ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Computational Considerations of Voting Rules
(Παιγνιοθεωρητική Ανάλυση συστημάτων Ψηφοφορίας)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος, Π. Αγγελής

Επιβλέπων : Ευστάθιος Ζάχος
Καθηγητής

Αθήνα, Μάρτιος 2010

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Computational Considerations of Voting Rules

(Παιγνιοθεωρητική Ανάλυση συστημάτων Ψηφοφορίας)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Π. Αγγελής

Επιβλέπων : Ευστάθιος Ζάχος
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Μαρτίου 2010.

....	....	....
Ευστάθιος Ζάχος	Δημήτρης Φωτάκης	Άρης Παγουρτζής
Καθηγητής	Λεκτορας	Επίκουρος Καθηγητής

Αθήνα, Μάρτιος 2010

Abstract

In this thesis we are examining various results in the field of Voting seen as a subfield of mechanism design. The fact that Voting lies on the interface of computer science, operations research and political science seems of particular interest to us. In the introduction a general description of the field of mechanism design and game theory is given. Next is considered the traditional track of study, i.e. the direction of research before computational considerations were introduced. We stress out the importance of the Gibbart-Satterthwaite theorem on the field of Social Choice also in the sense that it gave rise to a new direction of research: studying Social Choice with computational terms. The main focus of the thesis is put on this direction of research. Tasks such as winner determination, manipulation, bribery and control of different voting rules are examined from this point of view. We present results concerning algorithmic issues, worst-case computational complexity analysis and see what steps have been taken towards the average-case complexity analysis, which was really the researchers' desired measure of analysis from the beginning. The focus is put heavily on the problem of manipulation.

We hope that this thesis will serve as a starting point for other people who want to study the field.

Acknowledgements

I would like to deeply thank my three teachers and members of the committee Prof. S. Zachos, Prof. D. Fotakis and Prof. A. Pagourtzis. I feel very lucky I had the chance to be taught by them; I feel they have influenced me in a very positive way not only academically but also on a personal level. Particularly I want to thank the two supervisors of this thesis, Prof. S. Zachos and Prof. D. Fotakis for their guidance and their persistence in helping me make this thesis better. I express my thanks to all of the members of the Computation and Reasoning Laboratory (CoreLab) for their help, for their constructive observations during the preliminary presentations, and for putting the bar high: Andreas, Andreas, Eleni, Georgia, Manos, Paris, Thanasis, Vangelis, Vasilis. Finally I want to thank my friends and family for their support; particularly Haris for his gruelling proof-reading of part of the text.

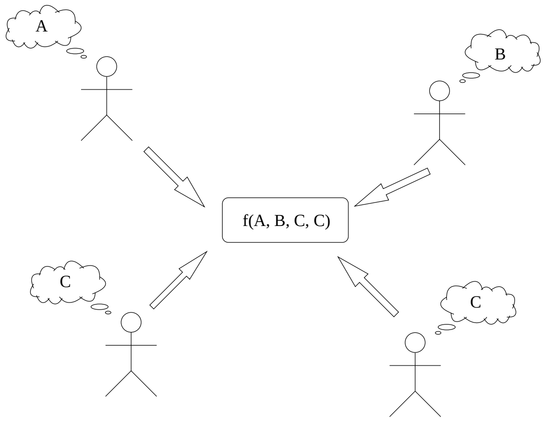
Georgios Angelis, Athens, March 2010

Contents

Chapter 1

Introduction

Mechanism Design is traditionally considered a subfield of economic theory but as we will see it can have applications in a number of scenarios. It is rather unique within economics in having an engineering perspective. It is interested in designing economic mechanisms, just like computer scientists are interested in designing algorithms, protocols, or systems. It is best to view the goals of the designed mechanisms in the very abstract terms of *social choice.* A social choice is simply an aggregation of the preferences of the different participants towards a single joint decision, which is optimal in some sense. In other words we can - and it helps to do so - view each mechanism design problem as a distributed optimization problem, where the constrains are the preferences of the participants and are distributed in the sense that they are *private.*



Social Choice aggregates various preferences

Central to Mechanism Design is a *game-theoretic approach.* By this we mean that mechanism design attempts *implementing* desired social choices in a strategic setting assuming that the different members of society each act *rationally* in a game theoretic sense, meaning that they are self-interested. Such strategic design is necessary since, as we said, usually the preferences of the participants are private.

In the sections to come, we will define formally what we mean by *implementing* and give thorough explanations about the game-theoretic notions implicated in Mechanism design. For now we would just like to stress out the fact that the goal of mechanism design is captured through the notion of *social choice* and that we use *game theory* to analyse the interaction of the various participants in the mechanism. We can express the above mentioned with an equation:

$$\textit{Mechanism Design} = \textit{Social Choice} + \textit{Game Theory}$$

The notion of *social choice* that captures the essence of mechanism design is quite general and it can in fact be seen as a generalisation of a multitude of scenarios in economics, political science, or of course technical settings, such as networks. Here are some classic examples:

Elections: In political elections each voter has his own preferences between the different candidates, and the outcome of the elections is a single social choice. The voting system used is the mechanism that gives the outcome of the election.

Markets: Classical economic theory usually assumes the existence and functioning of a "perfect market." In reality, of course, we have only interactions between people, governed by some protocols. Each participant in such an interaction has his own preferences, but the outcome is a single social choice: the reallocation of goods and money.

Auctions: Generally speaking, the more buyers and sellers there are in a market, the more the situation becomes close to the perfect market scenario. An extreme opposite case is where there is only a single seller which is a case of an auction. The auction rules define the social choice: the identity of the winner.

Governments Policy: Governments routinely have to make decisions that affect a multitude of people in different ways: Should a certain bridge be built? How much pollution should we allow? How should we regulate some sector? Clearly each citizen has a different set of preferences but a single social choice is made by the government.

Networks: When running in an environment with multiple owners of resources or requests, the algorithm(protocol) must take into account the different preferences of the different owners. The algorithm should function well assuming strategic selfish behaviour of each participant. Thus we desire a Mechanism Design approach for a multitude of algorithmic challenges such as routing, task scheduling etc. Maybe the most interesting example of such a setting is the Internet, in which the various agents - users and devices - are self-interested, with private preferences and goals.

In this section we only roughly sketched the field of mechanism design. In the sections to come we give formal definitions, and explain things further. Here is an outline of our Introduction. Section 1.1 introduces fundamental notions of game theory and gives two important solution concepts: *Nash equilibrium* and *Dominant-strategy equilibrium.* Section 1.2 introduces mechanism design: we define what is a *utility function*, what is a *mechanism*, what is a *social choice function*, what we mean when we say that *a mechanism implements a social choice function* and we see all that in an example. Furthermore, we present some very important and classic results in the field such as *the revelation principle* and the *VCG mechanisms.* In section 1.3, we move to voting methods, and lay the general electoral setting.

1.1 A brief Introduction to Game Theory

1.1.1 Basic Definitions

Game theory is a method to study a system of agents in conditions of strategic interaction. In game theory we model agents as *rational* and *intelligent*. An agent is *rational*, when he is self-interested, i.e. his objective is to maximize his own payoff. An agent is *intelligent*, if he knows all that *we* - as analysts of the game - know and if he can make all the inferences that *we* can make [**?**].

The games we will be talking about are known as *one-shot simultaneous games*. We say that a game consists of a set of n players, $\{1, 2, ..., n\}$. To play the game each player i will choose a *way of playing*. The fundamental concept of agent choice in game theory is expressed as a *strategy*. To play the game, each player i selects a strategy s_i from a set of strategies S_i he has available. The choice of strategy of each player will depend on his *preferences*; informally, each player has some kind of preferences which will determine which strategy he will chose to play. We will introduce a more formal way of perceiving the *preferences* of each player in the next section; for now this intuitive way will do.

We name *pure strategies* all the available choices an agent can make. A player may not choose a single strategy, but a probability distribution over her available strategies and therefore play each (pure) strategy with the given probability. In this case we talk about *mixed strategies*. Therefore the strategy s_i of each player may be either a pure strategy or a mixed strategy. It can be easily seen that the concept of mixed strategy is a generalization of pure strategy.

We use the notation $s = (s_1, ..., s_I)$ for the *strategy vector(strategy profile)* of all the agents, or equivalently $s = (s_i, s_{-i})$, where $s_{-i} = (s_1, ..s_{i-1}, s_{i+1}, ..., s_I)$, i.e the strategy vector of every agent except agent i.

The vector of strategies s selected by *all* players determines the outcome of the game. To specify the game we need to assign to each player a preference ordering over these different outcomes. The simplest way to do this is by assigning, for each player, a value to each outcome. This value may be perceived as the pay off of the player. In other words this value is a measure of the *happiness* of the player for the outcome: the greater the value, the greater the *happiness* of the player. Hence, we define a function $u(s) : S \rightarrow \mathbb{R}$. This function is called a *utility function*; we will define it more formally in the next section.

1.1.2 Solution Concepts

A number of solution concepts to compute the outcome of a game are defined, each of which makes different assumptions about the information available to agents and methods used by agents to select their *strategies*. The most intuitive solution concept in game theory is the notion of *Nash equilibrium*, which states that in equilibrium every agent will select a utility-maximizing strategy, given the strategies of every other agent.

Definition 1 (Nash equilibrium(NE))**.** A strategy vector s is said to be a Nash equilibrium if for all players i and each alternative strategy s'_i, we have that

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$$

In words, no player i can change his chosen strategy from s_i to s'_i and thereby improve his payoff, assuming that all other players stick to their strategies. Observe that such a solution is self-enforcing in the sense that once the players are playing such a solution, it is in every player's best interest to stick to his strategy.

The following distinction is made: If we consider pure strategies, we are talking about a *pure Nash Equilibrium*. If we consider mixed strategies we are talking about *mixed Nash Equilibrium*.

John F. Nash proved in 1951 that NE exists for mixed strategies in all games that have a finite set of strategies and players. Maybe it is this universality property under such lenient conditions and the intuitiveness of the NE that has made it the most widely accepted solution concept of the field.

It has however some serious shortcomings. For a NE to be played(in one-shot games), rather stringent conditions about the available information to the players must hold. Aumann and Bradenburger [**?**] have shown that those conditions are not as stringent as one might have expected(roughly everything to be *common knowledge* [1]) but still the necessary conditions for a NE to be played are much stronger than those of our next solution concept, dominant-strategy equilibrium. Furthermore, NE gets rather problematic as a solution concept in the event of multiple equilibria, a case that is often seen in practical applications, in the sense that even under the previous conditions the players do not know which equilibrium to coordinate on. Finally, another very serious drawback, that was pretty much overseen until recently, is the computational cost that each player must undergo, in order to solve game-theoretic problems so that she can choose her optimal*(best response)* strategy [**?**].

A stronger solution concept than NE is a *Dominant strategy equilibrium*. In dominant strategy equilibrium, each agent has the same best(utility-maximizing) strategy to play, independent of the strategies played by all other agents.

Definition 2 (Dominant strategy equilibrium)**.** The strategy s_i is called a dominant strategy

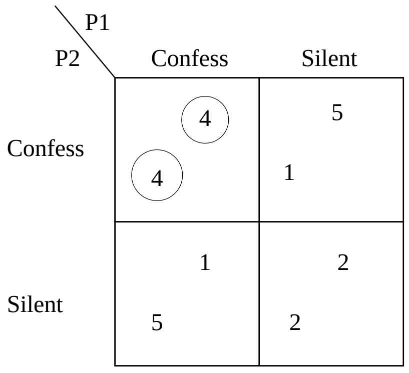
[1]A piece of information is *Common knowledge* when everyone knows it and everyone knows that everyone knows it and... ad infinitum.

if it (weakly) maximizes the agent's utility for all possible strategies of other agents:

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}) \qquad \forall s'_i \in \Sigma_i \text{ and } \forall s_{-i} \in \Sigma_{-i}$$

If all agents have a dominant strategy, these form a *Dominant strategy equilibrium.*

It is important to notice that a dominant strategy solution may not give an optimal payoff to any of the players. Also, it is easily verifiable that a dominant strategy equilibrium is also a Nash equilibrium. A very classic example of game that has a Dominant strategy equilibrium that does not bring the best output for either of the two players is the *prisoners dilemma.* In the following matrix the numbers represent the cost incurred on each player, expressed in years of jail he/she will get as a result of each pair of strategies played.



The prisoners Dilemma

Having a single dominant strategy for each player is an extremely stringent requirement for a game and very few games satisfy it. On the other hand, it is a very robust solution concept, because it makes no assumptions about the information available to agents about each other, and does not require an agent to believe that other agents will behave rationally in order to select his/her own optimal strategy. In this sense it is also very desirable from a computational complexity point of view, since it doesn't require any computations to be done about the *game theoretic reasoning* which will give the best response strategy of each player.

Looking ahead to *mechanism design*, we wish to design mechanisms which provide agents with a dominant strategy and also lead to a desirable outcome(the outcome is optimal according to some criterion). In mechanism design the dominant strategy equilibrium is much more desired than Nash equilibrium. The reason is that NE makes much stronger assumptions about the knowledge that must be available to the agents, that it is unrealistic to assume that they will indeed have such knowledge in the distributed setting of a mechanism. More precisely, for a player to realise she has a dominant strategy she only needs to know her own payoffs for each possible strategy vector. For a NE to be played she has to know the payoffs of all players for each strategy vector and further conditions must hold about what each player knows that the rest of the players know [?]. Thus we have established another drawback of NE: it is of no use in the field of mechanism design.

In this thesis we restrict ourselves in the use of dominant-strategy equilibrium.

1.2 Mechanism Design

1.2.1 Basic Definitions

Let us introduce the idea of the *type* of an agent, which is our way of modelling the preferences of an agent over different outcomes of a game. Let $\theta_i \in \Theta_i$ denote the *type* of agent i, from a set of possible types Θ_i. It is generally useful to think of θ_i as an ordering of the different possible outcomes. For example, if we have I possible outcomes, the preference of an agent could be $o_1 \succ o_2 \succ ... \succ o_I$, where $o_i \succ o_j$ means that outcome i is (weakly) preferred to j by the agent. Therefore the type of an agent defines a complete, transitive, reflexive binary relation on the set of all outcomes.

