possible sizes of voter groups. It starts

Algorithm 2 Winning committee with consecutive voter group

```
Input: CI election, committee size $k$
Output: Superset of a winning committee
for integer $s$ from $n$ to 1 do
    for all consecutive voter groups $V'$ of length $s$ do
        set $J$ to a set of candidates who receive points from all members of $V'$
        if $|J| < k$ then
            continue with next iteration
        end if
        return $J$
    end for
end for
```

from the greatest size and once it reaches a voter group that selects at least k-sized committee, it stops. The algorithm is described in pseudocode in Algorithm 2.

▶ **Claim 4.13.** *Algorithm 2 is correct.*

Proof. By Theorem 4.12, there is a consecutive voter group selecting a committee with the maximal score for a fixed voter group size. In Algorithm 2, we iterate over possible voter group sizes from the largest possible and we iterate over all consecutive voter groups of such size. If we find a committee J of size at least k for the given voter group, it means, that this committee has a maximal score for the given size of the voter group.

With a voter group size s, a committee can obtain maximum s points. Because we iterate from the largest possible voter group size, the committee J has also a maximal overall score, i.e., it is a winning committee. ◀

▶ **Claim 4.14.** *Algorithm 2 returns a winning committee in $\mathcal{O}(mn^3)$ time.*

Proof. The algorithm iterates over all voter groups of sizes from n to 1. In each iteration for a voter group size i, there are $n - (i + 1)$ voter groups. In each of these iterations, we also calculate which candidates receive points from this voter group, therefore we iterate over all of the candidates. We express this as

$$\sum_{i=1}^{n} mi \cdot (n - (i + 1))) = \frac{1}{6}mn(n^2 - 3n - 4) ,$$

therefore, the time complexity of this algorithm is $\mathcal{O}(\frac{1}{6}mn(n^2 - 3n - 4)) \subseteq \mathcal{O}(mn^3)$. ◀

Notice that Algorithm 2 calculates committees with the maximal score for certain subsections of voters repeatedly. This leads to the idea of using dynamic programming to improve the time complexity of the algorithm.

When checking which candidates are approved by the whole voter group $[a, b], a \neq b$, it is equivalent to finding an intersection of candidates being approved by intervals $[a, a]$ and $[a+1, b]$. This gives an idea of a recursive algorithm.

The base case is voter group $[a, a]$ where we have to iterate through candidates and find which of them are approved by voter associated to position a. Committees for other voter groups $[a, b], a < b$ are then calculated as intersection of committees of $[a, a]$ and $[a + 1, b]$.

Because some values are computed repeatedly, we use memoization. We use a $n \times n$ table T where entry $T[i][j]$ stores a set of candidates approved by a consecutive voter group $[i, j]$.

The algorithm then works similarly to Algorithm 2, we iterate over all possible voter group sizes from the largest one and we iterate over all consecutive voter groups of such size. Once we find a committee of size at least k for the given voter group, we return it as a winning committee.

We describe the algorithm in pseudocode in Algorithm 4, it uses a recursive function described in Algorithm 3.

Algorithm 3 Approved candidates for given voter group

```
Input: Integers $i, j$ denoting consecutive voter group
Output: Candidates approved by the given consecutive voter group
function FindCommitteeRecursive($i$, $j$)
    if $T[i][j]$ is defined then
        return $T[i][j]$
    end if
    if $i == j$ then
        set $T[i][j]$ to be the set of candidates approved by $i$
        return $T[i][j]$
    end if
    set $T[i][j]$ to be an intersection of FindCommitteeRecursive($i, i$) and FindCommitteeRecursive($i + 1, j$)
    return $T[i][j]$
end function
```

Algorithm 4 Winning committee with consecutive voter group with memoization

```
Input: CI election, committee size $k$
Output: Winning committee
for integer $s$ from $n$ to 1 do
    for all $i, j \in \mathbf{N}, j - i + 1 = s, i \geq 1, j \leq n$ do
        set $J$ to FindCommitteeRecursive($i, j$)
        if $|J| < k$ then
            continue with next iteration
        end if
        return $J$
    end for
end for
```

▶ **Claim 4.15.** *Algorithm 4 is correct.*

Proof. Algorithm 4 finds a winning committee in the same way as Algorithm 2 correctness of which we have already proven. The only difference in these algorithms is in finding which candidates are approved by any consecutive voter group $[i, j]$.

For voter group $[i, i]$, both algorithms use the same method, they iterate through the candidates. For $[i, j]$, Algorithm 4 finds an intersection of candidates $[i, i]$ and $[i + j, j]$ which is equivalent to finding the candidates directly as in Algorithm 2.

