1 Introduction

People hold different opinions and preferences over almost everything. Yet in many situations a common decision must be made. For example, sometimes people need to select a leader, or decide whether or not to provide a public good such as national defense. The best-known way to achieve these goals is by *voting*, which has been a critical component of democracy since ancient time. As early as around 350 B.C., Plato (424/423 B.C.–348/347 B.C.), in spite of being famous for his objection against democracy, proposed several multi-stage voting processes to elect the "guardians" of the law and officeholders, etc., in his unfinished book "*The Law*". Obviously Plato was not the first person who thought about voting. In fact, Socrates (469 B.C.–399 B.C.), Plato's teacher, was sentenced to death by a majority voting. Plato thus had good reasons to object to democracy. After Plato, the first well-known voting system that is not based on majority voting was proposed by Ramon Llull (1232–1315). Then, the systematic study of the theory of voting prospered with the French Revolution in the 18th century. During that time, two of the most famous philosophers who made significant contributions to the theory of voting are Marie Jean Antoine Nicolas de Caritat, marquis de Condorcet (1743–1794, also known as

Nicolas de Condorcet, who proposed the Condorcet criterion), and Jean-Charles, chevalier de Borda (1733–1799, who proposed the Borda voting rule). More recently, Kenneth Arrow (a co-recipient of the *Nobel Memorial Prize in Economics* in 1972) showed that it is impossible to design a voting rule that satisfies some very natural properties (Arrow, 1950). This seminal work is thus named *Arrow's impossibility theorem*, and is broadly regarded as the beginning of modern *Social Choice Theory*, which is an active research direction in Economics.

In recent years, rapid developments in computers and networks have brought big changes to human society. Computers not only have helped us solve problems faster, but also have brought revolutions to the ideology of the human society. For example, the ultimate goal of *Artificial Intelligence (AI)* is to build computers that are as "intelligent" as, if not more intelligent than, human beings. These changes have led to many new interdisciplinary areas. In particular, the interdisciplinary area lying in the intersection of Computer Science and Economics has attracted huge attention, partly due to the emerging electronic commerce of the Internet era. One place where Computer Science meets Economics is the new subarea of AI called *Multi-Agent Systems*, which studies interactions and collaborations in systems that consist of multiple intelligent agents (Wooldridge, 2009). Similar as for human beings, voting could help intelligent agents to make a joint decision in many situations. For example, in the system developed by Ephrati and Rosenschein (1991), agents use voting to decide the next step in their joint plan. There are also many applications of voting in electronic commerce, for example, Ghosh et al. (1999) proposed to use voting to help build recommendation systems; Pennock et al. (2000) adopted the core method in traditional Voting Theory—the axiomatic approach—to analyze collaborative filtering algorithms in recommendation systems; and Dwork et al. (2001) proposed to treat web-search engines as agents, and use voting to decide the best matching website.

In many new applications of voting, we encounter an extremely large number of alternatives or an overwhelming amount of information, which leads to significant computational challenges. To handle these situations, we need to design faster algorithms or build faster computers. On the other hand, higher computational capability makes it easier for voters to figure out beneficial strategic behavior, which might lead to undesirable outcomes. In order to reap the benefits of these potential applications and overcome the emerging problems, we need to develop new algorithms and methodologies. A burgeoning area—*Computational Social Choice*—aims to address problems in computational aspects of information/preference representation and aggregation in multi-agent scenarios (Chevaleyre et al., 2007).

A first question that should be asked is: why it is voting that people or intelligent agents should want to use to aggregate their preferences? Certainly in some situations people use other mechanisms. For example, sometimes auctions are used to determine an allocation of resources or tasks. A key feature in the situations where people or agents use voting is that they only have, or are limited to express, ordinal preferences, in contrast to cardinal preferences measured by real numbers that represent utilities and allow for monetary transfers. In this dissertation, I put aside the discussion of many important topics, including the comparison between voting and other mechanisms, cardinal vs. ordinal preferences, rationale behind the utility theory, etc. An interested reader may refer to Conitzer (2010) for discussions on such topics. Instead, I will focus on the situations where voting is used. It should be kept in mind that voting is a good option for preference/information aggregation in many, but not all situations. My research seeks to investigate and foster the interplay between Computer Science and Voting Theory. In particular, my research focuses the conceptual and methodological aspects of the interplay: (1) **how computational thinking** (Wing, 2006) **changes the traditional voting theory conceptually**, and (2) **methodologically how can we better use voting for**

preference/information aggregation with the help of Computer Science.

1.1 Structure of This Dissertation

The structure of my dissertation is illustrated in Figure 1.1. Most of my research focuses on Computational Voting Theory, which is the most active branch of Computational Social Choice (Node 1 in Figure 1.1). To make the dissertation coherent and to keep it at a reasonable length, I will discuss two research directions that belongs to the two high-level aspects mentioned in the end of the last section. The first direction focuses on investigating how computational thinking affects the game-theoretic aspects of voting (Node 2 in Figure 1.1). The second direction studies the design and analysis of novel voting rules when the set of alternatives is exponentially large and has a combinatorial structure, with the help of some recent developments in Artificial Intelligence (Node 3 in Figure 1.1). These two research directions converge to the study of the game-theoretic aspects of combinatorial voting (Node 4 in Figure 1.1).