Definition 3 (Utility function)**.** An agent's preferences over outcomes $o \in O$, for a set O of outcomes, can be quantified in terms of a utility function that is parametrized on the type. Let $u(o, \theta_i)$ denote the utility of agent i for outcome $o \in O$ given type θ_i. Agent i prefers outcome o_1 over o_2 when $u(o_1, \theta_i) > u(o_2, \theta_i)$. Intuitively we say that the utility is the *measure of hapiness* of the agent.

The ultimate goal in mechanism design is expressed by a *social choice function*, which selects the optimal (in some sense) outcome given agent types.

Definition 4 (Social Choice function)**.** Social choice function
$f : \Theta_1 \times .... \times \Theta_I \to O$ chooses an outcome $f(\theta) \in O$, given the types $\theta = (\theta_1, ...\theta_I)$.

Definition 5 (Mechanism)**.** A mechanism $M = (\Sigma_1, ..., \Sigma_I, g(s))$ defines the set of strategies Σ_i available to each agent, and an outcome rule
$g : \Sigma_1 \times ... \times \Sigma_I \to O$, such that $g(s)$ is the outcome given out by the mechanism when the players choose to play the strategy profile $s = (s_1, ..., s_I)$.

In words, a mechanism defines the strategies available to each player and the method used to select the final outcome based on the strategies each agent plays. We will soon see an example.

Game theory is used to *analyze* the outcome of a mechanism. Given mechanism M with outcome function $g(s)$, we say that a mechanism *implements* social choice function $f(\theta)$, if the outcome computed with equilibrium agent strategies is a solution to the social choice function for all possible agent preferences.

Definition 6 (Social Choice function implementation)**.** Mechanism $M = (\Sigma_1, ..., \Sigma_I, g(s))$ *implements* social choice function $f(\theta)$ if $g(s_1^*(\theta_1), ..., s_I^*(\theta_I)) = f(\theta), \forall\theta$, where strategy profile $(s_1^*(\theta_1), ..., s_I^*(\theta_I))$ is an equilibrium solution to the *game induced by M.*

It is interesting to observe that the above definition doesn't assume a concrete equilibrium notion, it may be Nash, Bayesian-Nash, dominant strategy or any other concept. Our desire is generally for as strong a solution concept as possible.

Putting together what we have mentioned so far, the mechanism design problem is to design a mechanism - a set of possible agent strategies and an outcome rule - to *implement* a social choice function with desirable properties, in as strong a solution concept as possible; dominant is the most preferred solution concept because it makes less assumptions about agents.

The problem of designing a mechanism to implement a social choice function is far from trivial. As we have mentioned above, each agent is self-interested (rational), which means that her mere goal is to maximize her utility; therefore she may act strategically, trying to manipulate the mechanism if she thinks this will improve her utility(and since we assume she is *intelligent*, she can find such a manipulative technique if one exists). For example, in a mechanism where the strategy of each agent is to report her preference, she may lie about it. The problem is that when many agents act in this way, the outcome may be far from the optimal we desired. It is therefore the task of the mechanism to guarantee that the agents have good reason to act in a way that does not jeopardize the good function of the mechanism, in essence to act according to their true preferences and not strategically.

Let us see all that we have mentioned so far in this section in a rather illuminating example.

1.2.2 An Example

Alice and Bob are brother and sister. As every usual brother and sister they quarrel from time to time. There are two states of the world: *Alice started it (the quarrel)* or *Bob started it.* Depending on who started the fight, they both have an inner sense of justice which yields a different ordering on the punishments they may receive from their parents, ranging from what each of them perceives as the most lenient punishment to the most harsh. Although only one of them starts the fight, they will both receive the same punishment because of quarrelling (and annoying the parents who are watching TV).

Alice started it		*Bob started it*	
Alice	***Bob***	***Alice***	***Bob***
no pocket money	apologise to each other	apologise to each other	no pocket money
no video games	no going out	no video games	no going out
no going out	no pocket money	no pocket money	no video games
apologise to each other	no video games	no going out	apologise to each other

Alice's & Bob's Preference orders

In our already introduced terminology, the set of possible types for both Alice and Bob is $\Theta_{Alice} = \Theta_{Bob} = \{$Alice started it, Bob started it$\}$. Their parents want to give them a *relatively just* punishment, meaning that no child will perceive it as extremely harsh for the situation. Of course each kid may try to get away with what he/she perceives as the most lenient punishment, if this is allowed. This requirement for *reasonable justice* is the optimality desideratum we see in every mechanism design setting; both kids however take interest only in getting away with the most lenient punishment and do not share their parents' efforts for *reasonable justice* (in our terminology, they are *rational*). Let as assume that the parents feel *reasonable justice* has been restored if they end up with the most lenient punishment that is not placed more than one place apart in the preference orders of both children, no matter which state they are in.

According to the above table, the social choice function the parents would use, is $f(\textit{Alice started it}) = \textit{no going out}$, and $f(\textit{Bob started it}) = \textit{no video games}$. The crucial point here is that the parents, who are to restore justice do not know *the state of the world*, or who started the fight. Of course the kids both know who started it but this is not necessarily helpful to the parents; kids tend to lie about things like that. If the parents knew it, things would be easy for the optimal punishment to be chosen. However they do not know that.

What can they do? A first approach would be to simply ask the children who started the fight. Experience has shown that this approach is not very helpful and let us see how game theory verifies this fact. Actually "just asking" constitutes a - naive - mechanism. The strategies provided to each child is to report their type and the outcome rule is the following: if both kids agree on who started the fight, the optimal (most just punishment) can be imposed. However if the kids disagree, then the parents have no choice but to flip a coin between the two punishments (of course we could say that in this latter case no punishment is imposed but then the children could get away with no punishment, which would make our whole discussion pointless). We must note however that Alice has incentive to report that *Bob started it* in both cases since she prefers *no video games* to *no going out* in both cases. This way if Bob also reports *Bob started it* she will receive a better punishment with a probability of 1, if Bob reports *Alice started it* she has increased her probability of getting a preferred punishment from 0 to 0.5. For the same reason Bob has incentive to claim that *Alice started it* no matter what the truth is. Hence, the dominant strategy of both children lead them disagreeing and the parents end up choosing the punishment randomly, which does not sound a very just system.

Let us see how the justice system of the parents can get much better by designing a slightly more complex mechanism.

		Bob	
		Left	**Right**
Alice	**Top**	appologise to each other	no video games
	Bottom	no going out	no pocket money

A more complex Mechanism

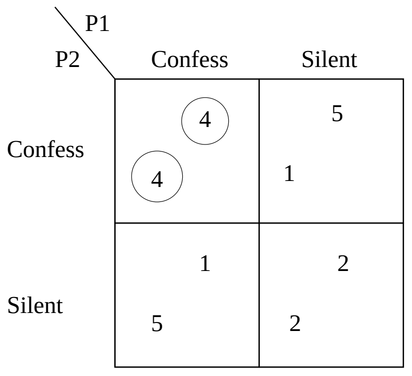
The parents have the children participate in the above game. Each kid has to choose between two strategies. Alice chooses between "Top","Bottom" and simultaneously Bob chooses between "Left", "Right"; the outcome of those choices (the outcome rule) is given in the corresponding entry of the matrix. Let us observe that, in the case "Alice started it", Bob is better off choosing "Left" regardless of what Alice does: if she plays "Top", then "Left" leads to "appologise to each other" as the outcome (Bobs preferred outcome), whereas "Right" gives rise to "no video games". If she plays "Bottom", then "no going out"(which Bob prefers) is the consequence of playing "Left", while "Right" leads to "no pocket money". That is, "Left" is the *dominant strategy* for Bob in state "Alice started it". In the state *Alice started it* Alice too has a dominant strategy: "Bottom". If he plays "Left" she prefers "no going out" to "appologise to each other", if he plays "Right" she prefers "no pocket money" to "no video games". Therefore, in the state *Alice started it* ("Bottom", "Left") is the (unique) Dominant NE. Furthermore, and this is the critical point, the resulting outcome, "no going out", is optimal in this state. Analogously we can reason that the unique dominant Nash equilibrium in the state "Bob started it" is ("Top","Right") and this again results in the optimal result; "no video games".

We have seen that in either state, the mechanism we proposed achieves the optimal outcome even though (i) the mechanism designers (the parents) do not know the actual state, and (ii) Alice and Bob are not interested in the goal of the parents for justice; they only want to have the most lenient punishment they can for themselves. More precisely, because the (dominant) Nash equilibrium outcomes of the proposed mechanism coincide with the optimal outcomes in each state (each type), we say that the mechanism *implements* the (parents') social choice rule.

1.2.3 Some more Definitions

Definition 7 (Pareto optimality)**.** A function is said to be pareto optimal, if it gives outcomes, for which no alternative outcome is strictly preferred from at least one agent, and weakly preferred from all other agents.

In other words, in a *pareto optimal solution*, no agent can be made happier, without making at least another agent less happy. A typical example of a non-pareto-optimal outcome is the dominant solution in the prisoners dilemma, since if they both switched to "Silent" they would both decrease their cost. Actually the state ("Silent","Silent") is pareto optimal.



The prisoners Dilemma

Utility functions may have arbitrary form(as long as they meet some very trivial restrictions): they may be linear, exponential etc. A form that has some very desirable properties for our purposes is the quasi-linear form.

Definition 8 (Quasi-linear preferences)**.** A quasi-linear utility function for agent i with type θ_i, is of the form:

$$u_i(o, \theta_i) = v_i(x, \theta_i) - p_i,$$

where x is some *choice* made as a result of the outcome o, p_i is a payment made by the agent to the system, v_i is the value the agent assigns to x(the agent's *valuation function*)

What is important to note is that, with quasi-linear agent preferences we can separate the outcome of a social choice function into a choice $x(\theta) \in K$ and a payment $p_i(\theta)$ made by each agent i:

$$f(\theta) = (x(\theta), p_1(\theta), ..., p_I(\theta))$$

Respectively, the outcome rule, $g(s)$, in a mechanism with quasi-linear agent preferences, is decomposed into a choice rule, $k(s) \in K$, that selects a choice from the choice set given strategy profile s, and a payment rule $t_i(s)$ that selects a payment for agent i based on strategy profile s.

We define this more formally since we will use it in the well known VCG mechanisms.

Definition 9 (quasi-linear Mechanism)**.** A quasi-linear mechanism $M = (\Sigma_1, ..., \Sigma_I, k(s), t_1(s), ..., t_I(s))$ defines: the set of strategies Σ_i available to each agent; a choice rule $k : \Sigma_1 \times ... \times \Sigma_I \to K$, such that $k(s)$ is the choice implemented for strategy profile $s = (s_1, ...s_I)$; and transfer rules $t_i : \Sigma_1 \times ... \times \Sigma_I \to \mathbb{R}$, one for each agent i, to compute the payment $t_i(s)$ made by agent i

Typical examples of quasi-linear mechanisms are auctions. In an auction the outcome has two components: a choice x, which is the agent who wins the auction, and the payments p_i each agent has to make to the auctioneer (usually only the winner pays a non-zero amount). Furthermore, the valuation function is some positive value if she wins and 0 if she doesn't.The utility she receives is $v_i - p_i$, therefore quasi-linear.

Social Choice functions have properties that may be desired in various contexts. One basic, is (allocative) efficiency. We say that an outcome is efficient, when it maximizes the total value over all agents:

Definition 10 (Social Choice function Efficiency)**.** Social choice function $f(\theta) = (x(\theta), p(\theta))$ is (allocatively) efficient if for all preferences θ

$$\sum_{i=1}^{I} v_i(x(\theta), \theta_i) \geq \sum_{i=1}^{I} v_i(x', \theta_i) \qquad \forall x' \in K$$

Therefore (allocative) efficiency captures what we vaguely say as "social good": a function is allocative efficient when the society as a whole receives the maximum utility possible.

In general, we say that *a mechanism has a property P*, if it implements a social choice function with property P.

1.2.4 Direct-Revelation, Incentive compatibility and The Revelation Principle

Informally, a direct-revelation mechanism is a mechanism in which the only actions (strategies) available to agents are to make direct claims about their preferences to the mechanism. In other words, in a direct-revelation mechanism the strategy of agent i is to report some type $\hat{\theta}_i = s_i(\theta_i)$, based on its actual preferences θ_i. Actually the "naive" mechanism of our example was a direct-revelation mechanism. When the strategy chosen is the true type θ_i we say that the strategy is *truth revealing.*

Definition 11 (Direct-revelation mechanism)**.** A direct-revelation mechanism $M = (\Theta_1, ..., \Theta_I, g(s))$ restricts the strategy set $\Sigma_i = \Theta_i$ for all i, and has outcome rule $g : \Theta_1 \times ... \times \Theta_I \to O$, which selects an outcome $g(\hat{\theta})$ based on reported preferences $\hat{\theta} = (\hat{\theta}_1, ..., \hat{\theta}_I)$.

Definition 12 (Incentive-compatible Mechanism)**.** An incentive-compatible mechanism, is a direct-revelation mechanism in which agents report *truthful information* about their preferences in equilibrium, i.e. the equilibrium strategy profile $s^* = (s_1^*, ..., s_I^*)$ is every agent to report his *true* preferences to the mechanism.

Incentive compatibility is a very important notion. It captures the essence of designing a mechanism, which is to overcome the self-interest of the agents. In an incentive-compatible mechanism each agent will choose to reveal his private information truthfully, rather than reporting any possible *lie*, since truthfulness will give him higher (in the weak sense) utility. Intuitively, in an incentive compatible mechanism what we do is that *we align each player's self-interest with our desire to know his true preferences*, or in other words that *we internalize the goal of the mechanism in the self-interest of each player.*

Definition 13 (Strategy proofness)**.** A mechanism is strategy proof, if it is incentive compatible and the equilibrium we arrive at is a dominant-strategy equilibrium.