Both algorithms are equivalent and therefore Algorithm 4 finds a winning committee. ◀

We state and prove the complexity of Algorithm 4 for two cases. One of them is the general case with no further restrictions. The second case is for a fixed number of candidates that a voter approves. We emphasize this case because it gives a better estimate of time complexity in t-approval election.

▶ **Claim 4.16.** *Algorithm 4 runs in $\mathcal{O}(mn^2)$ time or in $\mathcal{O}(tn^2)$ if the number of candidates approved by one voter is fixed as t.*

Proof. We fill a table T of size $n \times n$. Because voter groups $[i, j]$ where $i > j$ are invalid, we fill in table T only entries where $i \leq j$.

For calculating entries $T[i][i]$, we iterate over all m candidates and we fill at most n such entries.

For calculating entries $T[i][j], i < j$, we iterate over sets of candidates in two other entries of the table, i.e., over two sets of at most m candidates.

For a fixed number of approved candidates by a voter t, we iterate over two sets of at most t candidates.

There is no more than $\frac{1}{2}n^2$ such entries $T[i][j]$ that $i < j$.

Therefore, the overall time complexity is $\mathcal{O}(mn+\frac{1}{2}mn^2) \subseteq \mathcal{O}(mn^2)$ or $\mathcal{O}(mn+\frac{1}{2}tn^2) \subseteq \mathcal{O}(tn^2)$ for fixed number t of approved candidates. ◀

4.3 Interval algorithms

In this section, we show two algorithms for finding a winning committee in each candidate and voter interval. These algorithms build on the idea of finding an intersection of candidate (resp. voter) intervals (for the definition of an intersection of a multiset of intervals, see Definition 4.1). For these algorithms, we expect input in the form of an election matrix and a list of candidate (resp. voter) intervals.

We begin with the election with candidate interval. Roughly speaking, Algorithms 6 and 7 for candidate interval find a set of at least k candidates, such that the associated candidate intervals have the largest intersection. A winning committee is then any k-element subset of this set of candidates.

Before proposing the algorithm, we first prove that candidates with the largest candidate interval intersection indeed form a winning committee.

▶ **Theorem 4.17.** *Let $E = (C, V)$ be a committee approval election with committee size k and candidate interval preferences. Let $\mathcal{W}$ be the set of winning committees for E in OWA based election with OWA vector $(0, \dots, 0, 1)$.*

Let J be a subset of C, such that $|J| \geq k$ and the size of the intersection of the candidate interval set associated to J is maximal. Then every subset $W \subseteq J$ of size k belongs to $\mathcal{W}$.

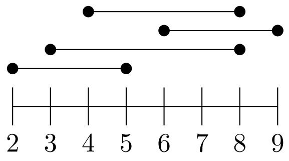
Proof. By Observation 4.6, a committee receives as many points as the number of candidates who approve all of its members. The intersection of candidate intervals associated to J is a set of voters who approve all members of J. Since we select the intersection to be maximal, candidates in J are approved by a maximum number of voters and have therefore the highest score. Since candidates from J are approved by the same voters, any k-element subset of J is a winning committee. ◀

Theorem 4.17 gives us an idea of how to find a winning committee. However, we first need an algorithm that from a multiset of intervals finds a subset of size at least k with the largest intersection. From Lemma 4.9, we know how to generate all possible interval intersections of a multiset of intervals. We do that by combining all lower bounds with all upper bounds of intervals from the given set, see pseudocode in Algorithm 5. To help with a better understanding of the algorithm, we demonstrate a few iterations of it in Example 4.

Algorithm 5 Largest intersection of a list of intervals

```
    Input: list $P$ of intervals, integer $j$
    Output: list of $j$ intervals, length of their intersection
set $G$ to a set of all possible interval intersections of list $P$
set max to 0, $R$ to empty list
for intervals $g$ in $G$ do
    set $I$ to empty list
    for intervals $p$ in $P$ do
        if $g$ is a subset of $p$ then
            append $p$ to $I$
        end if
    end for
    if $|I| \geq j$ and size of $g$ is greater than max then
        set max to size of $g$ and $R$ to $I$
    end if
end for
return $R$, max
```

▶ **Example 4.** We show a few iterations of Algorithm 5 for the input on the following intervals.