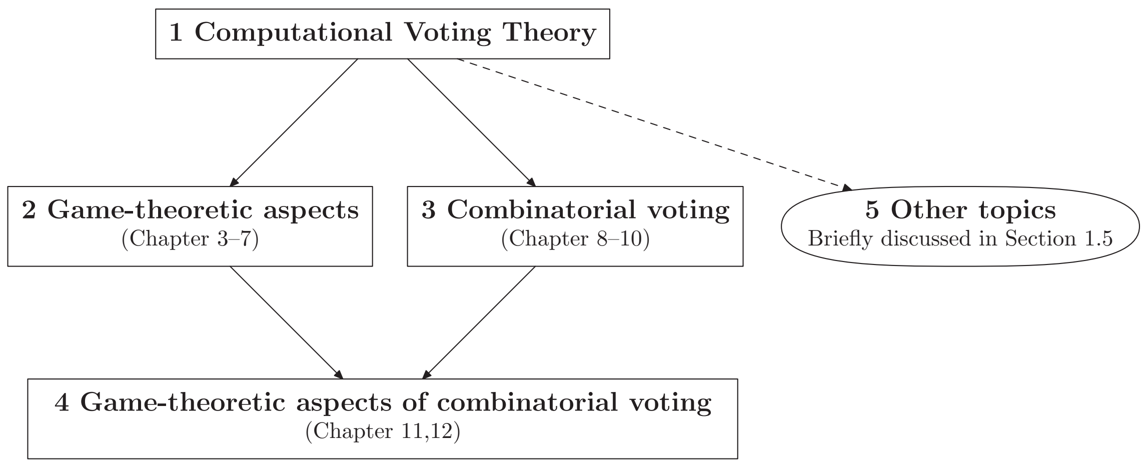
1 Computational Voting Theory
2 Game-theoretic aspects
(Chapter 3–7)
3 Combinatorial voting
(Chapter 8–10)
5 Other topics
Briefly discussed in Section 1.5
4 Game-theoretic aspects of combinatorial voting
(Chapter 11,12)

FIGURE 1.1: Structure of my dissertation.

In the remainder of this chapter, I will slightly expand on the nodes in Figure 1.1.

1.2 Computational Voting Theory

Computational Voting Theory, which studies computational issues in voting, is the most active branch of Computational Social choice. Throughout the dissertation, a vote is a linear order[1] over the set of alternatives (candidates), we ask each voter (agents) to cast one vote. These votes constitute a *profile*. Then, we apply a voting rule to the profile to determine the winning alternative (the *winner*).

Example 1.2.1. *Suppose three candidates {Clinton, Obama, McCain} are competing for a presidential position. We use the* plurality *rule to select the winner. That is, the candidate who is ranked at the top the most time in the votes wins, and suppose ties are broken alphabetically. Suppose there are five voters whose votes are as follows:*

$$\begin{array}{ll} \textit{Voter 1 :} & \textit{Clinton} \succ \textit{Obama} \succ \textit{McCain} \\ \textit{Voter 2,3:} & \textit{Obama} \succ \textit{McCain} \succ \textit{Clinton} \\ \textit{Voter 4,5:} & \textit{McCain} \succ \textit{Clinton} \succ \textit{Obama} \end{array}$$

Then, the winner is McCain, because he is ranked in the top position for two times (tied with Obama), and the tie is broken in favor of McCain.

The formal definition of voting systems and some popular voting rules can be found in Chapter 2. In Computational Voting Theory, researchers have extensively investigated at least the following questions.

- How can we compute the winner or ranking more efficiently?
- How can we communicate and elicit voters' preferences more efficiently?
- How can we use computational complexity to protect elections from bribery and control?

[1] However, see Pini et al. (2007), for a discussion of voting where preferences over the candidates are represented by a partial order.

- How can we prevent voters from misreporting their preferences?
- How can we analyze voters' incentive and strategic behavior?
- How can we design novel voting rules when the set of alternatives has a combinatorial structure, and is exponentially large?

Nodes 2–4 correspond to the last three questions. More detailed discussions as well as references can be found in Chapter 2.