A strategy proof mechanism implements a dominant-strategy equilibrium which, as we saw in the previous section, as a solution concept has very desirable properties, both game-theoreticaly and computationally. Therefore strategy-proofness is a very disirable property of a mechanism.

A well celebrated result in mechanism design is the *Revelation Principle*, which has proven to be a powerful theoretic tool. We first give its formal definition and then consider its importance and implications.

We give the version of the Revelation Principle for dominant strategy equilibria (although it has been extended for Bayesian-Nash equilibria too).

Theorem 1 (Revelation Principle)**.** Suppose there exists a mechanism M that implements the social-choice function $f(\cdot)$ in dominant strategies. Then - no matter how complex this mechanism is - there exists a direct-revelation mechanism M' that truthfully implements $f(\cdot)$ in dominant strategies; the new mechanism is a strategy-proof mechanism.

Proof. If $M = (\Sigma_1, ..., \Sigma_I, g(\cdot))$ implements $f(\cdot)$ in dominant strategies, then for the equilibrium strategies $s^*(\cdot) = (s_1^*(\cdot), ..., s_I^*(\cdot))$ it holds that $g(s^*)(\theta) = f(\theta)$ for all θ. Since s^* is the dominant equilibrium strategy vector it holds by definition that for all i and $\theta_i \in \Theta_i$

$$u_i(g(s^*(\theta_i), s_{-i}), \theta_i) \geq u_i(g(\hat{s_i}, s_{-i}), \theta_i), \qquad \forall \hat{s_i} \in \Sigma_i, \forall s_{-i} \in \Sigma_{-i}.$$

Substituting $s^*_{-i}(\theta_{-i})$ for s_{-i} and $s_i^*(\hat{\theta}_i)$ for $\hat{s_i}$ we get:

$$u_i(g(s^*(\theta_i), s^*_{-i}(\theta_{-i})), \theta_i) \geq u_i(g(s_i^*(\hat{\theta}_i), s^*_{-i}(\theta_{-i})), \theta_i), \qquad \forall \hat{\theta}_i \in \Theta_i, \forall \theta_{-i} \in \Theta_{-i}.$$

Finally, since $g(s^*(\theta)) = f(\theta)$, for all θ, we have:

$$u_i(f(\theta_i, \theta_{-i}), \theta_i) \geq u_i(f(\hat{\theta}_i, \theta_{-i}), \theta_i), \qquad \forall \hat{\theta}_i \in \Theta_i, \forall \theta_{-i} \in \Theta_{-i}.$$

This is precisely the condition that suggests that when the outcome rule g' of the mechanism M' is equal to the social choice function $f(\cdot)$ the vector of types $\theta = (\theta_1, ..., \theta_I)$ is a dominant strategy equilibrium. On top of that since, trivially, the outcome of the mechanism coincides with the outcome of the social choice function in equilibrium, we have proven that social choice function f is truthfully implementable in dominant strategies by a direct revelation (strategy proof) mechanism. $\square$

The above theorem can be proven for quasi-linear utility functions too, proving that there exists a mechanism for which also the payments of the players is the same with the original mechanism.

In words, the Revelation Principle states that any mechanism can be transformed into an equivalent *incentive-compatible direct-revelation mechanism* that implements the same social-choice function! We must note here, that the Revelation Principle only guarantees the existence of such a mechanism. It doesn't say anything about how to construct it, or how complex this mechanism may be. The only thing that we know is that if an indirect-revelation and/or non-truthful mechanism solves a distributed optimization problem, then we would also expect a direct-revelation truthful implementation of the problem.

However as we said earlier, the Revelation Principle is a powerful theoretic tool and has therefore some really important implications for the field of mechanism design. It actually suggests something very striking: in order to identify which social choice functions are implementable in dominant strategies, we need only identify those functions $f(\cdot)$ for which truth-revelation is a dominant strategy for agents in a direct-revelation mechanism with outcome rule $g(\cdot) = f(\cdot)$. In other words, when considering dominant strategies, it is possible to restrict our attention to truth-revealing direct-revelation mechanisms.

Except for restricting the field that we have to search, the Revelation Principle gives also some guidance on what *is* or *is not* possible; particularly the two following points hold:

- Suppose that the only direct, incentive compatible mechanisms with useful properties $P1$, $P2$ and $P3$ are in the class of mechanisms M. It follows that any mechanism m with properties $P1$, $P2$ and $P3$ must be *outcome equivalent* to a direct mechanism in M, in the sense that m must implement the same outcome as a mechanism in this class for all possible agent types.

- Suppose that *no* direct, incentive compatible mechanism has properties $P1$, $P2$ and $P3$. It follows that there can be no mechanism (direct or otherwise) with properties $P1$, $P2$ and $P3$.

1.2.5 Vickrey-Clarke-Groves (VCG) Mechanisms

In their significant papers, Vickrey [?], Clarke [?] and Groves [?], proposed the Vickrey-Clarke-Groves family of mechanisms for problems in which agents have quasi-linear preferences. The VCG mechanisms are *allocatively-efficient* - maximizing the sum of agents utilities - and *strategy-proof* - direct-revelation, incentive compatible, dominant-strategy-solution mechanisms. They are one of the most classic possibility results in the whole field of mechanism design.

In fact it has been proven by [?] that:

Theorem 2 (VCG uniquiness)**.** The VCG mechanisms are the only allocatively-efficient and strategy-proof mechanisms for agents with quasi-linear preferences and general valuation functions, among all direct-revelation mechanisms.

Putting in practice what we mentioned in the end of the previous section, by the revelation principle we know that any kind of mechanism(e.g. no matter how complex it may be) that achieves allocative-efficiency in dominant-strategy implementation must implement a VCG outcome.

Definition 14. VCG mechanisms are direct-revelation quasi-linear mechanisms. That informally means that the strategy of each agent is to report a type $\hat{\theta_i}$ and the utility function of each player is quasi-linear (for formal definitions of these notions see section 1.2.3). So to define the mechanism we have to give the choice rule, $k(\hat{\theta})$ and the payment rule, $t_i(\hat{\theta})$.

Given reported types $\hat{\theta} = (\hat{\theta_1}, ...\hat{\theta_I})$ the choice rule is:

$$k^*(\hat{\theta}) = \arg\max_{k \in K} \sum_i v_i(k, \hat{\theta_i})$$

Choice k^* is the selection that maximizes the total reported value over all agents.

Every agent pays to the mechanism what the payment rule tells him to pay, i.e $p_i = t_i$. The payment rule is defined as:

$$t_i(\hat{\theta}) = h_i(\hat{\theta_{-i}}) - \sum_{j \neq i} v_j(k^*, \hat{\theta_j})$$

$h_i : \Theta_i \to \mathbb{R}$ is an arbitrary function. This freedom in selecting $h(\cdot)$ leads to the description of VCG as a *family* or *class* of mechanisms. We will come back to $h(\cdot)$ in a while.

Our claim is that VCG mechanisms are *allocative efficient* and *strategy proof.*

Allocative efficiency is self-explanatory by the way we defined the choice rule k^*.
We will give an intuitive explanation of strategy proofness. The key idea lies in the term $-\sum_{j \neq i} v_j(k^*, \hat{\theta_j})$, which means that each player is paid an amount equal to the sum of the values of all other players. When this term is added to his own value $v_i(k^*, \hat{\theta_i})$, the sum becomes exactly the total social welfare of k^*. Thus this mechanism *aligns all players' incentives with the social goal of maximizing social welfare*, which is exactly archived by telling the truth.

Now we come back to the first term of the payment rule $h_i(\theta_{-i})$. This term has no strategic implications for player i since it does not depend, in any way, on what he says, and thus from player i's point of view it is just a constant. Of course, the choice of h_i does change significantly how much money is paid and in which direction, but we will not explore this further. We will only mention that a classic choice for h_i is what is known as the *Clark pivot payment.*

Since in this thesis we are primarily concerned with the computational aspects of the problems we discuss, we want to note that computing the outcome of a VCG mechanism is NP-hard in general. The most interesting (or discouraging) part however is not this. A common way to circumvent computational intractability in computer science is to resort to approximation. The interesting (or discouraging) part is that in VCG mechanisms approximation and incentive compatibility do not mix [**?**]. In other words it is proven that if we resort to approximation the agents loose their incentive to act truthfully!

1.3 Laying the Electoral Setting

In general a *voting system*[2] provides a framework for aggregating voters' preferences, in a way that the outcome is "optimal" in some sense (we will see what this optimality demand may be).

The general electoral setting that we consider consists of:

- A set of m alternatives C(candidates)
- A set of n voters V
- A set L of linear orders on C. Every member of L is a transitive, complete and strict (antisymmetric and irreflexive) order on C. We denote as $a \succ_i b$ the case where a is (strictly) preferred to b by agent i.
- A subset W of C, who are the winners
- A *voting method* E that dictates the rules of the procedure

We note the outcome of a voting procedure with a set of candidates C and a set of voters V under voting method E as $E(V, C)$

Definition 15 (Voting Scheme)**.** A voting scheme is a mechanism whose strategies and outcome rule are defined by the *voting procedure.*

The applications of voting are innumerable. Societies use elections to select their leaders, establish their laws, and decide their policies. However, practical applications of elections are not restricted to people and politics. Parallel algorithms start by electing leaders, multiagent systems sometimes use voting for the purpose of planning, web search engines can aggregate results using techniques based on elections.

In the web for example, one may consider web pages as voting on other pages by linking to

[2]In the literature, the terms "voting system", "voting schemes", "voting rules", "voting methods", "voting procedures" are often used interchangeably. Here we make a slight distinction but we, too, will use them interchangeably when there is no confusion.

them (an idea used in Google's PageRank algorithm), or may consider humans to be voting on pages at a site by the time they spend on each. Dwork et al. [**?**] address the general problem of aggregating various rankings in one, which is quite similar to voting. They suggest designing a meta search engine that treats other search engines as voters who provide their ranking on the web pages (candidates) or performing multi-criteria searches by aggregating the results of more single-criteria searches.

We note that such vast range of applications implies that the voting setting may vary greatly from one case to another. For example in the political election case the voters are infinite compared to the candidates while in the web search framework this relation is reversed. Therefore the ratio of voters to candidates may differ a lot from case to case.

Furthermore, we note that many voting scenarios make just as much sense when each voter has a different voting power. For example, U.S. presidential elections are in some sense weighted (different states have different voting powers in the Electoral College). When agents vote in partisan groups (e.g. in parliament) the weights may correspond to the power of the group (all members of a group are expected to vote identically, so we may model each group as one voter with a corresponding weight). Shareholders in a company have votes weighted by the number of shares they own. Search engines in the above example could be weighted by their trustworthiness. Thus, weighted voting as well, makes sense in many settings and indeed many of the results we will come across refer to weighted voting.

Lastly we would like to stress the fact that "voting" is a nice intuitive way of perceiving the hole field of social choice theory; every social choice decision can be perceived as a voting procedure. We have already said that the goal of social choice is to aggregate different preferences in a joint decision which is optimal in some sense. This can be seen as a voting procedure, where the different preferences are aggregated in a joint decision, the winner of the election. Therefore although from now on we will primarily be referring to voting, one can keep in mind that what we say applies to all social choice theory.

Having already talked about mechanism design and strategy-proofness we want to establish what our ideal goal would be: to design a voting rule (or prove that one of the existing satisfies this property) which provides as a dominant strategy to each agent to report his true preference order about the candidates. We will soon see a very important theorem has been proven, which roughly states that only trivial voting rules can have this property! But more on this later.

Chapter 2

Voting Schemes: The (Traditional) Social Choice Perspective

In voting theory we say that the outcome of the procedure must be "optimal" in some sense; we want to proclaim winner "the most desired candidate". We will soon see that this desire for optimization is not that easily captured by the various voting systems and various problems may come up. We begin here with a prominent such problem, known as *"condorcet's paradox"*.

Let us consider three voters v_1, v_2, v_3 and three candidates a, b, c. The preferences of the voters are the following:

v_1: $a \succ b \succ c$
v_2: $b \succ c \succ a$
v_3: $c \succ a \succ b$

The joint choice of the majority of the candidates is $a \succ b \succ c \succ a$, which is clearly inconsistent. In words, for any candidate that is proclaimed winner, there will be a majority of voters who would want to change the chosen outcome.

The importance of the condorcet paradox is the following: *Rational individual preferences can yield irrational aggregated preferences in terms of majority!*

2.1 Some important voting rules

Let us present some of the most widely known voting procedures. Maybe the first thing that comes to a person's mind when she hears the term "voting" is the *plurality rule.* It is one of the most easy to apply and widely used voting systems around the world, for centuries.

2.1.1 The Plurality Rule

The procedure of the plurality rule is very easy: each candidate gets a vote for each time she is placed at the top of a voter's preference order; the candidate who receives the most votes is proclaimed the winner. A small variation of the plurality rule is the *majority rule.* The procedure is the same only the winner has to receive more than 50% of the votes. If this is not accomplished in the first round, a second round is held only between the top two candidates of the first round.

The thing now is that, when there are only two candidates, plurality rule (which is equivalent to majority rule) captures fully the notion of the "most desired candidate". Between two candidates the one that is preferred by most voters is unequivocally the "best choice". However it is very common that the plurality rule is considered the best way of choosing the winner in the case that there are more than two candidates. In such cases however, although common sense may say that plurality or majority means wide acceptance, problems may appear. The source of these problems is rather intuitively explained: in both plurality and majority rules only the top candidate of each voter is taken into consideration; the rest of his preference order is ignored.

Let us now see an example of such problems of the plurality and majority rules. Suppose there are 4 candidates (x,y,z,w) and 100 voters. In the next figure we see four different preference orders and the number of voters that has each.

20 voters: $z \succ x \succ y \succ w$
24 voters: $y \succ z \succ x \succ w$
26 voters: $x \succ z \succ y \succ w$
30 voters: $w \succ z \succ x \succ y$

According to the plurality rule, the winner of the election is w.
What is the problem with this choice?
First of all, w is only preferred by a minority, a 30% of all voters. Therefore the winner is the optimal choice of only a minority of the voters. More precisely, the rest think w is the worst possible choice. Plurality rule however proclaims him the winner.