Input variables have the following values.

```
P = ((2,5), (3,8), (6,9), (4,8))
j = 2
```

In the first step, we generate a set of possible interval intersections. We omit invalid intervals and include each interval only once. We set values to max and R. We obtain the following.

```
G = ((2,5),(2,9),(2,8),(3,5),(3,9),(3,8),(6,9),(6,8),(4,5),(4,9),(4,8))
max = 0
R = ()
```

Now, for the first element g of G, we iterate over P and append intervals from p that g is a subset of to list I. After iterating over P, we have the following.

```
g = (2,5)
I = ((2,5))
```

Since the size of I is smaller than 2, we do not change variables max, R and we continue with the next iteration for the next element of G. We get the following.

```
g = (2,9)
I = ()
```

Again, max and R are not changed and this repeats also for the next element of G; therefore, we skip this iteration. We show the state of variables for the fourth element of G.

```
g = (3,5)
I = ((2,5), (3,8))
```

Finally, I has sufficient size (i.e., greater than 2) to change the values of max and R to the following.

```
max = 3
R = ((2,5), (3,8))
```

The following iterations work similarly; therefore, we fast forward to the last step of the algorithm. Variables R and max remain unchanged and the algorithm outputs the following.

```
max = 3
R = ((2,5), (3,8))
```

Please note that later on, we use the algorithm for committee selection. In that case, the set P is a set of candidate intervals with the associated candidates. We omitted the associated candidates here for the sake of simplicity. Having P as a list of intervals amended with the information about associated candidates (see, e.g., Figure 4.1b) does not change the principle of the algorithm.

▶ **Claim 4.18.** *Algorithm 5 finds a list of at least j intervals with the largest intersection.*

Proof. From Lemma 4.9 we generate a set P of all possible intersections. We then iterate over this set and for each possible intersection g find a list of intervals I such that it intersects all of them. From all g that are subsets of at least k-sized I, we select the largest one. That means that g is indeed in the intersection of I (there is no element w such that w is a subset of all intervals of I and w is not in g, see Definition 4.1). Since we choose g to be maximal, it is also the largest of all intersections. ◀

▶ **Claim 4.19.** *Algorithm 5 runs in $\mathcal{O}(b^3)$ time, where $b = |P|$.*

Proof. We set $b = |P|$.

We generate a list of no more than b^2 possible intersections. For each of these possible intersections, we iterate over all the b intervals of the given list P. In each of these iterations, we check, if an interval is a subset of another interval, this is done in constant time. We also compare the lengths of found intersections to the current maximum but that is only a constant number of operations. Therefore, the overall time complexity is $\mathcal{O}(b^3)$. ◀

Now, we can continue with the committee finding algorithm. We introduce two different approaches for using Algorithm 5.

In the first of them, Algorithm 6, we simply apply Algorithm 5 to the list of all candidate intervals and input parameter k. We find the largest intersection of at least k of these intervals. The chosen multiset of candidate intervals has candidates associated to them. We select any k of them as the winning committee.

Please note that Algorithm 5 returns two values, a list of intervals and the size of their intersection. In Algorithm 6, we use only the first of the output values, the list of intervals. However, we use both of the output values in Algorithm 7.

▶ **Claim 4.20.** *Algorithm 6 finds a winning committee.*

Proof. Algorithm 6 finds a list of k candidates, with the largest intersection of voters. The intersection corresponds to the number of voters giving a point to this list of candidates. Since this intersection is maximal, the score of the list of k candidates is maximal and it is therefore a winning committee.

If there is no k-sized committee with an intersection, all committees have no points. Therefore all committees are winning and we can output any of them. ◀

▶ **Claim 4.21.** *Algorithm 6 runs in $\mathcal{O}(m^3)$ time.*

Algorithm 6 Winning committee in CI election through largest intersection

```
Input: CI election, committee size $k$, list of candidates and their associated intervals
Output: Winning committee
apply Algorithm 5 with the list of candidate intervals and $k$ as parameters, set $G$ to be the output, discard second output value
if $G$ is empty then
    return any $k$ candidates
end if
return any $k$ candidates associated to intervals in $G$
```

Proof. There are at most m candidate intervals, we apply Algorithm 5 to them. That takes $\mathcal{O}(m^3)$ time.

From the output intervals, we select k associated candidates. We suppose that the intervals contain information about the associated candidate; therefore, we can simply select the first k candidates. That can be done in constant time.

Therefore the overall time complexity is $\mathcal{O}(m^3)$. ◀

We now propose the second algorithm for finding a winning committee in candidate interval election through the intersection of intervals. Algorithm 7 iterates over all voters and for each voter v finds a committee with a maximal score that also obtains a point from v. A winning committee is then a committee with the overall maximal score.