1.3 Node 2: Game-theoretic Aspects

An important yet always implicit assumption when most popular voting rules were designed is that all voters report their preferences truthfully. However, in many real world voting systems, a voter may well lie to make herself better off. This phenomenon is call a *manipulation*. For example, let us recall Example 1.2.1, and suppose that the votes described in the example are the voters' true preferences. We have already seen that if all five voters report truthfully, then McCain is the winner. However, if the first voter reports that her vote is Obama>Clinton>McCain, while the other voters all report truthfully, then Obama is the winner. Note that the first voter prefers Obama to MaCain, which means that she has an incentive to misreport her preferences to make herself better off. This kind of strategic behavior makes the outcome of the voting process unpredictable, and can sometimes hurt the voters, including the manipulators themselves, when there is more than one manipulator. Therefore, it is important to investigate the strategic behavior of the voters. This falls under *Game Theory* (Fudenberg and Tirole, 1991). First of all, it would be great if we can use a voting rule for which there is never any opportunity for manipulation, i.e., a *strategy-proof* voting rule. This objective might seem to be too ambitious at first glance, but in fact, there are many strategy-proof mechanisms in other settings where voters are allowed to express their cardinal preferences, their preferences are quasi-linear, and monetary transfers are allowed. For example, the well-known VCG mechanisms are strategy-proof (Vickrey, 1961; Clarke, 1971; Groves, 1973). Unfortunately, in voting settings where no monetary transfers are allowed, due to the celebrated Gibbard-Satterthwaite theorem (Gibbard, 1973; Satterthwaite, 1975), when there are three or more alternatives, no strategy-proof voting rule satisfies the following two desired properties: (1) non-imposition (i.e., each alternative wins for some profile) and (2) non-dictatorship (i.e., there is no dictator, a voter whose first-ranked alternative is always the winner). To circumvent this very negative result, economists have proposed to restrict the domain of preferences to obtain strategy-proofness. That is, we assume that voters' preferences always lie in a restricted set of linear orders. One example of such a class is the set of *single-peaked* preferences (Black, 1948). For single-peaked preferences, desirable strategy-proof rules exist, such as the *median* rule (Moulin, 1980). More details can be found in Chapter 12, where I will discuss our own results along this line as well.

Besides this, my research on the game-theoretic aspects of Voting Theory diverges into two directions, illustrated in Figure 1.1. The first direction (the left branch) focuses on exploring the idea of using computational complexity to prevent manipulation. The second direction (the right branch) focuses on analyzing the equilibrium outcome in a type of voting games.

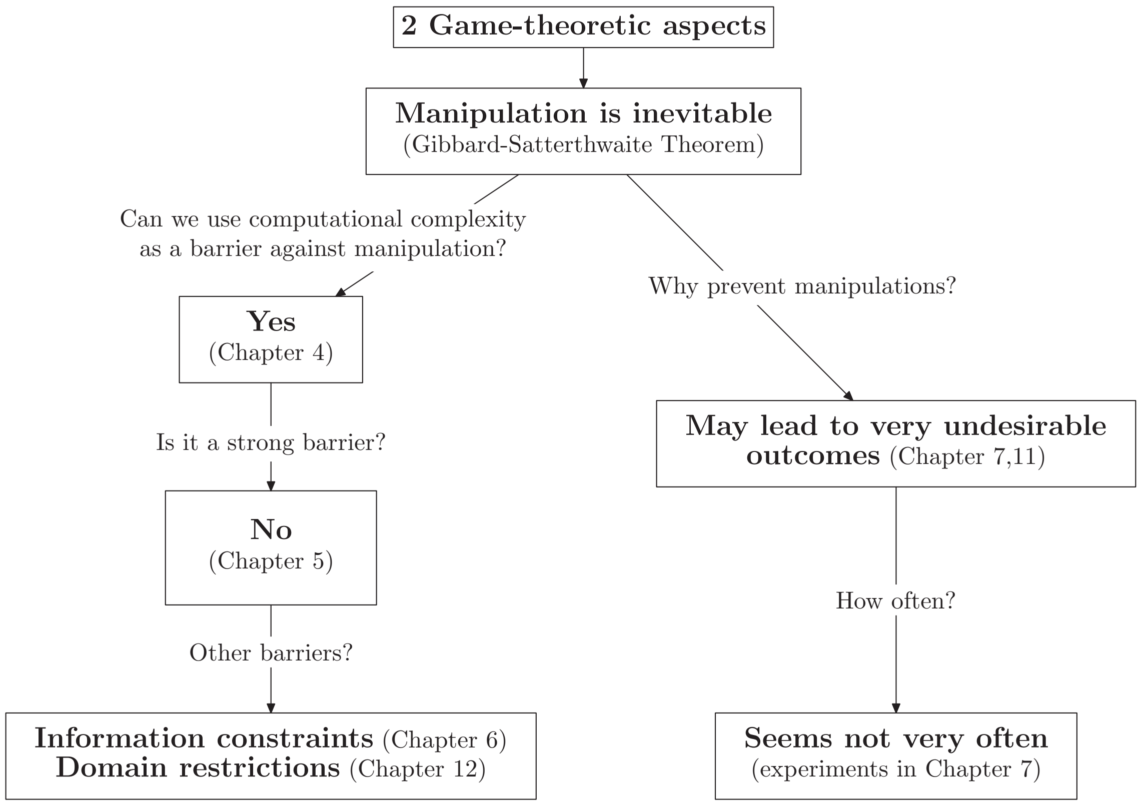
2 Game-theoretic aspects
Manipulation is inevitable
(Gibbard-Satterthwaite Theorem)
Can we use computational complexity as a barrier against manipulation?
Why prevent manipulations?
Yes
(Chapter 4)
Is it a strong barrier?
No
(Chapter 5)
Other barriers?
Information constraints (Chapter 6)
Domain restrictions (Chapter 12)
May lead to very undesirable outcomes (Chapter 7,11)
How often?
Seems not very often
(experiments in Chapter 7)

FIGURE 1.2: Two directions in game-theoretic aspects of voting.