Second, observe that w would lose in terms of majority by all other candidates if we held a head-to-head comparison of each pair of candidates. Citing the words of Marquis de Condorcet, "The apparent will of the plurality may in fact be the complete opposite of their will".

One might think that the majority rule would give a more rational outcome since it requires that the winner has a more than 50% acceptance. Applying the majority rule, candidate x

would win in the second round. However this outcome again seems unfair, since 74% of the voters prefer candidate z to candidate x. In the majority rule, some irrationality seems to slip in because of the second round.

Therefore, up to now we have established that the apparent "fairness" of the plurality and majority rules as "capturing the will of the majority" can be quite misleading. As we already said, the main reason for such problems is that both these rules take into account only the top preference of each voter and ignores the rest.

2.1.2 The Condorcet Criterion

These problems are known for more than four centuries now. In the late 1700s two influential personalities of the french intelligentsia were occupied with them and each proposed his own solution to the inconsistencies of the plurality and majority rules. Those two were Marie Jean Antoine Nicolas de Caritat, marquis de Condorcet (philosopher, mathematician, and political scientist) and Jean-Charles, chevalier de Borda (mathematician, physicist, political scientist, and sailor).

Marquis de Condorcet claimed that the most natural property that a voting rule should satisfy is the *Condorcet Criterion*. According to the Condorcet Criterion a winner of an election is the one who would beat all other candidates in head-to-head elections in terms of majority. In other words, if there is a winner who is preferred to all other candidates by the majority of voters, he is the *Condorcet Winner*. This property is indeed a very intuitive way of identifying the "most wanted candidate". In our above example, candidate z is a condorcet winner (note that z is not proclaimed winner by neither the plurality nor the majority rule). The problem is that there are circumstances where no Condorcet winner exists. This fact was noted by Marquis de Condorcet himself in his important paper of 1785 [**?**]. A typical example is "Condorcet's paradox" we saw earlier; whenever such preference circles exist there is no condorcet winner.

Although a condorcet winner may not always exist, it is considered a notion of fundamental importance in voting systems, capturing "fairness" when it is met, or "un-fairness" when it is violated by the outcome of the election. The *Condorcet rule* is the voting rule that proclaims winner the condorcet winner when one exists while the winner remains unidentified when no condorcet winner exists.

2.1.3 The Borda Count Rule

The other important researcher of the field that we mentioned is Jean-Charles, chevalier de Borda. He proposed (somewhere between 1781 and 1784) another voting method, known as

Borda score. Suppose there are m candidates. Each voter assigns to each of the m candidates an integer score in the range $[0, m-1]$. The scores assigned are summed for each candidate, and the one with the highest score wins. The Borda-count rule tackles indeed the drawback of the plurality and majority rules of ignoring all preferences except the top. However, rather peculiar things can happen with the Borda count rule, and Condorcet himself was strongly opposed to it.

Let us see an example demonstrating a problem of the Borda-count rule; the Borda rule does not respect the condorcet winner even when one exists. The following table shows the preferences of 7 voters and the corresponding borda scores of each of the 4 candidates.

v_1: $z \succ x \succ y \succ w$
v_2: $y \succ x \succ w \succ z$
v_3: $x \succ y \succ z \succ w$
v_4: $x \succ y \succ z \succ w$
v_5: $y \succ z \succ w \succ x$
v_6: $y \succ z \succ w \succ x$
v_7: $w \succ x \succ y \succ z$

According to these preference orderings the Borda score of each candidate is:
$w : 6$
$x : 12$
$y : 15$
$z : 9$

According to the Borda count rule candidate y is the winner. However candidate x is a Condorcet winner. Therefore, Borda count rule, like plurality and majority rules do not respect the Condercet Winner when one exists.

2.1.4 Approval-Veto Voting

In *approval voting* each voter gives his approval to a number of candidates. He does not rank the candidates, he only divides them to approved and disapproved. A variation is *k-approval voting* where each voter gives his approval for exactly k candidates. Let us note that 1-approval voting is equivalent to plurality voting. *Veto voting* is the opposite of approval voting, each voter approves all candidates except one.

2.1.5 A Unified perception of Scoring Protocols

It will be of great use for our discussion to see almost all of the voting rules we have mentioned as members of a class (family) of rules, the *scoring rules* [**?**]. Namely a scoring rule (for m-candidate elections) is defined by a *scoring vector* $a = (a_1, a_2, ..., a_m)$, satisfying $a_1 \geq a_2 \geq ... \geq a_m$. Each voter is represented by a voting list and the i_{th} candidate in a given voters list gains a_i points from this voter. Each candidate's point total is the sum of all the points she gets. Whoever gets the highest sum is a winner.

This way, plurality rule is defined by vectors of the form (), (1), $(1, 0)$, $(1, 0, 0)$,...., depending on the number of candidates m.

k-approval voting is defined by scoring vectors of the type $(\overbrace{1, 1, ..., 1}^{k}, \overbrace{0, ..., 0}^{m-k})$.

Veto voting is based on scoring vectors of the form (),(0),$(1, 0)$,$(1, 1, 0,)$,...

We have already discussed the notion of *Condorcet winner* and the fact that it does not exist always. Because however it is of such fundamental importance in voting, there have been proposed various voting rules which provide a fair compromise: when a condorcet winner exists they proclaim him the unique winner of the election, when none exists they proclaim winner(s) those who are closest to being a condorcet winner; such voting rules are sort of approximating the condorcet winner. Such systems are, Dodgson Rule, Copeland rule, Young rule, and Kemeny rule among others. Each rule assumes a different notion of proximity to the condorcet winner. We will see the 3 first.

2.1.6 The Dodgson rule

The Dodgson rule was proposed by the 19th century mathematician Charles L. Dodgson, better known as Lewis Carol, the pen-name with which he wrote his children books. In Dodgson's rule, each candidate c[1] is assigned a score, denoted by $dscore_{(c)}$. $dscore_{(c)}$ equals the smallest number of sequential exchanges (called "switches" henceforward) of adjacent candidates in the voters' preference lists that suffices to make c a Condorcet winner. Whoever has the lowest Dodgson score wins in Dodgson's system. When a Condorcet winner exists, he or she is the unique candidate with Dodgson score zero and thus is the unique Dodgson winner. We should note that in the case there is no Condorcet winner, Dodgson winners are not necessarily unique; however, at least one Dodgson winner always exists in Dodgson elections. In our formerly stated "Condorcet's paradox", all three a, b, c are winners under the Dodgson rule.

Let us give an example presented by BTT in [**?**] to better understand how we calculate

[1]when we want to refer to a distinguished candidate in this thesis, we usually name him c

the dodgson score of each candidate.

We have 4 candidates a, b, c, d and three voters with the following preference orders:
$v_1 : a \succ d \succ b \succ c$
$v_2 : b \succ c \succ a \succ d$
$v_3 : c \succ a \succ d \succ b$

The Dodgson score of a is 1, since switching c and a in the preferences of v_3 is sufficient to make a the Condorcet winner, and no fewer switches will do so. Similarly, the Dodgson score of b is 2, since at least 2 switches are necessary to beat both a and d, and this can be accomplished by switching b to the top of the preference order of v_1. The Dodgson score of c is 1, since c can be made to defeat b by a single switch within the preference order of v_2 to become a Condorcet winner. Finally, the Dodgson score of d is 3 since 2 switches are required to defeat a (for voters v_1,and v_2 or v_3), and an additional switch is required for d to defeat c(for voters v_2 or v_3). Thus a and c tie for Dodgson winner of this election.

2.1.7 The Copeland rule

As with dodgson rule, in this protocol every candidate c is associated with a score, *Copeland Score*, which we will note $cscore(c)$. For every two distinct candidates, i, j we have $C(i,j) = +1$, if i beats j in their pairwise election in terms of majority, $C(i,j) = -1$, if j beats i in their pairwise election in terms of majority, and $C(i,j) = 0$, in the remaining case. *cscore* is calculated for each candidate c, as $cscore(c) = \sum_{c' \neq c} C(c,c')$. The candidate with the highest *cscore* wins. It is easy to verify that when a condorcet winner exists, he will have the highest *cscore* since by definition he is the only one who beats all other candidates in pairwise election.

This definition is for the case that all voters are of the same weight. In case of weights there is a slight difference: $cscore(c)$ equals again $\sum_{c' \neq c} C(c,c')$ only now majority does not suffice to show the winner of the pairwise elections. Therefore $C(i,j) = 1$ if the sum of weights of the voters who prefer i to j is greater than the sum of weights of voters who prefer j to i. Analogously $C(i,j) = -1$ if the sum of weights of the voters who prefer j to i equals that of weights of voters who prefer i to j and $C(i,j) = 0$ if the sum of weights of the voters who prefer i to j equals that of voters who prefer j to i.

2.1.8 The Young rule

As with the two previous rules, in this protocol every candidate c is associated with a score. This score is the maximum subset of voters whose preferences can be taken into consideration, so that c is proclaimed condorcet winner. The candidates with the maximum such score are proclaimed winners. There is a nice way to perceive the young rule. We are facing an optimization problem which is to find the "best" candidate, i.e. the condorcet winner. The constrains are the preference orders of the candidates. When a condorcet winner exists, he is the unique winner, the optimal outcome that satisfies all constrains. If no condorcet winner exists, the problem is infeasible. In this case, the young rule examines which constrains have to be deleted in order to make the problem feasible. Winners under the young rule are those for whom the minimum number of constrains have to be deleted.

2.2 Social Choice Theory Criteria

The traditional track followed in voting rules was to examine whether the various voting rules satisfied certain properties, known as rationality criteria, that seemed only natural desiderata. We will see three of them.

Pareto optimality criterion, *unanimity criterion* or *collective rationality criterion*: if every voter prefers candidate x to y, then y is not chosen. The pareto optimality criterion seems a natural demand and non-compliance with it shows some *collective irrationality* of the voting rule.

All of the voting rules we examine are known to be compatible with the pareto optimality criterion: Copeland, Dodgson, Condorcet, Plurality, Approval, and Borda [**?**].

The next criterion is the Weak Axiom of Revealed Preference(WARP). In our already introduced notation suppose we have a subset of the candidates $C' \subset C$. We say that a voting rule E satisfies WARP if, when $C' \bigcap E(V,C) \neq \varnothing$, then $C' \bigcap E(V,C) = E(V,C')$.

From our voting rules only Condorcet, and Approval are known to be compatible with WARP [**?**]. Nurmi claims in [**?**] that also Plurality satisfies WARP, but this is not true, which is what Bartholdi et.al. and Hemmaspaandra et. al. correctly claim. Let us see the following example:

1 voter: $x \succ w \succ y \succ (z)$
2 voters: $(z) \succ x \succ w \succ y$
3 voters: $y \succ w \succ (z) \succ x$

The parenthesis mean that $C = \{x, y, w, z\}$ while $C' = \{x, y, w\}$. It is easily seen that

$E(V, C') = \{x, y\}$ while $E(V, C) = \{y\} \neq E(V, C') \bigcap C'$.

For our purposes we define a slightly different version of WARP, *unique-WARP* which says that a *unique* winner among a set of candidates remains a *unique* winner among each subset of candidates that include him/her.

Our third criterion is *path-independence.* If we have that $C = C_1 \bigcup C_2$, then a voting rule is said to be path-independent if it satisfies that: $E(V, C) = E(V, E(V, C_1) \bigcup C_2)$. In words, path-independence means that the outcome of the election is not affected by any partitioning of the candidates.

Nurmi claims that only approval voting satisfies path-independence.

2.3 Arrow's Theorem (Paradox)

For historical reasons and because of its striking importance, we present here a classic result of social choice theory known as *Arrow's theorem* or *Arrow's paradox.* Economist Kenneth Arrow (co-recipient of the 1972 nobel prize) proved in 1950 that there is no voting rule (Social Choice function) [2] over a set of 3 or more candidates that satisfies all 3 of the next very natural properties for all instances:

unanimity: If every voter prefers candidate x over y, then y is not elected as winner.

independence of irrelevant alternatives(IIA): If candidate x is proclaimed winner under the voting rule and the relevant position of x and y remains unchanged in the preferences of all voters, although the place of some other candidate z may change upwards or downwards, then y can not be proclaimed winner (although z may be proclaimed winner).

non-dictatorship: There is no voter whose first choice is always selected no matter what the other voters vote for.

What does Arrow's theorem mean?

We have already seen that the optimality desideratum of voting is not that easily captured by the various voting rules and outcomes that seem not at all optimal when examined in retrospect may come up. However we expect that all voting rules must be "fair" in a sense. That means that there are some very natural properties that we expect them to have. *Arrow's theorem* clashes our hopes for fairness too: it states that there is **no** fair voting system, in the

[2] Actually Arrow stated the theorem for the slightly different notion of Social Welfare Functions but it can be stated for SCF as well

sense that for each voting system there will exist some instances that such “fairness criteria” are violated. *Un-fairness is intrinsic in all voting systems!*

Let us demonstrate Arrow’s theorem by giving an example for the plurality rule, which is clearly not a dictatorship:
- plurality satisfies unanimity (trivial to verify)

- plurality violates IIA. Let us take the following instance:

v1: $x \succ y \succ z$
v2: $z \succ y \succ x$
v3: $y \succ z \succ x$
v4: $x \succ y \succ z$

Under plurality rule x is proclaimed winner

But if voters v2, v4 change their votes in the following way (note that the relevant positions of x and y remain unchanged),

v2: $y \succ z \succ x$
v4: $z \succ x \succ y$
while v1, v3 remain unchanged:

Now y is proclaimed winner!

2.4 The Gibbart Satterthwaite Theorem

Now it is time to state the very important Gibbard-Satterthwaite theorem. Although it can be stated in the general context of mechanisms in a natural way (actually Gibbard himself gives such a more general statement in his paper) we present it for voting rules because this is our main interest in this thesis.