For finding a committee with the maximal score for a fixed voter v, we use Algorithm 5. We take a list of candidate intervals associated to candidates approved by v and apply Algorithm 5 to this list (with parameter k). This outputs the committee with the maximal score that receives a point from v. We describe the Algorithm 7 in pseudocode.

Algorithm 7 Winning committee in CI election through largest intersection over voters

```
Input: CI election, committee size $k$, list of candidates and their associated intervals
Output: winning committee
set max to 0, $R$ to empty list
for voter $v$ in $V$ do
    if $v$ approves less than $k$ candidates then
        continue with next iteration
    end if
    set $A$ as the list of candidate intervals associated to candidates approved by $v$
    apply Algorithm 5 to $A$ and $k$, denote output as $K, g$
    if $g >$ max then
        set $max$ to $g$ and $R$ to $K$
    end if
end for
if $R$ is empty then
    return any $k$ candidates
end if
return first $k$ intervals of $R$
```

▶ **Claim 4.22.** *Algorithm 7 finds a winning committee.*

Proof. For each voter v, we find a list of at least k candidates with the largest intersection of voters who approve them. This corresponds to the score of these k candidates. We then select a committee with the overall maximal intersection, i.e., a committee with the maximal score.

If there is no list of k candidates with an intersection, all committees receive no points. Therefore all committees are winning and we can output any of them. ◀

In the same way as for Algorithm 4, we express the time complexity of Algorithm 7 in two ways. One is for the general case with no other restrictions, the other is for a fixed number of candidates approved by one voter. The second case gives a better estimate of time complexity for t-approval election.

▶ **Claim 4.23.** *Algorithm 7 runs in $\mathcal{O}(m^3n)$ time or in $\mathcal{O}(nt^3 + mn)$ time if the number of candidates approved by one voter is fixed as t.*

Proof. We iterate over n voters. For each of them, we need to find a list A of candidate intervals. We do that by iterating over all m candidates.

There are n iterations of applying Algorithm 5 to A. In the general case, A has size at most m, in the special case with a fixed number of approved candidates, it has size t. This has time complexity $\mathcal{O}(m^3n)$ or $\mathcal{O}(nt^3)$ for fixed number of approved candidates.

The overall time complexity is then $\mathcal{O}(m^3n + mn) \subseteq \mathcal{O}(m^3n)$ or $\mathcal{O}(nt^3 + mn)$ for a fixed number of approved candidates. ◀

We proposed two algorithms for candidate interval election. Now, we describe two algorithms for voter interval election that select the winning committee in a similar manner. These are Algorithm 9 and Algorithm 10.

The main idea of these algorithms is to find a multiset of voter intervals, such that its size is maximal and the length of their common subset interval is at least k. Any k candidates contained in this common subset interval are then a winning committee. We state this formally in Theorem 4.24.

▶ **Theorem 4.24.** *Let $E = (C, V)$ be a committee approval election with committee size k and voter interval preferences. Let W be the set of winning committees for E in an OWA based election with OWA vector $(0, \dots, 0, 1)$.*

Let J be a subset of V such that $|J|$ is maximal and there is a set i such that for all $j \in J$ it holds that $i \subseteq j$ and $|i| \geq k$. Then $\forall g \subseteq i, |g| = k \colon g \in W$.

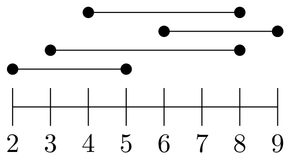
Proof. Common subset i corresponds to candidates approved by all voters from J. If we select a committee as any k members of i, it receives points from all voters in J by Observation 4.6. Since $|J|$ is maximal, the committee has a maximal score and therefore is a winning committee. ◀

Now, we have a general idea of how to construct algorithms for finding a winning committee in voter interval election. First, we describe Algorithm 8 that from the given multiset of intervals selects a subset of maximal size with a common subset of size at least j. We demonstrate a few iterations of the algorithm in Example 5.

Algorithm 8 j-sized common subset of a largest list of intervals

```
Input: list $P$ of intervals, integer $j$
Output: at least $j$-sized common subset of the largest multiset of intervals from $P$
set $G$ to a set of all possible interval intersections of list $P$, such that the intervals are valid
    and at least size of $j$
set max to 0, $r$ to empty set
for intervals $g$ in $G$ do
    for intervals $p$ in $P$ do
        set $I$ to empty list
        if $g \subseteq p$ then
            append $p$ to $I$
        end if
    end for
    if $|I| >$ max then
        set max to $|I|$, set $r$ to $g$
    end if
end for
return $r$
```

▶ **Example 5.** We show a few iterations of Algorithm 8 for the input of the following intervals.