1.3.1 First Direction: Computational Complexity of Manipulation

Even though a manipulation is guaranteed to exist, if we can prove that finding a manipulation is computationally hard for some common voting rules, then a manipulation might not occur simply because the manipulator(s) cannot find it in a reasonable amount of time, or it is computationally too costly to do so. This idea was first explored by Bartholdi et al. (1989a), which, together with Bartholdi et al. (1989b, 1992), have been broadly considered the starting point of Computational Social Choice. After that, a number of results have been obtained on the computational complexity of manipulation in various settings. See Faliszewski et al. (2010b); Faliszewski and Procaccia (2010) for recent surveys. More details will also be given in Chapter 4.

Chapter 4 focuses on the most natural setting where voters are equally weighted, and there are multiple manipulators who want to cast their votes collaboratively to make a favored alternative win. I will show that for some common voting rules, finding a manipulation is NP-hard, while for some other voting rules, there exist polynomial-time algorithms to find a manipulation. Therefore, at least for some common voting rules, the answer to the question "Can we use computational complexity as a barrier against manipulation?" is "Yes". This answer is quite positive, because it implies that at least for these voting rules, even if the potential manipulators use the fastest computer in the world, they are unlikely to find an algorithm that can always tell them the answer quickly even for large instances (assuming P$\neq$ NP). Consequently, these potential manipulators might have less incentive to misreport their preferences.

Proving the NP-hardness of finding a manipulation is only a first step. Even though it is NP-hard to find a manipulation, the manipulators may still not always report their true preferences. For example, they can certainly run a heuristic algorithm for a certain amount of time, say one minute, and if the algorithm returns a successful manipulation, then they will cast the votes returned by the algorithm; otherwise, if the algorithm fails to compute an answer in one minute, they may then report their true preferences. Technically, this problem is due to the fact that NP-hardness is a *worst-case* concept. Therefore, it is natural to ask, informally, whether manipulations are computationally hard to find in "most" cases. Some previous work gave partial answers to this question. Again, more details and discussion can be found in Faliszewski et al. (2010b); Faliszewski and Procaccia (2010) and/or Chapter 4. We will see in Chapter 5 that, for a very general class of voting rules called *generalized scoring rules*, which include many common voting rules, the cases where manipulations are hard to find are exceptions rather than the rule. Therefore, computational complexity does not seem to be a very strong barrier against strategic behavior, so that we need to seek other barriers. For example, we may try to limit the manipulators' information about the preferences of the other voters (Chapter 6), or only allow the voters to pick a vote from a restricted set of linear orders (Chapter 12).[2]

1.3.2 Second Direction: Equilibrium Outcomes in Voting Games

In fact, the very first question that should be asked is, is it ever desirable to prevent the voters' strategic behavior? After all, the ultimate objective of voting is to select a "good" alternative. So if somehow the strategic behavior of the voters leads to the same, or an even better, outcome, then there is no reason to even try to prevent the voters from being strategic. Moreover, in such cases, maybe the strategic behavior should actually be encouraged! Surprisingly, this question was not answered before. To analyze the outcome when voters are strategic, the most natural way is to use Game Theory to model the voting process as a game, and then focus on the winner in the outcome of the game in terms of some solution concept, e.g., *Nash equilibrium*.[3] However, in general a voting game has too many (Nash) equilibria. This makes it very hard to draw any useful conclusions on the impact of strategic behavior on the outcome of voting.

In Chapter 7, we study a type of voting games where voters cast their votes one after another sequentially. We call such games *Stackelberg voting games*. We will focus on a finer solution concept called *subgame-perfect Nash equilibrium*. Fortunately, in any Stackelberg voting game, the outcome is unique in all subgame-perfect Nash equilibria. One might expect that the strategic behavior would sometimes harm the voters, but there are two main difficulties in drawing such a conclusion, which come

[2] As mentioned earlier, this idea has been approached mainly by economists. I will further explore it in the setting of combinatorial voting.

[3] In general simultaneous-move voting games, a Nash equilibrium is a profile where no voter can benefit from casting a different vote. The formal definition of voting games, Nash equilibrium, and its refinement *subgame-perfect Nash equilibrium* can be found in Chapter 7.

from the following two natural questions.

1. *To what extent* can the strategic behavior harm the voters? The main difficulty here is that voting aims at aggregating voters' *ordinal* preferences, which means that generally it is nontrivial to measure how good/bad an alternative is.[4]

2. How *often* does the strategic behavior harm the voters?

Chapter 7 answers the above two questions. The first question is answered by showing some paradoxes, which state that sometimes the (unique) equilibrium winner is ranked in extremely low positions in almost all voters' true preferences. Without doubt this is an extremely undesirable outcome. Therefore, these paradoxes illustrate the cost of strategic behavior of the voters, and suggest that at least in some cases, strategic behavior should be prevented. The second question is partly answered by simulations. Surprisingly, for most common voting rules, the winner in the equilibrium outcome is slightly "better" for the voters on average, compared to the winner when they vote truthfully.