We remind that a Voting scheme is *strategy-proof* if no agent can *ever* benefit from lying, no matter what other agents have reported as their preferences. To put it in our familiar game-theoretic context, truth-telling is a dominant strategy for every agent. If a voting scheme is not strategy proof, we say it is *manipulable.*

We say that a voting scheme satisfies *non-imposition* if it does not preclude beforehand any candidate from the outcome. In other words, the voting rule (social choice function) that determines the outcome must be *onto* the set of candidates.

In 1973 Gibbard in his seminal paper [?] and Satterthwaite [?] in his PhD thesis proved independently one of the most well known impossibility results in the field of mechanism design.

Theorem 3 (Gibbard-Satterthwaite impossibility Theorem on Voting Schemes)**.** Every voting scheme which satisfies non-imposition, over a set of at least three candidates, is either dictatorial or manipulable (i.e. not strategy proof).

The Gibbard-Satterthwaite theorem states that only "trivial" in a sense voting schemes are strategy proof. Triviality can be of two types. When there are only two candidates all "reasonable" voting rules are equivalent to the plurality rule, therefore it is a trivial case. If a voting rule is a dictatorship, it is trivial in the sense that it is not much of a voting rule, since it does not take into consideration the preferences of any of the voters but one.

Since the theorem was stated in 1973 many different proofs have been given. Here we prefer to take a more informal approach showing which parts of the theorem can be intuitively perceived.

- If there are only 2 candidates $\rightarrow$ there is no possible lie that will not harm the actually preferred candidate of each voter. *So the scheme is strategy proof*

- What changes when there are more than 2 candidates? $\rightarrow$ "good" lies exist. For example:

5 voters: $a \succ b \succ c$
4 voters: $b \succ a \succ c$
2 voters: $c \succ b \succ a$

Under let's say the plurality rule, candidate a is the winner. The 2 candidates that have c as their top choice may be tempted, since their top choice will not become a winner anyhow and the candidate that will become winner is their worst choice, to report a false preference order $b \succ c \succ a$ in order to make candidate b win, which is a better outcome than having a win. Therefore they have an incentive to lie, i.e. to manipulate the election.

Now, we note that only the one direction of the theorem is difficult to prove:

- If the voting rule is a dictatorship (the identity of the dictator is not known to the voters) $\rightarrow$ it is strategy proof: this direction is easy. Each voter has incentive to vote truthfully in case he/she is the dictator; the voting rule is indeed strategy proof.
- If there are more than 2 candidates and the voting rule is strategy proof $\rightarrow$ the voting rule is dictatorial: this direction is more difficult to prove.

We just note, in order to be precise, that the Gibbard-Satterthwaite theorem supposed that the voting scheme is single-valued, i.e. there is always exactly one winner. However it was generalized in 2000 by Duggan & Schwartz [**?**] for the case that the winner set is some subset(possibly empty) of the candidate set and indeed in the following sections we will mostly assume this generalized result.

The Gibbard-Satterthwaite theorem has great implications. Actually it clashed any hope of our "ultimate goal" being realized, designing a voting system that would "force"every voter to vote according to her true preferences in all situations! We have to be careful though. The Gibbard-Satterthwaite theorem is stated "in a sufficiently rich environment". This means that it makes no assumptions about the setting of the problem, for example about the form of the preferences of the agents. Although the fact that it is so general gives the theorem its striking importance, it gives us at the same time one way to circumvent it: by restricting the setting we are studying.

The thought of how to circumvent Gibbard-Satterthwaite impossibility theorem plays a rather central role in the whole field of mechanism design. For now we would like to enumerate some of the already known "paths" taken in order to do this in the literature:

- Restrict the structure of agents' preferences
- Use a weaker equilibrium concept
- Design mechanisms where finding a beneficial manipulation is computationally hard
- Use Randomization

Actually we have already seen one case of the first type: the VCG mechanisms. VCG mechanisms are as we saw strategy-proof (and non-dictatorial). We achieved that by introducing money, which is equivalent to studying a *restricted* field of the preferences (utilities) of each agent by assuming that they are quasi-linear.

A word of note about this "first path". There is no way for a designer of a mechanism to impose a type of preference on the agents, for example quasi-linear preferences. The results we have for restricted preferences state that when the setting of the problem justifies our assumption that preferences are modelled as a special type of function, for example quasi-linear functions, then our results can take place, e.g. VCG mechanisms.

The core of this thesis examines results that have to do with the third “path”, “the path less taken”.

Chapter 3

Voting Schemes: The Computational Perspective

Until the end of the 1980's the research in the field of voting rules focused on proving that voting procedures satisfied or not various criteria, which were in essence mathematical properties of the social choice function. It was Bartholdi et. al with a series of seminal papers that gave the research of the field a new direction. As they put it in [**?**], a natural refinement of the classic work in social choice, which until then had dealt with the distinction between the possible and the impossible, is a distinction between the tractable and the intractable.

First of all, it is only natural to want to know that the winner of an election can be computed in a "reasonable" time. Second, since the Gibbart-Satterthwaite theorem states that manipulation is inherent in voting in a way, it would be quite comforting to know that computing a *proper lie* is computationally intractable. These were their (and our) incentives to examine the various voting methods from a computational hardness point of view. We note that the idea of having a voting protocol that is hard to manipulate is quite similar to the cryptographic setting, where computational complexity is desirable (contrary to most of the rest of the field of Computer Science).

The known results in this direction, which we will present, argue that Computational Complexity is another aspect, apart from the traditional we presented in the previous section, one should consider when judging the various voting rules.

It is quite logical to say that a central goal of the computational study of elections, is to have natural, attractive voting systems whose winner-evaluation problems are computationally simple but nonetheless are hard to manipulate. In this section we study important known results concerning complexity issues of the following three directions:

- Winner Determination

- Manipulation & Bribery
- Control

We will define explicitly each of the previous notions in the sections to come but let's give an informal description of them: manipulation is to report false preferences in order to get a favourable outcome; *bribery* is what we all know: someone tries to buy-out some voters in order to get a favourable outcome; *control* is maybe not self-explanatory like bribery but it is quite easy to understand: suppose a chair oversees the voting procedure; we are concerned whether he/she can apply the procedure in such a way, so as to have a favourable outcome. In manipulation people "play" against the rules. In control the chair plays along with the rules but he/she may still have the power to twist the outcome to his/her favour.

A word of note here is in order.
(1) we restrict ourselves to deterministic voting schemes, although randomized have also been proposed
(2) the results we present make the assumption of complete knowledge of the manipulators, i.e. they know the preferences of all non-manipulator voters. This assumption is quite natural:

(2a) it is worst-case assumption, meaning that hardness under this assumption implies hardness under incomplete information; easiness results are weakened under this assumption but our main goal is to prove that systems are hard to manipulate

(2b) the results measure in this way only the *inherent* complexity of manipulation rather than any potential complexity introduced by the model of uncertainty.

Finally, we would like to stress out that "the literature on the complexity of election systems is a bit schizophrenic" [**?**]. Some areas of this literature - such as most of the work on the complexity of winner determination problems - focus on the issue of whether a particular candidate is (or can be made to be) *a* winner (possibly among others). Other areas of the literature on the complexity of election systems - such as most of the work on the complexity of control problems - focus on the issue of whether a particular candidate is (or can be made to be) the unique winner. In the literature on manipulation one finds multiple examples of focus on winners and of focus on unique winners, as well as cases where the manipulators is a subset of the voters or only a unique voter. Furthermore, some areas of this literature take the number of candidates as unbounded (it can get arbitrarily large depending on the setting), while others consider such an assumption unrealistic and assume that the number of candidates is a *small* constant.

Such differences in the assumptions probably underline the fact that as we said in the introduction, elections have so many applications that the setting can vary largely.

We will try to stress out the assumptions made in each case so that there is no confusion.

Before we begin let us give a brief overview of the complexity classes we will come across in this section.

3.1 A Brief Overview of Complexity Classes

Decision problems may belong to the following classes:

- P (Polynomial): The class of decision problems solvable in polynomial time (in terms of the size of the input). In Computer Science we say that the problems in this class are *tractable*, meaning that they are efficiently solved.

  It is known however that in some cases efficiency may be more of theoretic importance than any practical; for example, if we have that the time needed to solve a problem is n^{1000} (where n is the size of the input) the problem is technically in P but even for small inputs it will take impractically long to give an answer. Actually we will see such a case in this thesis.

- NP (Nondeterministic-Polynomial): The set of problems for which, if the answer is "yes", there exists a proof of this fact that can be verified in polynomial time. An extremely important subset of problems in NP are the NP-complete problems (actually every class has a subset of problems that are *complete* for the class). A problem is NP-complete if every other problem in NP can reduce to it in polynomial time.

- Θ_2^p There have been a number of definitions for this class of problems that have proven to be equivalent. A first definition is given by S. Zachos and C. Papadimitriou in 1983 [**?**]: The class of decision problems solvable in polynomial time, that can make $O(\log n)$ queries to an NP oracle (where n is the length of the input). Another equivalent definition is given by L. Hemaspaandra in 1989: the class of decision problems solvable in polynomial time that can make polynomially many (as to the size of the input) non-adaptive queries to an NP oracle (parallel access to NP) [**?**].

All problems in P are in NP and all problems in NP are in Θ_2^p, i.e

$$P \subseteq NP \subseteq \Theta_2^p.$$

Let us give an explanation of why these relations hold:
If a problem is in P we can solve it in polynomial time without a proof, so if we are given a proof we can solve it in polynomial time simply by ignoring the proof, and so the problem is also in NP. If a problem is in NP, then we can solve it by asking a single question to the oracle, so it is also in Θ_2^p.

It is commonly believed that there are problems that are in Θ_2^p that are not in NP, and problems in NP that are not in P, although this has never been proven!

3.2 Winner Determination

3.2.1 The General Case

Perhaps the most essential question when talking about any voting rule, is whether a given candidate is proclaimed a winner. Let us formalize this in our familiar decision-problem way:

Name: E **- Winner**
Input: Set of candidates C, one distinguished candidate c, a set of preference orders V on C
Question: Is c a winner under voting scheme E ?

First of all let us note that the above decision problem (first formulated by BTT), captures the complexity of the corresponding winner determination problem; in other words the problem of actually finding the winner of an election E is at least as hard as answering the E-winner decision problem. So if the E-winner decision problem is hard, the general winner determination problem is hard as well. What if it is easy? Then the general winner determination problem can be harder only by a factor of the number of the candidates .

In most voting schemes in use only polynomial time is required to answer the E-winner problem. For example, to solve *Plurality Winner* requires only $O(|V|+|C|)$ work to count the first-place votes and identify whether c is (one of) the candidate(s) with the most. Actually we can state the following easy to prove theorem:

Theorem 1. The Winner problem for all scoring protocols is in P.

Proof. One must go through the $|C|$ positions of the scoring vector for each of the $|V|$ voters, in order to calculate the score of each candidate. Then he finds the winner(s), with the maximum score and checks whether c is the winner (one of them). This requires $O(|C| \cdot |V| + |C|)$ steps. It is therefore polynomial both on the number of the candidates and on the number of the voters. $\square$

This result seems only natural since each voting system in practical use is expected to have a winner determination rule that is efficiently computed. The next result however, proven by Bartholdi, Tovey and Trick (from now on BTT for simplicity) [**?**] in 1987 may take someone by surprise:

Theorem 2. Dodgson Winner is NP-hard

It is interesting to note here that C. Dodgson gave a description of who is considered a winner under the Dodgson rule, but gave no algorithm to compute one. Probably he would not anticipate that his method, although explicitly described, could be proven "inpractical". The above result underlines the reasonable choice of analysing voting rules by a computational perspective.

We give a sketch of the proof. BTT first define two decision problems.

Name: Dodgson Score
Input: Set of candidates C, one distinguished candidate c, a set of preference orders V on C, a positive integer k
Question: Is the dodgson score of candidate c less than or equal to k ?

Name: Dodgson ranking
Input: Set of candidates C, two distinguished candidates c, c' a set of preference orders V on C
Question: Did c defeat c' in the election ?

The proof of the theorem is based on two lemmas. The first lemma claims that *Dodgson score* is NP-complete and the second that *Dodgson ranking* is NP-hard. The proof for both lemmas is a reduction from the NP-complete *Exact Cover by 3 sets* problem. Then, the two lemmas are combined to get the theorem.

We see that BTT gave only a lower bound of the complexity of the problem. Ten years later, in 1997, Hemaspaandra E., Hemaspaandra L. and Rothe managed to strengthen the hardness result to a completeness result. Namely they raised the lower bound and gave a matching upper bound of the hardness of the problem, yielding a *completeness* result.

Theorem 3. Dodgson Winner is $\Theta_2^p - complete$

To prove that Dodgson winner is in Θ_2^p (give the upper-bound) is quite easy. Given an instance (C, V, c), we can ask the NP oracle Dodgson Score polynomially many questions in parallel about all candidates. If c has the minimum score, c is the Dodgson winner. Otherwise not. However to prove that the problem is $\Theta_2^p - hard$ is far more complex. First of all, as in BTT the assumption of unbounded number of candidates is made for the reduction. The reduction made from the NP-complete problem 3-dimensional-matching(3DM) has some properties that allow a $\Theta_2^p - hardness$ tool of Wagner [**?**] to be used. The proof is omitted.

Θ_2^p-completeness suggests that the relevant problem is far from being efficiently solvable, and completeness for this higher level of the polynomial hierarchy speaks more powerfully than would completeness for NP.

The drawback of having a voting scheme for which the winner determination problem is computationally intractable, could not be put better than BTT in [**?**] about the Dodgson rule:

We think that Lewis Carroll[1] *would have appreciated the idea that a candidate's mandate might have expired before it was ever recognized.*

So, since the determination of the winner in the Dodgson rule is intractable, is there any interest in studying it? Yes! First of all, one of the reasons the Dodgson rule is so interesting to study for computer scientists, is that it is one of the simplest and most naturally perceived examples of a Θ_2^p-complete problem. Until Hemaspaandra et.al. proved their result, all Θ_2^p-complete problems were artificial, while the Dodgson-winner problem is very natural. As we know any complete problem of a class is indicative of its complexity (therefore we have a strong theoretical interest in this practical problem) Furthermore, we will see in the following sections that the Dodgson winner problem falls in P for special cases, i.e. when the number of candidates or voters is bounded in advance, and we will present a heuristic that claims that the Dodgson winner problem is efficiently solved "quite frequently".