Input variables have the following values.

```
P = ((2,5), (3,8), (6,9), (4,8))
j = 2
```

We generate set G of possible intersections, include each interval only once, omit invalid intervals and intervals smaller than 2. We set values to variables max and r. We get the following.

```
G = ((2,5),(2,9),(2,8),(3,5),(3,9),(3,8),(6,9),(6,8),(4,9),(4,8))
max = 0
r = ()
```

Now, for element $g = (2, 5)$, we iterate over intervals in P. We append to I all intervals p such that g is a subset of p. After iterating over all intervals p of P we have the following.

```
g = (2,5)
I = ((2,5))
```

The size of I is greater than max which was previously set to 0. Therefore we update values of max and r.

```
max = 1
r = (2,5)
```

Other iterations work in the same manner and we only update max and r if we find a greater list I. That happens for intervals $(3,5)$ and $(6,8)$. We show the state of variables only for the latter.

```
g = (6,8)
I = ((2,8),(6,9),(4,8))
```

Since the previous value of max is 2, we reassign the variables to get the following.

```
max = 3
r = (6,8)
```

There is no greater common subset of a set of intervals P, therefore these values remain unchanged and are returned as the output of the algorithm.

▶ **Claim 4.25.** *Algorithm 8 finds at least j-sized common subset of the maximal possible subset of the given intervals.*

Proof. We iterate over all possible j-sized intersections of the list of voter intervals and select the one, where the list size is the largest. By Observation 4.3, for all common subsets of a multiset of intervals, there is an intersection of this same multiset. Therefore, iterating over all possible intersections is sufficient to find a common subset of the largest possible list of intervals. ◀

It is worth noting that Algorithm 8 does not necessarily output an intersection of any list of intervals. We iterate through possible intersections but since we do not check for its maximality, it may only be a common subset. However, the common subset has a size at least j which is sufficient for us.

▶ **Claim 4.26.** *Algorithm 8 runs in $\mathcal{O}(b^3)$ time where $b = |P|$.*

Proof. Let $b = |P|$.

We create a set G of no more than b^2 intervals. For each interval of G, we iterate over all b intervals of P and perform a constant number of operations on them.

The time complexity is then $\mathcal{O}(b^3)$. ◀

We now introduce two different approaches for using Algorithm 8 for finding a winning committee in voter interval election.

The first of them is Algorithm 9. This algorithm simply applies Algorithm 8 to all voter intervals. We then select any k candidates from the output as the winning committee.

Algorithm 9 Winning committee in VI election through largest intersection

```
Input: VI election, committee size $k$, list of voters, and their associated intervals
Output: Winning committee
apply Algorithm 8 with the list of voter intervals and $k$ as parameters, set $g$ to be the output
if $g$ is empty then
    return any $k$ candidates
end if
return any $k$ candidates of $g$
```

▶ **Claim 4.27.** *Algorithm 9 returns a winning committee.*

Proof. Algorithm 9 finds a committee of k candidates that are a subset of the maximum of voter intervals. That means that a maximum of voters approve all of these k candidates and therefore the committee has the maximal score. ◀

▶ **Claim 4.28.** *Algorithm 9 runs in $\mathcal{O}(n^3)$ time.*

Proof. There are at most n voter intervals, we apply Algorithm 8 to them. That takes $\mathcal{O}(n^3)$ time, other operations are in constant time. The overall complexity is, therefore, $\mathcal{O}(n^3)$. ◀

We now describe the second approach to the use of Algorithm 8. It is somewhat similar to Algorithm 7. We iterate over all candidates and for each candidate c, we use Algorithm 8 to find a committee containing candidate c with the maximal score. We then select from these committees one with the maximum score.

We describe the algorithm in pseudocode in Algorithm 10.

Algorithm 10 Winning committee in VI election through largest intersection over voters

```
    Input: VI election, committee size $k$, list of voters and their associated intervals
    Output: winning committee
set max to 0, set $r$ to empty set
for candidate $c$ in $C$ do
    set $A$ as the list of voter intervals associated to voters who approve $c$
    apply Algorithm 8 to $A$ and $k$, denote output as $s$
    if $s >$ max then
        set max to $|s|$, set $r$ to $s$
    end if
end for
if $s$ is empty then
    return any $k$ candidates
end if
return any $k$ candidates from $r$
```

▶ **Claim 4.29.** *Algorithm 10 finds a winning committee.*

Proof. For each candidate, we find a committee containing this candidate, such that the committee is a common subset of a list of voter intervals of maximal size. The committee receives points from all voters associated to this list of intervals, therefore the committee has the maximal score.