1.4 Node 3: Combinatorial Voting

So far we have been discussing voting over unstructured sets of alternatives. In many real-life situations, there are multiple *issues* (*attributes*, or *characteristics*), and each alternative can be uniquely characterized by a vector of the values these issues take. Such settings are called *combinatorial voting* (or *voting in combinatorial domain*). For instance, when agents vote to select a president and a treasurer, each position corresponds to an issue whose value corresponds to the person selected to hold the

[4] This is in sharp contrast to the settings where there is a well-defined social welfare function, especially in the settings where the agents have quasilinear utility functions, and are allowed to express their cardinal preferences, for example in auctions. In those situations, the cost of strategic behavior can be measured by the *price of anarchy* (Koutsoupias and Papadimitriou, 1999), that is, the ratio of the optimal social welfare over the worst social welfare in equilibrium outcomes.

position. In combinatorial voting, selecting a winner amounts to making a public choice for each of the issues. The main difficulty resides in the exponentially large number of alternatives. Therefore, it is computationally impractical to directly apply a common voting rule designed for unstructured sets of alternatives in the setting of combinatorial voting.[5] For combinatorial voting, we need to design new voting rules that are computationally tractable.

In the literature, researchers in Economics and Political Science have extensively studied voting processes where the agents vote over issues *separately in parallel*. This method works well when agents' preferences over one issue do not depend on any other issues. However, in general agents' preferences over one issue may depend on the value of other issues. For example, if a Democrat is selected to be the president, then a voter may prefer selecting a Republican to be the treasurer; but if a Republican is selected to be the president, then the voter may prefer selecting a Democrat to be the treasurer. There are two main challenges for combinatorial voting: Language-wise we need a more natural way for the agents to truthfully report their preferences. Methodology-wise we also need a more general theory of computational tractable combinatorial voting.

My research in combinatorial voting can be roughly categorized into two directions, illustrated in Figure 1.3. The first direction focuses on designing computationally tractable voting rules for combinatorial voting. The second direction (Node 4 in Figure 1.1) focuses on game-theoretic aspects of combinatorial voting, where we aim at analyzing and preventing voters' strategic behavior in combinatorial voting.

[5] Some voting rules that only use a very small portion of the voters preferences to select the winner, for example the plurality rule, do not have significant computational issues when they are used in combinatorial voting. However, in general these rules will not select a "good" outcome in combinatorial voting. More discussions will be given in Chapter 8.

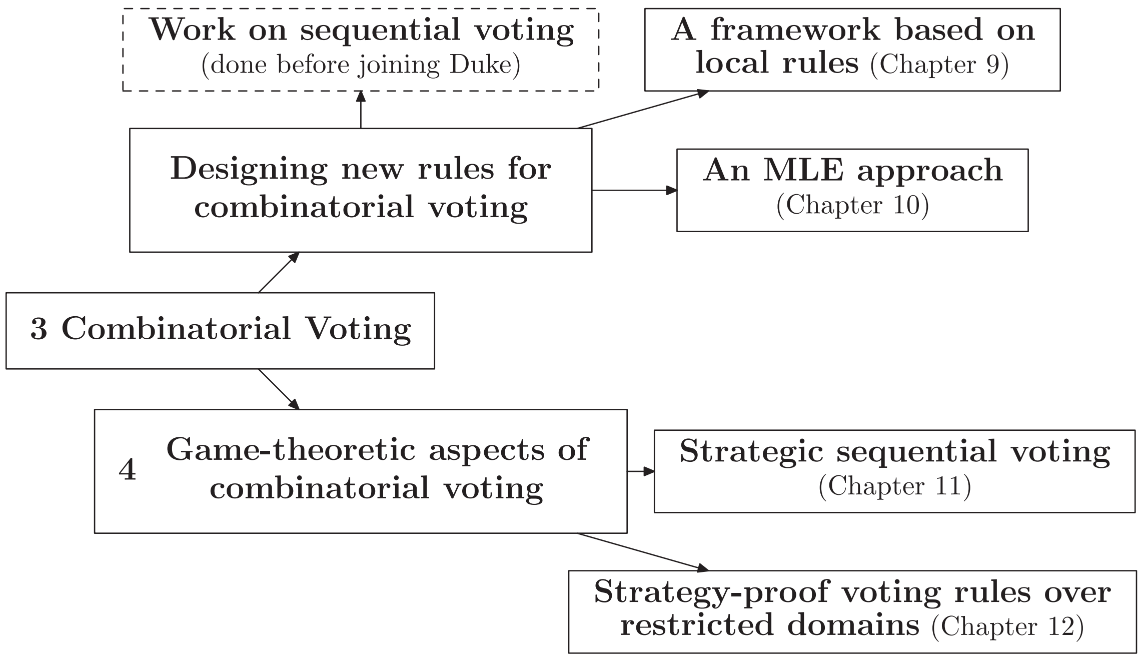
Work on sequential voting
(done before joining Duke)
A framework based on local rules (Chapter 9)
Designing new rules for combinatorial voting
An MLE approach (Chapter 10)
3 Combinatorial Voting
4 Game-theoretic aspects of combinatorial voting
Strategic sequential voting (Chapter 11)
Strategy-proof voting rules over restricted domains (Chapter 12)

FIGURE 1.3: Two directions in combinatorial voting.