3.2.2 Hardness when Candidates or Voters are Bounded

Interestingly enough, BTT note, that when we bound in advance either the number of voters or the number of candidates, the problem of Dodgson Winner falls into P. However for very large $|V|$ or $|C|$ the constants may be so big that being in P is more of a technicality than having any practical use.

When the number of voters is bounded in advance then for each instance of the problem there are only $|C|^{|V|}$ possible places of candidate c, which is a constant since $|V|$ is bounded. Each of these $|C|^{|V|}$ possible places implies a number of "switches" which is easy to compute. Therefore, we compute all these implied "switches" and the Dodgson score of c is the minimum

[1] pen name of Charles L. Dodgson

of such "switches". This procedure is technically in P but it is clearly inefficient.

When the number of candidates is bounded by a constant, we can calculate the Dodgson score in polynomial time. How? We form the problem as an integer program with a fixed number of variables. Lenstra [?] proved that Integer programming though NP-complete for the general case, is in P, when the number of variables is fixed. Let us see the ILP we are talking about.

Index by i the $|C!|$ *types* of preference orders found among the voters, and let N_i(constant) be the number of voters of type i. Let x_{ij}(variable) be the number of voters with preferences of type i for which candidate c will be moved upwards by j positions. Let e_{ijk}(constant) be 1 if the result of moving candidate c by j positions upward in a preference order of type i is that c gains an additional vote against candidate k, and 0 otherwise. Let d_k(constant) be the minimum number of votes that c must gain against k to defeat him in a pairwise election. If c already defeats k, then $d_k = 0$. Then the Dodgson score of c is the value of the following integer linear program.

$$\begin{aligned}
&\min \sum_{ij} j \cdot x_{ij} \qquad s.t. \\
&\sum_{j} x_{ij} = N_i \qquad (i \rightarrow \text{different preference order types}) \\
&\sum_{ij} e_{ijk} \cdot x_{ij} \geq \text{def(k)} \qquad (k \rightarrow \text{candidates}) \\
&x_{ij} \geq 0, \text{integer} \qquad (j \rightarrow \text{upward pushes})
\end{aligned}$$

The objective is to minimize the number of switches, i.e the cost function will give the dodgson score. The first set of constraints restricts the numbers and types of preferences to those actually present among the voters. The second set of constrains guarantees that enough "switches" occur in order to make c a condorcet winner.

The number of different types of voters (preference orders) is no greater than $|C|!$, and the number of different positions in any preference order is $|C|$. Consequently there are no more than $|C| \cdot |C|!$ variables x_{ij} and no more than $|C| + |C|!$ non-trivial constraints. So, for a constant number of candidates $|C|$, the problem is in P, although the constants that are hidden by the asymptotic notation may be huge . By running this program for every candidate($|C|$ times), we calculate the dodgson score of each candidate and then find the winner(s).

We note that only x_{ij} are variables of the program; the rest are parameters (constants) that are determined before we run the program, having available the preferences of the voters. In order to agree that this solution is indeed in P one must verify (it is not difficult) that these pre-computations can be performed in polynomial time.

3.2.3 Approximating the Dodgson score

Up to now we have said that computing the Dodgson score is generally NP-complete. We tried to find a way out of this inefficiency by loosing in terms of generality, i.e. bounding the number of candidates or voters beforehand. But even in these cases we can see that being in P is more of a technicality, than meaning efficiency. A next step to circumvent intractability is as usually approximation. Caragiannis et. al. [?] propose a slightly different formation of the problem as ILP, that yields an algorithm (using a randomized rounding technique similar to the one used for Set Cover [?]) that gives a $\Theta(\log |C|)$ approximation of the Dodgson score with high probability ($> 1/2$). The ILP they use is the same with the one given by BTT only varied to make the variables binary:

$$\begin{aligned}
&\min \sum_{ij} j \cdot x_{ij} \qquad s.t. \\
&\sum_{j} x_{ij} = 1 \qquad (i \rightarrow \text{voters}) \\
&\sum_{ij} e_{ijk} \cdot x_{ij} \geq \text{def(k)} \qquad (k \rightarrow \text{candidates}) \\
&x_{ij} \in \{0,1\} \qquad (j \rightarrow \text{upward pushes})
\end{aligned}$$

3.2.4 A heuristic for finding the Dodgson-winner

In the previous section we said that the dodgson winner can be found in polynomyal time when the number of candidates or voters is bounded. We also mentioned there is an efficient algorithm that gives a good approximation of the Dodgson score. Is there anything better that can be expected? In 2009 L. Hemaspaandra and C. Homan [?] gave a heuristic which, when the number of voters is much greater than the number of candidates - even polynomially greater, but more than quadratic - a simple greedy algorithm very “frequently” finds the Dodgson winners in such a way that it “knows” that it has found them, and furthermore, the algorithm never incorrectly declares a non-winner to be a winner. Of course what “frequently” means has to be further explained. We just want to warn the reader - so that not too high expectations are raised - that “frequently” refers to the fraction of the instances that the algorithm finds a solution and not to any kind of randomized algorithm!

In this section we give the algorithm as it was given by the authors, we explain it and present the guarantees of correctness that it gives.

For two candidates c, d we denote $c <_v d$ if in vote v d is preferred to c. If the two are also adjacent, we denote it $c \prec_v d$.

The first algorithm we present is a heuristic for the Dodgson Score decision problem which as we have already said is NP-complete. The second is for the Dodgson Winner decision problem which is Θ_2^p-complete.

One of the most interesting properties the algorithms that we present have is that they are "self-knowingly" correct. This can be defined formally but it suffices to say that an algorithm is "self-knowingly" correct, if whenever it claims it has found the correct solution and it is sure about it, the solution is indeed correct.

```
GreedyScore(C,V,c)
 1  ▷ $C$ is the set of candidates
 2  ▷ $V$ is the list of votes
 3  ▷ $c \in C$ whose dodgson score the algorithm tries to compute.
 4  for $c \in C \backslash \{c\}$
 5        do Deficit[d] $\leftarrow 0$
 6           Swaps[d] $\leftarrow 0$
 7  ▷ We just initialized the counter variables. Deficit[d] shows
 8  ▷      the number of votes by which $d$ beats $c$ in their pairwise contests.
 9  ▷      Swaps[d] counts the number of votes for which $c \prec d$
10  for each vote $v \in V$
11        do $i \leftarrow 1$
12           while $v[i] \neq c$          ▷ for $d <_v c$
13                do $d \leftarrow v[i]$
14                   Deficit[d] $\leftarrow$ Deficit[d]$-1$
15                   $i \leftarrow i+1$
16           if $i < length(v)$          ▷ $v[i] = c$
17             then $d \leftarrow v[i+1]$
18                  Swaps[d] $\leftarrow$ Swaps[d]+1
19           for $i \leftarrow i+1$ to $length(v)$          ▷ for $c <_v d$
20                do $d \leftarrow v[i]$
21                   Deficit[d] $\leftarrow$ Deficit[d]+1
22  confidence $\leftarrow$ “definitely”
23  ▷ Calculate the score of $c$
24  score $\leftarrow 0$
25  for each $d \in C \backslash \{c\}$
26        do if Deficit[d] $\geq 0$
27             then score $\leftarrow$ score + $\lfloor$Deficit[d]/2$\rfloor + 1$
28                  if Deficit[d] $\geq 2\cdot$ Swaps[d]
29                    then confidence $\leftarrow$ “maybe”
30                         score $\leftarrow$ score+1
31  return (score, confidence)
```

The part that maybe needs some further explanation is where the score of the candidate is being calculated in lines 24 - 31. If we have some positive deficit for a candidate d, then $\lfloor$Deficit[d]/2$\rfloor + 1$ swaps between c and d, if such exist, are necessary and sufficient to make c beat d(we will prove this more formally right next, for now let us take it for granted). The point is that if such swaps exist - this is the case Deficit[d]$< 2 \cdot Swaps[d]$ - our algorithm calculates the score correctly, and knows it has done so (outputs "definitely"). If however not so many swaps, i.e. votes where d is adjacent to c, exist (the case Deficit[d]$\geq 2 \cdot Swaps[d]$), the algorithm computes a score that is a lower bound of the actual dodgson score of candidate c and outputs it declaring uncertainty. Since the algorithm is a heuristic and not an approximation algorithm the error in this case may be arbitrarily large but that is not the problem, since a heuristic is no approximation algorithm to give a guaranteed maximum error.

The next algorithm decides whether c is a winner of a dodgson election by giving again a degree of certainty.

```
GreedyWinner($C, V, c$)
1   ▷ $C$ is the set of candidates
2   ▷ $V$ is the list of votes
3   ▷ $c \in C$. We wish to check whether $c$ is a Dodgson winner in election $(C, V)$.
4   (cscore,confidence) ← GreedyWinner($C, V, c$)
5   winner ← "yes"
6   for all candidates $d \in C \backslash \{c\}$
7          do (dscore,dcon) ← GreedyWinner($C, V, d$)
8             if dscore < cscore
9                then winner ← "no"
10            if dcon = "maybe"
11               then confidence ← "maybe"
12  return (winner,confidence)
```

Both algorithms run in polynomial time. The total number of line-executions in a run of *GreedyScore* is clearly $O(|V| \cdot |C|)$, and for *GreedyWinner* is, including the line executions within the subroutine calls, $O(|V| \cdot |C|^2)$.

We already said that the authors claim the heuristic is "self-knowingly" correct. What they mean by that is that, when their heuristic gives an answer accompanied by "definitely", it can be proven that the answer is indeed correct.

Theorem 4. Both *GreedyScore* and *GreedyWinner* are self-knowingly corect algorithms

Proof. Suppose that *GreedyScore*, on input (C, V, c), returns "definitely" as the second component of its output. Then, as it is clear from lines 24 - 31 *every* candidate $d \in C \backslash \{c\}$ for which Deficit[d] ≥ 0 must also have Deficit[d] $<$ 2·Swaps[d]. In this case, the algorithm sets its score to score $= \sum_{d \in C \backslash \{c\}: Deficit[d] \geq 0} \lfloor Deficit[d]/2 \rfloor + 1$. For this value of score to actually be the Dodgson score of c, we must show (a) that we can by performing "score" swaps turn c into the Condorcet winner of election (C, V) and (b) that by performing fewer than "score" swaps we cannot make c a Condorcet winner. Both claims depend on the following observation: Let v be a vote having some candidate d such that $c \prec$d. Then swapping c and d would decrease by two(!) the difference between "the number of votes where c is preferred to d" and "the number of votes where d is preferred c"(Deficit[d] of our algorithm). Also, c's standing against any candidate other than d would not be affected by this swap. So, regarding the number of swaps needed to make c beat some $d \in C \backslash \{c\}$, let $|V_d|$ be the set of votes in V in which $c \prec d$. If Deficit[d] is, by line 26 nonnegative (if Deficit[d] is negative then c already beats d and the number of swaps needed is trivially 0), and if $|V_d| \lfloor Deficit[d]/2 \rfloor + 1$, then we can make c beat d by choosing exactly $\lfloor Deficit[d]/2 \rfloor + 1$ votes in $|V_d|$ and swapping the positions of c and d in these votes. But Swaps[d] is precisely $|V_d|$ and since we are in case Deficit[d] $<$ 2·Swaps[d], Swaps[d] is at least $\lfloor Deficit[d]/2 \rfloor + 1$. Thus, by summing over all such $d \in C \backslash \{c\}$, "score" is enough swaps to make c a Condorcet winner, and we have satisfied (a). To see (b), suppose that it is possible to make c a Condorcet winner with fewer than $\sum_{d \in C \backslash \{c\}: Deficit[d] \geq 0} \lfloor Deficit[d]/2 \rfloor + 1$ swaps. Then for some $d \in C \backslash \{c\}$ such that Deficit[d] $geq0$ it must hold that at most $\lfloor Deficit[d]/2 \rfloor$ of these swaps are applied to c's standing against d. But then, since as noted above one swap changes the deficit between c and d by exactly two, we do not have enough swaps to make c beat d. So we have proven that when *GreedyScore* returns "definitely" as the second component of its output, that score is the optimal number of swaps needed to make c a Condorcet winner. Therefore it is "self-knowingly" correct.

It can then easily be argued that *GreedyWinner* correctly checks whether c is a Dodgson winner if every call it makes to GreedyScore correctly calculates the Dodgson score. GreedyWinner then returns "definitely" exactly if each call it makes to *GreedyScore* returns "definitely". Therefore it is too "self-knowingly" correct. □

What we have proven already is that the algorithm *GreedyScore* (the same holds for *GreedyWinner*) computes the dodgson score of c correctly when there are enough votes with c adjacent to each candidate it loses from, so that these swaps can make c a condorcet winner. In other words when there are enough votes of the form ... $\prec c \prec d \prec$ ..., $\forall d \in C \backslash \{c\}$ then the algorithm works fine. When however there exists even one candidate d for which most of the votes are in the form ... $\prec c \prec$ ... $\prec d \prec$ ... then the algorithm cannot answer with certainty. The thing is that Homan and Hemaspaandra prove that as the ratio of voters to candidates gets higher, the probability to get an unwanted instance goes asymptotically to 0 as the number of candidates increases, assuming that we consider all $(m!)^n$ elections with m candidates and n voters equally probable to come up. Let us see how.

The main result of Homan and Hemaspaandra is the following:

Theorem 5.

- For each (election, candidate) pair it holds that if $GreedyWinner$ outputs "definitely" as its second output component, then its first output component correctly answers the question, "Is the input candidate a Dodgson winner of the input election?"
- For each $m, n \in \mathbb{N}$, when a Dodgson election E selected uniformly at random from all Dodgson elections having m candidates and n votes (i.e., all $(m!)^n$ Dodgson elections having m candidates and n votes have the same likelihood of being selected) then the probability, that the election has one or more candidates c such that $GreedyWinner$ on input (E, c) outputs "maybe" as its second output component, is less than $2(m^2 - m)e^{\frac{-n}{8m^2}}$.