We then select a committee with an overall maximal score. ◀

▶ **Claim 4.30.** *Algorithm 10 runs in $\mathcal{O}(mn + n^3)$ time.*

Proof. Algorithm 10 iterates over all m candidates.

For each of the candidates c, we find a list A of voters approving c. For that, we iterate over all n voters.

We apply Algorithm 8 to A, which has a size at most n. Time complexity is therefore $\mathcal{O}(n^3)$.

The overall time complexity is then $\mathcal{O}(mn + n^3)$. ◀

4.4 Summary

We introduced multiple algorithms for finding a winner in voter and candidate interval restricted preferences. We analyzed the time complexity of these algorithms and we provide an overview in Tables 4.1 and 4.2.

	Time complexity
Algorithm 1	$\mathcal{O}((m-k)kn)$
Algorithm 9	$\mathcal{O}(n^3)$
Algorithm 10	$\mathcal{O}(mn+n^3)$

Table 4.1 Time complexity of algorithms for voter interval restriction

	Time complexity
Algorithm 2	$\mathcal{O}(mn^3)$
Algorithm 4	$\mathcal{O}(mn^2)$, $\mathcal{O}(tn^2)$
Algorithm 6	$\mathcal{O}(m^3)$
Algorithm 7	$\mathcal{O}(m^3n)$, $\mathcal{O}(nt^3+mn)$

Table 4.2 Time complexity of algorithms for candidate interval restriction

We examined a special case of domain restriction and OWA vector. To our knowledge, there is no algorithm in the current literature to which we could directly compare our results.

Bredereck et al. [36] describe an algorithm for OWA approval election with vector $(0,\dots,0,1)$ and unrestricted preferences. However, they focus on parametrized algorithms and they describe an algorithm parametrized by the number of agents. Their algorithm simply tries to guess a large enough subset of voters who would give points to a committee of sufficient size. The idea is similar to our approach for candidate interval restricted preferences. In Algorithms 2, 4, 6 and 7 we select a maximal voter group that gives points to a committee of sufficient size. Unlike Bredereck et al., we do not guess this voter group and we find it in polynomial time instead. That is possible due to restricted preferences.

Our only other point of comparison is a naive approach to estimating the score of all committees. For m candidates and committee size k, there are $\binom{m}{k}$ possible committees. For calculating their score, we would need to iterate over all n voters. The time complexity is then $\mathcal{O}(\binom{m}{k}n)$. This naive approach could be used for small values of k. However, our algorithms run in polynomial time and therefore clearly perform better for the general size of the input.

Chapter 5
Conclusion

The goal of this thesis was to review the current literature on committee election and structured preferences. The focus of the thesis was on OWA election. The aim was to propose algorithms for certain combinations of OWA vectors and structured preferences.

In Chapter 2 we described elections and introduced related terminology. We described a number of voting systems for single- and multi-winner election. We introduced OWA based voting systems, a committee scoring system that uses an ordered weighted average to calculate a score of a committee. We also described how the preferences of voters may be structured which may help to find a winner more efficiently.

In Chapter 3, we reviewed a number of articles researching election. We summarized research related to structured and almost structured preferences. We investigated committee scoring rules in restricted domains and the usage of ordered weighted average in voting systems.

In Chapter 4, we introduced several polynomial-time algorithms for winner selection for OWA vector $(0, \ldots, 0, 1)$ and interval restricted preferences. We described two main approaches to winner selection and we find that modifying the proposed algorithms for other combinations of OWA vectors and structured preferences could be an interesting topic for further work.