1.4.1 Designing New Rules for Combinatorial Voting

One attempt to design computationally tractable voting rules consists of *sequential voting rules*, where agents vote *sequentially*, in the sense that they vote to make the choice for the first issue by a "local" voting rule, then move on to the second issue and vote to make the choice by another local voting rule, etc., given an order over the issues (Lang, 2007). Much of my work was built on the idea of sequential voting, which allows the agents' preferences over one issue to depend on some (but not necessarily all) other issues. Formally, the voters are advised to use a compact voting language called *CP-nets* (Boutilier et al., 2004), which was recently proposed in the Artificial Intelligence community as a preferential counterpart of *Bayesian networks* (Pearl, 1988).

However, in order for sequential voting to work well, there are two levels of technical constraints. First, each voter's preferences must be represented by an *acyclic* CP-net. In other words, for each voter, there exists at least one linear ordering over the issues, such that the voter's preferences over later issues in this order only depend on the values of all previous issues. That is, the voter's preferences are *compatible* with that ordering over issues. The second is that all voters' preferences must be compatible with a common (linear) ordering over the issues. For example, consider a combinatorial voting setting where there are two issues: an issue for "main dish", which can be either fish or beef, and another issue for "wine", which can be either red wine or white wine. The two constraints state that there exists an ordering over the two issues, w.l.o.g. main dish>wine, with which all voters' preferences are compatible. That is, each voter' preferences over wine depends on the value of main dish. From a high-level point of view, these two constraints imply that sequential voting rules have high computational efficiency, but the voting language (i.e., acyclic CP-nets that are compatible with a common ordering over issues) is too restrictive. On the other hand, common voting rules designed for unstructured sets of alternatives have low computational efficiency in the setting of combinatorial voting, but the voters have more flexibility in expressing their preferences.

Designing a good voting rule with high computational efficiency and a fully expressive language seems to be a mission impossible. Therefore, my work in combinatorial voting aims to design voting rules that tradeoff computational efficiency and expressiveness of the voting language. We will start designing such voting rules by assuming all voters vote truthfully (Chapter 9, 10). Complications caused by the strategic behavior of the voters will be examined later (Chapter 11, 12). In Chapter 9, we will see a framework that first considers a directed graph over all alternatives by applying local voting rules, then uses a *choice set function* to select the winner from this graph. This framework allows a voter to use any CP-net (even an acyclic one) to represent her preferences. We will also see that whether or not the voting rule defined by this framework satisfies some desired properties for voting rules, e.g., *anonymity, neutrality, etc.*, depends on both the choice set function and whether the local voting rules satisfy these properties. In general, computing the winners in this framework is hard. However, we will see an algorithm that could save significant amounts of time when the (possibly cyclic) CP-nets that represent voters' preferences share some common structure.

Chapter 10 takes a different approach towards defining new rules for combinatorial voting. Suppose there is a "correct" winner and the voters' preferences are noisy perceptions of it. If we have a probabilistic model that generates voters' preferences given the "correct" winner, and a probability distribution for an alternative to be the "correct" winner, then having seen the voters vote, we can compute the posterior probability for each alternative to be the "correct" winner via standard Bayesian reasoning. In other words, the voting rule defined by this process can be viewed as the *maximum likelihood estimator (MLE)* of the probabilistic model. This idea was actually introduced two hundred years ago by Condorcet (1785) to design a voting rule for unstructured sets of alternatives. The main question is, how should we define the probabilistic model? In Chapter 10, we will see a natural probabilistic model for sets of alternatives composed of binary issues, called *distance-based noise models*, where the conditional probability given the "correct" winner is decomposed into local distributions, one for each issue i. More precisely, the local distribution over any issue i under some setting of the other issues depends only on the Hamming distance from this setting to the restriction of the "correct" winner to the issues other than i. Some results on the computational complexity of winner computation will be presented, followed by discussions about the relation between the MLE approach and sequential voting rules.