The first part of the theorem was proven in Theorem 4. The second part has a rather interesting proving procedure therefore we present it here.
We will need a theorem that gives some variant of Chernoff Bounds.

Theorem 6 (Alon & Spencer 2000)**.** Let $X_1 ... X_n$ be mutually independent random variables. If there exists some $p \in [0, 1]$ such that, for each $i \in \{1, ..., n\}$,

$$Pr(X_i = 1 - p) = p \text{ and}$$
$$Pr(X_i = -p) = 1 - p$$

then for all $a \in \mathbb{R}$ where $a > 0$ it holds that $Pr(\sum_{i=1}^{n} X_i > a) < e^{-2a^2/n}$.

Next we prove a lemma that we will need.

Lemma 7.

1. $Pr(|i \in \{1, ..., n\} : c <_{v_i} d| > \frac{2mn+n}{4m}) < e^{\frac{-n}{8m^2}}$
2. $Pr(|i \in \{1, ..., n\} : c \prec_{v_i} d| > \frac{3n}{4m}) < e^{\frac{-n}{8m^2}}$

Proof. 1. For each $i \in \{1, ..., n\}$ define X_i as

$$X_i = \begin{cases} 1/2, & if c <_{v_i} d \\ -1/2 & otherwise \end{cases}$$

Then $|i \in \{1, ..., n\} : c <_{v_i} d| > \frac{2mn+n}{4m}$ iff

$$\sum_{i=1}^{n} X_i > \tfrac{1}{2}(\tfrac{2mn+n}{4m}) - \tfrac{1}{2}(n - \tfrac{2mn+n}{4m}).$$

We first note that $\frac{1}{2}(\frac{2mn+n}{4m}) - \frac{1}{2}(n - \frac{2mn+n}{4m}) = \frac{n}{4m}$. The crucial point here is to note that $c <_{v_i} d$ holds for half the votes of all possible $m!$ a voter i may adopt. So if we let X_i follow the uniform distribution $Pr(c <_{v_i} d) = 1/2$ or equivalently $Pr(X_i = 1 - 1/2) = 1/2$, and $Pr(X_i = -1/2) = 1 - 1/2$. Therefore, setting $a = \frac{n}{4m}$ and $p = \frac{1}{2}$ in Theorem 6 we prove the first part of the lemma.

The proof of the second part is pretty much analogous.

2. For each $i \in \{1, ..., n\}$ define X_i as

$$X_i = \begin{cases} 1/m - 1, & if c \prec_{v_i} d \\ 1/m & otherwise \end{cases}$$

Then $|i \in \{1, ..., n\} : c \prec_{v_i} d| < \frac{3n}{4m}$ iff

$$\sum_{i=1}^{n} X_i > \tfrac{1}{m}(n - \tfrac{3n}{4m}) + (\tfrac{1}{m} - 1)(\tfrac{3n}{4m}).$$

We note that $\frac{1}{m}(n - \frac{3n}{4m}) + (\frac{1}{m} - 1)(\frac{3n}{4m}) = \frac{n}{4m}$. The crucial point is again to note that $c \prec_{v_i} d$ holds for exactly m of the votes of all possible $m!$ a voter i may adopt. So if we let X_i follow the uniform distribution $Pr(c \prec_{v_i} d) = 1/m$ or equivalently $Pr(X_i = 1/m - 1) = 1/m$, and $Pr(X_i = 1/m) = 1 - 1/m$. Therefore, setting $a = \frac{n}{4m}$ and $p = 1 - \frac{1}{m}$ in Theorem 6 we prove the second part of the lemma. □

We are now ready to prove the second part of the main theorem, Theorem 5. The proof is broken down in 3 sub-results.

1. Let V satisfy $|V| = n$. For each $c \in C$, if for all $d \in C \backslash \{c\}$ it holds that $|i \in \{1, ..., n\} : c <_{v_i} d| \leq \frac{2mn+n}{4m}$ and $|i \in \{1, ..., n\} : c \prec_{v_i} d| \geq \frac{3n}{4m}$ then $GreedyScore(C, V, c) = (Score(C, V, c), definitely)$.

Proof. $\frac{2mn+n}{4m} = \frac{n}{2} + \frac{n}{4m}$, so if $|i \in \{1, ..., n\} : c <_{v_i} d| \leq \frac{2mn+n}{4m}$ then either c already beats d or if not then the defection of more than $n/4m$ votes from preferring-d-to-c to preferring-c-to-d would (if such votes exist) ensure that c beats d. If now, $|i \in \{1, ..., n\} : c \prec_{v_i} d| \geq \frac{3n}{4m}$ then (excluding as invalid all cases where at least one of n or m equals zero) $|i \in \{1, ..., n\} : c \prec_{v_i} d| \geq \frac{n}{4m}$ and so *GreedyScore* will be able to make enough swaps (in fact, and this is critically important in light of the *GreedyScore* algorithm, there is a sequence of swaps such that any vote has at most one swap operation concerning c and d performed on it) so that c beats d. □

2. For each c, $d \in C$ such that $c \neq d$, $Pr(|i \in \{1, ..., n\} : c <_{v_i} d| > \frac{2mn+n}{4m} \vee |i \in \{1, ..., n\} : c \prec_{v_i} d| < \frac{3n}{4m})$. The probability is taken over drawing uniformly at random an m-candidate, n-voter Dodgson election $V = (v_1, ..., v_n)$ (i.e., all $(m!)^n$ Dodgson elections having m candidates and n voters have the same likelihood of being chosen).

Proof. This follows from applying the union bound to the above Lemma 7. □

3. This is the final result:
$Pr((\exists c \in C)[GreedyWinner(C, V, c) \neq (DodgsonWinner(C, V, c), definitely)])) < 2(m^2 - m)e^{\frac{-n}{8m^2}}$, where the probability is taken over drawing uniformly at random an m-candidate, n-voter Dodgson election $V = (v_1, ..., v_n)$.

Proof. It follows from 1. and by applying 2. and the union bound to

$$Pr(\bigvee_{c,d \in C \wedge c \neq d}((|i \in \{1, ..., n\} : c <_{v_i} d| > \tfrac{2mn+n}{4m}) \vee (|i \in \{1, ..., n\} : c \prec_{v_i} d| < \tfrac{3n}{4m}))),$$
$$\text{where } |(c, d) : c, d \in C \wedge c \neq d| = m^2 - m.$$

□

Now some observations about the heuristic we just presented. Although the guarantees provided by the heuristic are very satisfactory we must keep in mind two things. First of all the assumption that all $(m!)^n$ instances of the problem have equal probability to come up certainly requires further explanation! In real situations this may be far from the case; for example in political elections this is definitely not the case. For the analysis to be of any practical importance reasons must be given that justify the assumption. Second, although the guarantees of the heuristic may be very satisfactory theoretically, and the algorithm never proclaims a winner falsely, heuristics are by definition unreliable, in the sense that some instances are lost cases. In other words in some cases we will only get an unsure answer and this is the best the algorithm can give us; it is no randomized algorithm which if we run many times we increase the probability of a correct answer. We can imagine that in a political election for example, this may be a problem.

3.3 Manipulation & Bribery

We have already talked about strategic voting and manipulable voting schemes. Here we will treat all this in a more formal way. When talking of **manipulation**, we suppose that among the voters exists a *manipulative group*, a subset of voters who strategically change their preference lists in order to influence the election's outcome in their favour. Let us see an example of what exactly *manipulation* means in practice.

Our voting system is the *plurality rule*. We have three candidates, a, b, and c, and eleven voters, whose preferences are the following:
5 voters: $a > b > c$
4 voters: $b > a > c$
2 voter : $c > b > a$.

Under the plurality rule, candidate a is proclaimed winner since he receives the most votes; 5, against 4 and 2 that candidates b and c receive respectively. However the 2 voters whao have c as they first choice may look for a compromise: since they see that given the votes of the rest of the voters their preferred candidate will not win, and if they vote truthfully their worst option - a - will be elected, they may be tempted to report $b > c > a$ as their preference order and thus have candidate b win which is their second to best choice.

It is interesting to point out that such way of voting captures the problem of "lost vote" to small parties we see in the political elections. Many voters realize that their small party has no chance to be elected in the government and compromise by voting for one of the two leading parties. In other words it seems that the plurality rule leads to a two-leading-parties political setting, something verified by our experience.

There are two main variations of manipulation that are considered in the literature of voting rules; *constructive*, where the objective is to make a distinguished candidate one of the (possibly many) winners, and *destructive* where the objective is to preclude a distinguished candidate from winning. In this section the attention is focused on the constructive scenario, because this is the pervasive direction of research (to our knowledge only Conitzer, Sandholm and Lang in [**?**] have studied the destructive case). From now on when the type of manipulation is not mentioned explicitly, the constructive case is implied.

The problem of (constructive) manipulation is formulated as a decision problem in the following way:

Name: E-manipulation
Input: A set C of candidates, a set N of non-manipulative voters, a set M of manipulative voters with $N \bigcap M = \varnothing$ ($N \bigcup M = V$), and a distinguished candidate $c \in C$.
Question: Is there a way to set the preference lists of the voters in M such that, under election system E, c is a winner of the election?

A variation of the above problem is to also have different weights for voters both in N and M; this is the E-weighted-manipulation problem. We note that although in the beginning, for example in the work of Bartholdi et.al, weights weren't taken in consideration, most later work, for example Hemaspaandra et. al., Conitzer et. al., incorporated weights. The reason is that (1) without weights manipulation is easy in most cases, as we shall see in section 3.3.3, and (2) because, as we have already mentioned, in many real-world elections, voters are in fact weighted.

Bribery is a very similar setting to manipulation, only that in this case the manipulative group is not known beforehand; in algorithmic terms, the manipulative group is not part of the input. We suppose that an outside agent, who has a finite budget, wants to determine which voters to bribe, so that they will change their preferences, in order to influence the outcome in a desired way. In manipulation the manipulative group is fixed; in bribery it is to be found.

The problem of bribery is formulated as a decision problem in the following way:

Name: E**-bribery**
Input: A set C of candidates, a set V of voters, a distinguished candidate $c \in C$, and a nonnegative integer k.
Question: Is it possible to change the preference lists of at most k voters such that, under election system E, c is a winner of the election?

An intuitive way to think of k is to think that it is the maximum number of voters that the manipulator can bribe with his fixed budget.

E-bribery has also some variations. In E-weighted-bribery each voter has a weight, in E-\$-bribery each voter has a price tag, so here k is the budget in monetary units. If the voters have both price tags and weights then we have the case of E-weighted-\$-bribery.

3.3.1 Plurality- Manipulation and Bribery in various Settings

Let us again begin with the plurality rule which is probably the most common voting rule. First we recall that plurality-rule election with m voters is described by the scoring vector $(1, \overbrace{0, ..., 0}^{m-1})$. BTT in [?] state the following theorem:

Theorem 8. *Plurality-manipulation and plurality-weighted-manipulation are both in P.*

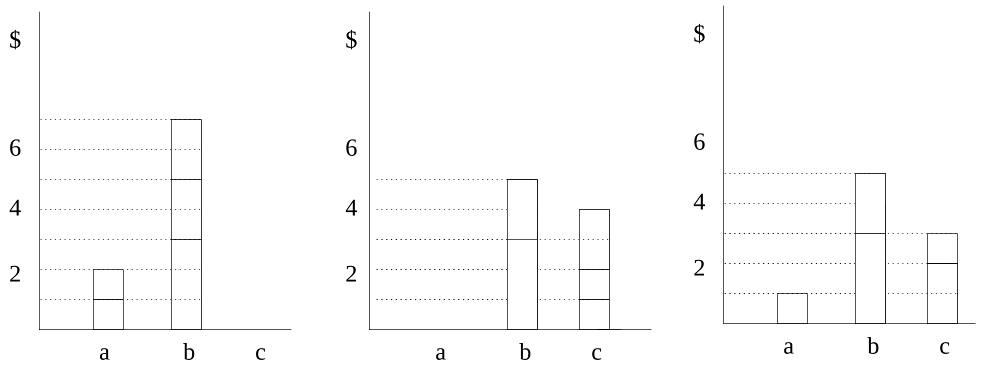
The proof is quite straightforward: all the manipulators vote for their desired candidate c, and c is either proclaimed a winner, or the manipulators can not influence the result.

BTT weren't concerned with the bribery problem but Faliszewski, Hemaspaandra and Hemaspaandra were and in [?] they give the following theorem:

Theorem 9. Plurality-bribery is in P.

Proof. A (polynomial time) greedy algorithm works for plurality-bribery: we keep bribing the voters of a current winner to vote for c, until c is a winner or we have already bribed k voters (we have run out of budget). □

However this simple and natural approach doesn't work when we make the setting slightly more complex, i.e for either the plurality-weighted-bribery or the plurality-\$-bribery. In the \$ (weighted) case it is not always clear whether one should first bribe the cheapest (heaviest) voter of some current winner or just the globally cheapest (heaviest) voter who does not yet vote for c. In the latter case we get an extra vote for c, with the minimum cost (we get the greatest additional vote weight for c) but in the former case we gain a vote (some vote weight) for c while simultaneously potentially decreasing the total votes (vote weight) that c needs in order to become a winner. (We say "potentially decrease" since if there are multiple winners then the total votes c needs to win won't change immediately. But if we keep on bribing the voters of current winners, this decrease will occur eventually.) Let us consider the following example which demonstrates why the simple greedy approach fails for plurality-\$-bribery. We have candidates a, b, and c and five voters with the following prices: 2 of them 1\$, 2 of them 2\$, and 1 3\$. Both voters with price 1\$ have a as their top candidate, and all the others have b as their top candidate. Thus a receives 2 votes, c receives no votes, and b is the winner with 3 votes (see Figure). If we proceed greedily, first bribing the two (globally) cheapest voters - voters preferring a - then c still loses to b, with 3 to 2 votes. In order for c to win one voter of b must be bribed and the total cost will then be 4\$. Can we do better? Yes. If we bribe one 1\$ voter preferring a and one 2\$ voter preferring b then c wins (in tie with b, but still) having spent 3\$, and actually this is the best we can do.