Bibliography

1. ROTHE, Jörg (ed.). Economics and computation. In: 1st ed. Berlin, Germany: Springer, 2015, pp. 1–8, 197–391. Springer Texts in Business and Economics. ISBN 978-3-662-47903-2.
2. ELKIND, Edith; FALISZEWSKI, Piotr; SKOWRON, Piotr; SLINKO, Arkadii M. Properties of Multiwinner Voting Rules. *CoRR*. 2015, vol. abs/1506.02891. Available from arXiv: `1506.02891`.
3. ELKIND, Edith; LACKNER, Martin; PETERS, Dominik. *Preference Restrictions in Computational Social Choice: A Survey*. 2022. Available from arXiv: `2205.09092 [cs.GT]`.
4. FALISZEWSKI, Piotr. Committee Scoring Rules: A Call to Arms. In: 2017, pp. 5121–5125. Available from DOI: `10.24963/ijcai.2017/734`.
5. CONDORCET, Nicolas de. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge University Press, 2014. Cambridge Library Collection - Mathematics. Available from DOI: `10.1017/CBO9781139923972`.
6. COPELAND, Arthur H. *A reasonable social welfare function*. 1951. Tech. rep. Mimeo, University of Michigan USA.
7. FALISZEWSKI, Piotr; HEMASPAANDRA, Edith; HEMASPAANDRA, Lane A.; ROTHE, Jörg. Llull and Copeland Voting Computationally Resist Bribery and Control. *CoRR*. 2008, vol. abs/0809.4484. Available from arXiv: `0809.4484`.
8. MAREŠ, Martin; VALLA, Tomáš. *Průvodce labyrintem algoritmů*. 2nd ed. CZ.NIC, 2022. ISBN 978-80-88168-66-9.
9. CHAMBERLIN, John R.; COURANT, Paul N. Representative Deliberations and Representative Decisions: Proportional Representation and the Borda Rule. *American Political Science Review*. 1983, vol. 77, no. 3, pp. 718–733. Available from DOI: `10.2307/1957270`.
10. MONROE, Burt L. Fully Proportional Representation. *American Political Science Review*. 1995, vol. 89, no. 4, pp. 925–940. Available from DOI: `10.2307/2082518`.
11. ROBERTS, Kevin W.S. Voting over income tax schedules. *Journal of Public Economics*. 1977, vol. 8, no. 3, pp. 329–340. ISSN 0047-2727. Available from DOI: `https://doi.org/10.1016/0047-2727(77)90005-6`.
12. MELTZER, Allan H.; RICHARD, Scott F. A Rational Theory of the Size of Government. *Journal of Political Economy* [online]. 1981, vol. 89, no. 5, pp. 914–927 [visited on 2023-04-07]. ISSN 00223808, ISSN 1537534X. Available from: `http://www.jstor.org/stable/1830813`.
13. ELKIND, Edith; LACKNER, Martin; PETERS, Dominik. *Preference Restrictions in Computational Social Choice: A Survey*. 2022. Available from arXiv: `2205.09092 [cs.GT]`.

14. FALISZEWSKI, Piotr; HEMASPAANDRA, Edith; HEMASPAANDRA, Lane A.; ROTHE, Jörg. The Shield that Never Was: Societies with Single-Peaked Preferences are More Open to Manipulation and Control. *CoRR*. 2009, vol. abs/0909.3257. Available from arXiv: `0909.3257`.

15. BRANDT, F.; BRILL, Markus; HEMASPAANDRA, Edith; HEMASPAANDRA, Lane A. Bypassing Combinatorial Protections: Polynomial-Time Algorithms for Single-Peaked Electorates. *J. Artif. Intell. Res.* 2010, vol. 53, pp. 439–496. Available from DOI: `https://doi.org/10.1613/jair.4647`.

16. ELKIND, Edith; LACKNER, Martin. Structure in Dichotomous Preferences. *CoRR*. 2015, vol. abs/1505.00341. Available from arXiv: `1505.00341`.

17. FULKERSON, Delbert Ray; GROSS, Oliver Alfred. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*. 1965, vol. 15, pp. 835–855. Available from DOI: `10.2140/pjm.1965.15.835`.

18. BOOTH, Kellogg S.; LUEKER, George S. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*. 1976, vol. 13, no. 3, pp. 335–379. ISSN 0022-0000. Available from DOI: `https://doi.org/10.1016/S0022-0000(76)80045-1`.

19. MEIDANIS, João; PORTO, Oscar; TELLES, Guilherme P. On the consecutive ones property. *Discrete Applied Mathematics*. 1998, vol. 88, no. 1, pp. 325–354. ISSN 0166-218X. Available from DOI: `https://doi.org/10.1016/S0166-218X(98)00078-X`. Computational Molecular Biology DAM - CMB Series.

20. HABIB, Michel; MCCONNELL, Ross; PAUL, Christophe; VIENNOT, Laurent. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*. 2000, vol. 234, no. 1, pp. 59–84. ISSN 0304-3975. Available from DOI: `https://doi.org/10.1016/S0304-3975(97)00241-7`.

21. MCCONNELL, Ross M. A certifying algorithm for the consecutive-ones property. In: MUNRO, J. Ian (ed.). *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*. SIAM, 2004, pp. 768–777. Available from DOI: `10.1145/982792.982909`.

22. BARTHOLDI, John; TRICK, Michael A. Stable matching with preferences derived from a psychological model. *Operations Research Letters*. 1986, vol. 5, no. 4, pp. 165–169. ISSN 0167-6377. Available from DOI: `https://doi.org/10.1016/0167-6377(86)90072-6`.