1.5 Node 4: Game-Theoretical Aspects of Combinatorial Voting

The formulation of a voting game largely depends on the voting rule used in the voting process. As I argued in the last section, in combinatorial voting it is generally computationally costly to use common voting rules designed for unstructured sets of alternatives. Therefore, the arguments and results in Section 1.3, which were made for common voting rules designed for unstructured sets of alternatives, do not directly apply to combinatorial voting. Since sequential voting is one of the most natural approaches in combinatorial voting, this suggests to study a voting game where voters cast votes strategically on one issue after another, following some ordering over the issues. Indeed, strategic voting is arguably more likely in such a sequential game than in "one shot" voting. We call this type of voting games *strategic sequential voting*, which is the main topic of Chapter 11.[6] Compared to (truthful) sequential voting mentioned in the previous subsection, for strategic sequential voting the focus is on different aspects. In truthful sequential voting, a major concern is how expressive the voting language is. In strategic sequential voting, however, the expressivity of the voting language is not the most important issue. Instead, what really matters is how a strategic voter's preferences and knowledge about the other voters' preferences determine her behavior in the voting game, and thus influence the outcome of the game. Therefore, in the game-theoretic part on combinatorial voting, we are interested in the following two questions. The first question is exactly the same as question 1 asked in Section 1.3.2, but here it is asked for strategic sequential voting.

1. To what extent can the strategic behavior harm the voters in strategic sequen-

[6] We note that strategic sequential voting is different from the Stackelberg voting games mentioned in Section 1.3.2. In Stackelberg voting games voters cast their votes one after another, while in strategic sequential voting, voters cast votes simultaneously on individual issues, one issue after another.

tial voting?

2. If the strategic behavior of the voters can harm the voters badly, how can we prevent it?

The first question is answered by three types of *multiple-election paradoxes*: there exists a profile for which (1) the winner under strategic sequential voting is ranked nearly at the bottom in *all* voters' true preferences, (2) the winner is *Pareto-dominated* by *almost every* other alternative, and as a consequence, (3) the winner is an almost *Condorcet loser*.[7] Even worse, changing the ordering over the issues on which the voters vote cannot completely prevent these paradoxes. Hence, the outcome of strategic sequential voting can be extremely undesirable to all voters. Similar paradoxes have been shown for other models of behavior in combinatorial voting in the literature (Scarsini, 1998; Brams et al., 1998), but as far as we know, we were the first to discover these paradoxes in a strategic environment, to illustrate the cost of the strategic behavior of the voters. See Chapter 11 for more references and discussion.

One approach to addressing the concern raised by the second question is restricting the voters' preferences. We will see in Chapter 11 that by restricting the voters' preferences to be *separable* or *lexicographic*, all three types of multiple-election paradoxes mentioned earlier disappear. In fact, by putting more constraints on the voters' preferences, we can obtain strategy-proof sequential voting rules for combinatorial voting. We can further show that if the domain restriction satisfies some mild conditions, then a voting rule is strategy-proof if and only if it is a sequential voting rule, where each local rule is strategy-proof over its respective local domain. This will be discussed in Chapter 12.

[7] The definitions for Pareto-domination and Condorcet loser will be found in Chapter 11.

1.6 Node 5: Work Excluded from My Dissertation

During my Ph.D. studies, I also have worked on some other important topics in preference/information representation and aggregation. These works will not be discussed in detail in the dissertation due to considerations of length and coherence of the dissertation. In this section, I will briefly describe these works, illustrated in Figure 1.4. An interested reader may also refer to Xia (2010).

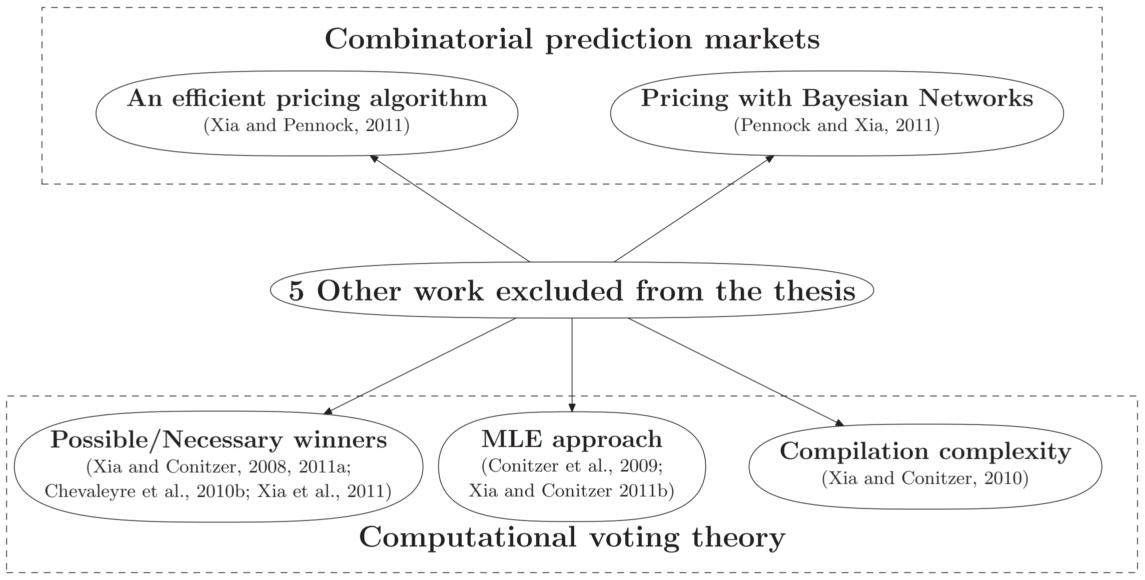
Combinatorial prediction markets
An efficient pricing algorithm
(Xia and Pennock, 2011)
Pricing with Bayesian Networks
(Pennock and Xia, 2011)
5 Other work excluded from the thesis
Possible/Necessary winners
(Xia and Conitzer, 2008, 2011a; Chevaleyre et al., 2010b; Xia et al., 2011)
MLE approach
(Conitzer et al., 2009; Xia and Conitzer 2011b)
Compilation complexity
(Xia and Conitzer, 2010)
Computational voting theory

FIGURE 1.4: Topics excluded from my dissertation.