23. DOIGNON, J.P.; FALMAGNE, J.C. A Polynomial Time Algorithm for Unidimensional Unfolding Representations. *Journal of Algorithms*. 1994, vol. 16, no. 2, pp. 218–233. ISSN 0196-6774. Available from DOI: `https://doi.org/10.1006/jagm.1994.1010`.

24. ESCOFFIER, Bruno; LANG, Jérôme; ÖZTÜRK, Meltem. Single-Peaked Consistency and Its Complexity. In: *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*. NLD: IOS Press, 2008, pp. 366–370. ISBN 9781586038915.

25. BREDERECK, Robert; CHEN, Jiehua; WOEGINGER, Gerhard. A characterization of the single-crossing domain. *Social Choice and Welfare*. 2013, vol. 41. Available from DOI: `10.1007/s00355-012-0717-8`.

26. GODZISZEWSKI, Michał; BATKO, Paweł; SKOWRON, Piotr; FALISZEWSKI, Piotr. An Analysis of Approval-Based Committee Rules for 2D-Euclidean Elections. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021, vol. 35, pp. 5448–5455. Available from DOI: `10.1609/aaai.v35i6.16686`.

27. KNOBLAUCH, Vicki. Recognizing one-dimensional Euclidean preference profiles. *Journal of Mathematical Economics*. 2010, vol. 46, no. 1, pp. 1–5. ISSN 0304-4068. Available from DOI: `https://doi.org/10.1016/j.jmateco.2009.05.007`.

28. ELKIND, Edith; FALISZEWSKI, Piotr. Recognizing 1-Euclidean Preferences: An Alternative Approach. In: LAVI, Ron (ed.). *Algorithmic Game Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 146–157. ISBN 978-3-662-44803-8.

29. PETERS, Dominik. Recognising Multidimensional Euclidean Preferences. *CoRR*. 2016, vol. abs/1602.08109. Available from arXiv: `1602.08109`.

30. ELKIND, Edith; LACKNER, Martin; PETERS, Dominik. Structured Preferences. In: 2017.

31. FALISZEWSKI, Piotr; HEMASPAANDRA, Edith; HEMASPAANDRA, Lane A. The Complexity of Manipulative Attacks in Nearly Single-Peaked Electorates. *Artif. Intell.* 2014, vol. 207, no. C, pp. 69–99. ISSN 0004-3702.

32. ERDÉLYI, Gábor; LACKNER, Martin; PFANDLER, Andreas. Computational Aspects of Nearly Single-Peaked Electorates. *CoRR*. 2012, vol. abs/1211.2627. Available from arXiv: `1211.2627`.

33. ELKIND, Edith; FALISZEWSKI, Piotr; SLINKO, Arkadii M. Clone structures in voters' preferences. *ArXiv*. 2011, vol. abs/1110.3939.

34. CORNAZ, Denis; GALAND, Lucie; SPANJAARD, Olivier. Bounded Single-Peaked Width and Proportional Representation. In: *Proceedings of the 20th European Conference on Artificial Intelligence*. Montpellier, France: IOS Press, 2012, pp. 270–275. ECAI'12. ISBN 9781614990970.

35. PIERCZYNSKI, Grzegorz; SKOWRON, Piotr. Core-Stable Committees under Restricted Domains. *CoRR*. 2021, vol. abs/2108.01987. Available from arXiv: `2108.01987`.

36. BREDERECK, Robert; FALISZEWSKI, Piotr; KACZMARCZYK, Andrzej; KNOP, Dušan; NIEDERMEIER, Rolf. Parameterized Algorithms for Finding a Collective Set of Items. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020, vol. 34, no. 02, pp. 1838–1845. Available from DOI: `10.1609/aaai.v34i02.5551`.

37. SKOWRON, Piotr; FALISZEWSKI, Piotr; LANG, Jérôme. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artificial Intelligence*. 2016, vol. 241, pp. 191–216. ISSN 0004-3702. Available from DOI: `https://doi.org/10.1016/j.artint.2016.09.003`.

38. ELKIND, Edith; ISMAILI, Anisse. OWA-Based Extensions of the Chamberlin–Courant Rule. In: WALSH, Toby (ed.). *Algorithmic Decision Theory*. Cham: Springer International Publishing, 2015, pp. 486–502. ISBN 978-3-319-23114-3.

39. AMANATIDIS, Georgios; BARROT, Nathanaël; LANG, Jérôme; MARKAKIS, Evangelos K.; RIES, Bernard. Multiple Referenda and Multiwinner Elections Using Hamming Distances: Complexity and Manipulability. In: *Adaptive Agents and Multi-Agent Systems*. 2015.