1.6.1 My Other Work in Computational Voting Theory

In addition to the topics discussed in Section 1.3, 1.4, and 1.5, I have also worked on the following three topics.

- **Computing possible/necessary winners.** In practice, we may not need to know the voters' full preferences to compute the winner. That is, information elicited at an early stage might suffice to conclude who the winner is. For this purpose, it is important to know the answers to the following two computational questions when only part of the voters' preferences are elicited: (1) Is it still possible for a given alternative to win? (2) Has the winner already been determined, so that we may terminate the elicitation process and announce the winner? These two problems are known as the *possible/necessary winner problems*, respectively (Conitzer and Sandholm, 2002; Konczak and Lang, 2005). I investigated the computational complexity of these possible/necessary winner problems for many common voting rules (Xia and Conitzer, 2008a, 2011a), as well as in the special case where the alternatives do not arrive at the same time (Chevaleyre et al., 2010b,a; Xia et al., 2011).

- **Compilation complexity.** One closely related topic to possible/neccessary winner determination is the *compilation complexity* of common voting rules. Here the agents do not arrive at the same time, and we are asked in the middle of the election, what is the lowest number of bits required to store enough information about the votes cast so far to determine the winer (Chevaleyre et al., 2009). In recent work (Xia and Conitzer, 2010a), we proved asymptotically matching upper and lower bounds on the compilation complexity for many common voting rules. We also devised polynomial-time algorithms to "compress" and store the votes in the middle of an election. These algorithms can significantly speedup the algorithm used to compute the subgame-perfect Nash equilibrium in Stackelberg voting games (Chapter 7).

- **A maximum-likelihood estimator approach towards general voting.** As I discussed in Section 1.4.1, one principled way to design a reasonable voting rule is by setting up a probabilistic model, and then define the voting rule to be the maximum-likelihood estimator of this model. Of course this idea is not limited to multi-issue domains, as the idea of using it for unstructured sets of alternatives dates back to Condorcet (1785). In recent work (Conitzer et al., 2009b), we showed that the MLE approach gives us a group of aggregation functions called *ranking scoring rules*, which are used to output an aggregated linear order over all alternatives. The MLE approach can also be used to systematically extend common voting rules that aggregate linear orders to aggregate partial orders (Xia and Conitzer, 2011b).

In addition to the above three topics, I also did some work on sequential voting in combinatorial domains before starting my Ph.D. studies at Duke. This work will be mentioned in Chapter 8 as a part of the literature in combinatorial voting.

1.6.2 Combinatorial Prediction Markets

Prediction markets are financial markets that aggregate agents' probabilistic beliefs about the outcome of a random event. The Iowa Electronic Markets and Intrade are two examples of real prediction markets with a long history of tested results. See Chen and Pennock (2010) for a recent survey of prediction mechanisms. Unfortunately, if the space has a combinatorial structure, then the central problem of computing the prices for securities is #P-hard (Chen et al., 2008a). For example, in the NCAA mens basketball tournament, there are 64 teams and therefore 63 matches in total to predict, where each match can be seen as a binary variable. Such settings are known as *combinatorial prediction markets*.

Recently, I revealed two natural relationships: the first (Xia and Pennock, 2011) bridges combinatorial prediction markets and the *weighted model counting* problem, a central problem in AI; and the second (Pennock and Xia, 2011) bridges combinatorial prediction markets and *probabilistic belief aggregation*, a well-studied problem in both Statistics and AI. Inspired by the first relationship, I designed an efficient novel Monte Carlo sampling technique based on importance sampling that has a good theoretical guarantee, for combinatorial prediction markets for tournaments (Xia and Pennock, 2011). The second relationship helped us further explore the idea of using a compact representation scheme (formally, a Bayesian network) to represent the prices of securities (Chen et al., 2008b), and completely characterize all structure-preserving securities (meaning that these securities can be computationally efficiently priced) (Pennock and Xia, 2011).

1.7 Summary

In this chapter, I categorized some of my Ph.D. work in Computational Voting Theory into two lines of research directions: the game-theoretic aspects and combinatorial voting. I briefly discussed the motivating questions in both lines of research and their intersection, and the results that will be presented in later chapters. To make the dissertation coherent and to keep it at a reasonable length, some of my work that are not included in this dissertation. Some of them were briefly discussed in Section 1.6.