(a) Initial State (b)Greedy bribery of cost 4\$ (c) Optimal bribery of cost 3\$

We have therefore established that plurality-\$-bribery requires “more” than a simple greedy algorithm. The same is true for plurality-weighted-bribery. How much “more”? Not much: Faliszewski, Hemaspaandra, and Hemaspaandra [**?**] (from now on FHH) obtained polynomial-time algorithms for both plurality-weighted-bribery and plurality-\$-bribery proving the following theorem:

Theorem 10. Plurality-weighted-bribery and plurality-\$-bribery are both in P.

Proof. We give the proof-idea for the case of plurality-\$-bribery, which is quite interesting, avoiding the special details. The idea is very similar for Plurality-weighted-bribery.

Assume that c will be capable of getting at least r votes , where r is some number to be specified later. If this is to make c a winner, we need to make sure that everyone else gets at most r votes. The algorithm is the following:

1) We choose greedily the cheapest voters of candidates that defeat c and bribe those voters to vote for c until those candidates have no more than r votes.
2) We make sure that c gets at least r votes by bribing the cheapest of the remaining voters.
3) If during this process c ever becomes a winner without exceeding the budget then we know that bribery is possible. Otherwise not.
The question is how do we pick the value of r. Simple: we don't! We just run this procedure for all $|V|$ possible values, and accept if it succeeds for at least one of them. □

Let us see how our algorithm works for our previous example, with a budget of 3\$.

r=1
1) we bribe one voter of b of price 2\$. Then we try to bribe the other one too but run out of budget.

r=2
1) we bribe one voter of candidate b of price 2\$.
2) we bribe one voter of candidate a of price 1\$.
3) c is a winner!

Now, although both problems are in P, when we combine them to make the setting a bit more complicated we jump to NP-completeness! In other words, neither prices, nor weights alone make the problem hard enough. Only when they are combined, the problem becomes NP-hard.

Theorem 11. Plurality-weighted-\$-bribery is NP-complete (even for two candidates)

Proof. The proof is as usually based on reduction. The reduction is from the known to be NP-complete problem *PARTITION*

Name:PARTITION
Input:A set of n non-negative integers $s_1, ..., s_n$ summing to $2M$
Question: Is there a subset of these integers that sums to M?

First we prove that plurality-weighted-\$bribery is in NP: Given (suppose we can guess one) a correct subset of the voters to bribe and test in polynomial time whether such a bribe both makes our designated candidate a winner and does not exceed the budget. It remains to show that the problem is NP-hard. To show NP-hardness, we will construct a reduction from PARTITION. Let $s_1, ..., s_n$ be a set of nonnegative integers and let $\sum_{i=1}^{n} s_i = 2M$. Our goal is to design an election $E = (C, V, c, k)$ in which c can become a winner by bribery of cost at most k *if and only if* there is a set $A \subseteq 1, ..., n$ such that $\sum_{i \in A} s_i = M$. We define the election to have two candidates, c and c', and exactly n voters, $v_1, ..., v_n$, with each v_i having both weight and price equal to s_i. All voters originally prefer c' to c. The budget k is set to M. The claim is that c can become a winner if and only if $s_1, ..., s_n$ can be partitioned into two equal-sum groups. Let us assume that there is a set $A \subseteq 1, ..., n$ such that $\sum_{i \in A} s_i = M$. This means that for each $i \in A$ we can bribe v_i to vote for c and get for c a total vote weight of M. This makes c a winner. On the other hand, assume that c can be made a winner by bribes of total cost at most $k = M$. The weight of each voter is equal to his or her price and so c can obtain at most vote weight $k = M$. In fact, c must obtain exactly vote weight M, since from our setup it is clear that if c gains strictly less than vote weight M then c' will be the unique winner. This means that there is a way of picking some voters whose weights sum up to exactly M, and thus the set $s_1, ..., s_n$ can be partitioned into two subsets that each sum up to M. The reduction can be carried out in polynomial time and so the proof is complete. □

We know that NP-hardness is a worse-case analysis of the problem. This means that in the worse case it is hard to solve the problem. This of course means that there may still be instances that the problem is easy to be solved ($\in P$). FHH proved that we know in advance some instances of the plurality-weighted-\$-bribery problem for which it is easy to manipulate plurality voting.

We mentioned in the above paragraphs that for the plurality election system we need to *accumulate enough complexity* in our setting in order to make manipulation computationally hard ($\in NP$). More precisely, by removing either price-tags or weights of voters we jump back to P. FHH make it even more interesting by proving in [**?**] that, when either the prices or the weights are represented in unary, the respective problem jumps back in P. Informally put, if either the weights or the prices are *fairly small numbers* the problem slips into P.

The natural question that rises is why does the unary encoding of the prices or the weights even matter? The reasoning is quite analogous to the known *discrete Knapsack problem.* When for example weights (the same for prices) are encoded in unary, then there are only linearly many different total weights of subsets of voters; this allows some dynamic programming technique to be applied.

FHH [**?**] investigate also another interesting variation of the bribery setting, *negative-bribery.* They note that the setting of bribery considered so far, where people are bribed to vote for c, is quite natural but it also may have real-world downsides: the more people are bribed, the more likely it may be that the malicious attempts will be detected and will work against c. To minimize the chances of that happening we might instead bribe voters to vote not for c but for some other candidate(s), who already lose against c. This way c does not get extra votes but might be able to take away enough from the most popular candidates to become a winner. In this context they prove the following theorem:

Theorem 12. Plurality-negative-\$-bribery is in P, but plurality-weighted-negative-bribery is $NP-complete$.

Let us first see how plurality-negative-\$-bribery is in P and then what makes plurality-weighted-negative-bribery more difficult.

Proof. FHH give in [**?**] a polynomial-time algorithm for plurality-negative-\$bribery. Suppose we have a voting instance, where we want to make candidate c win and have a budget of k (say in \$) to use for bribes . We need to make c a winner by taking votes away from popular candidates and distributing them among the less popular ones. (Note that as we are doing in the whole context of bribery and manipulation we say “a winner”, i.e. we are addressing the non-unique case. However, it is clear that a similar approach works for the unique case). We partition the set of all voters into three sets: candidates that defeat c, from whom votes need to be taken away, candidates that are defeated by c, to whom we can give extra votes, and candidates that have the same score as c. We note as score(c) the number of votes candidate c has received.

$$C_{above} = \{c' | c' \in C, score(c') > score(c)\}$$
$$C_{below} = \{c' | c' \in C, score(c') < score(c)\}$$
$$C_{equal} = \{c' | c' \in C, score(c') = score(c)\}$$

Since all candidates have the same weight (weight 1) in plurality-negative-\$bribery, it is not hard to argue that in order to have a successful negative bribery, there is no reason to bribe voters into or out of C_{equal} or to bribe voters to move within their own "group", e.g., bribing a voter to shift from one C_{below} candidate to another. To make sure that c becomes a winner, for each candidate $c' \in C_{above}$ we need to bribe as many of her voters as are needed to reduce her score to at most $score(c)$. Thus, altogether, we need to bribe $\sum_{c' \in C_{above}} (score(c') - score(c))$ voters. The number of votes that a candidate $c'' \in C_{below}$ can accept without preventing c from winning is $\sum_{c'' \in C_{below}} (score(c) - score(c''))$. Thus, it is not hard to see that the necessary condition for a negative bribery to be possible is

$$\sum_{c' \in C_{above}} (score(c') - score(c)) \leq \sum_{c'' \in C_{below}} (score(c) - score(c''))$$

If the above inequality does not hold then we immediately reject. Otherwise, it remains to check whether the cost of our negative bribery is within our budget: For every candidate c$' \in C_{above}$ let $p_{c'}$ be the price of bribing all of her $score(c') - score(c)$ cheapest voters. If it holds that $\sum_{c' \in C_{above}} p_{c'} \leq k$ then we accept, as the negative bribery is possible. Otherwise we reject.
Clearly, the algorithm works in polynomial time. The correctness follows from the fact that we need to make all candidates in C_{above} have score at most $score(c)$ and for each $c' \in C_{above}$, $p_{c'}$ is the lowest possible cost of achieving that. The inequality we gave guarantees that the votes taken from candidates in C_{above} can be re-distributed among those in C_{below} without preventing c from winning.
Hence plurality-negative-\$-bribery is in P □

What makes though plurality-negative-weighted-bribery more difficult? In plurality-negative-weighted-bribery, because of the different weights of the voters, we cannot assume that the only re-distributions that make sense are between candidates of C_{above} and C_{below}. To see this let us consider the following case: candidate Big has $score_{Big} = 12$, having the votes of one weight-10 voter and one weight-2 voter; candidate c has $score_c = 10$ with the vote of one weight-10 voter; candidate Eq has $score_{Eq} = 10$ with the votes of ten weight-1 voters; finally candidates $Small$, $Small2$ have $score_{Small} = score_{Small2} = 9$ with the vote of one weight-9 voter each. The limit on the number of bribes is 3. $C_{above} = \{Big\}$, $C_{equal} = \{c, Eq\}$, and $C_{below} = \{Small, Small2\}$. The only way to have a successful negative bribery is by moving from Big to Eq the weight-2 voter, and then by moving one weight-1 voter to each of Small and Small2 from Eq. More intuitively, the candidates in the C_{equal} set may have to be used as intermediates for larger weights to be "broken down" to smaller ones. The proof given by FHH is a reduction from PARTITION but we omit it.

We have already said that manipulation and bribery are closely related problems, so it would be interesting to have some formal connection between them. Indeed FHH prove that each manipulation problem is polynomially reducible to its analogous \$-bribery problem. More formally put, we have the following theorem:

Theorem 13. Let M be some manipulation problem and let B be the analogous \$-bribery problem (for the same election system). It holds that $M \leq^p B$.

This result is quite notable in the sense that it states that the \$-bribery problem complexity is an upper bound for the respective manipulation problem, or that the \$-bribery problem is at least as difficult as the respective manipulation problem. This is not the case generally for the (simple) bribery problem and actually in [**?**] it is proven that there exists a voting rule for which manipulation is in NP, while bribery is in P.

3.3.2 Bounds on the number of Candidates that Guarantee Hardness

BTT gave a new direction of research for voting rules, i.e. exploring them from a computational complexity point of view. In these first results on computational hardness of manipulation, the reductions were based on the assumption that the number of candidates can be arbitrarily large. Although this can in fact make sense in some settings (when candidates are web pages on the internet for example), in many settings with most notable the political elections setting, this assumption looses relevance with reality, since the number of candidates is usually a small number. Hence, if algorithms that are exponential only in the number of candidates (and only slightly so) exist, they may be practical in such cases. In order to tackle this problem Conitzer, Sandholm, and Lang (CSL from now on for simplicity) in [**?**] proved their intractability results even for a small number of candidates. The two following results are found in this paper.

Theorem 14. Manipulation is in P for the Copeland rule, when the number of candidates is ≤ 3.

The proof (which we omit) proceeds by showing that for each case of the possible relations of weights of manipulators and non-manipulators there is a simple way for the manipulators to determine how to make their desired candidate win by voting all identically or to conclude that no such manipulation exists.

Theorem 15. For the Copeland protocol, manipulation is NP-complete when the number of candidates is ≥ 4

Proof. Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following constructive-manipulation instance. There are 4 candidates, a, b, c and p. In the non-manipulators set there are $2S+2$ voters voting (p,a,b,c), $2S+2$ voting (c,p,b,a), $S+1$ voting (a,b,c,p), and $S+1$ voting (b,a,c,p). In the manipulators, for every s_i there is a voter of weight s_i .We show the instances are equivalent. First, every pairwise election is already determined without the manipulators , except for the one between a and b. p defeats a and b; a and b each defeat c; c defeats p. Particularly, $C(p,a) = 2S+2 = C(p,b)$, $C(p,c) = -2S-2$, $C(a,c) = 2S+2 = C(b,c)$, $C(a,b) = 0$. Therefore, $score(p) = +1$, $score(a) = score(b) = 0$, $score(c) = -1$. If there is a winner in the pairwise election between a and b, after the votes of the manipulators, that winner will tie with p. So p wins the Copeland election, if and only if a and b tie in their pairwise election. But, after the votes of the non-manipulators, a and b are tied. Thus, votes of the manipulators maintain this tie if and only if the combined weight of manipulator votes preferring a to b is the same as the combined weight of votes preferring b to a. This can happen if and only if there is a partition. □

3.3.3 Three Dichotomy Results on Manipulation & Bribery

A rather interesting direction of research has been proposed by Faliszewski, E. Hemaspaandra and L. Hemaspaandra. That is, to manage to classify voting rules in families with similar characteristics; they refer to such results as "dichotomy results". The explicit reference of the importance of having such results was first made by FHH in [**?**]. In [**?**] they put it very nicely: "The ultimate goal of the project is to move from simply analysing individual election systems to finding the source of the complexity of elections". In other words the ultimate goal is to find a simple rule that tells which election systems are computationally simple and which are computationally hard with respect to whichever one of the core questions: winner determination, manipulation, bribery, and control.

We present the following dichotomy results:

Theorem 16 (Dichotomy Result 1)**.** For each voting protocol, for which winner-determination is in P, *a-manipulation* is in P.

Proof. When there are no weights, for each of the $m!$ orderings there are between 0 and n manipulators who can choose this ordering. By evaluating all of the possibilities, which are $O((n+1)^{m!})$, we can answer the manipulation problem. □

The key to this approach is that the voters are un-weighted, so the only thing that matters is the number of voters who choose each preference. Of course as we can see in the proof, this theorem holds assuming the number of candidates is bounded by a constant.

As a corollary of the above we have that:

Corollary 1. For each scoring vector a, *a-manipulation* is in P.

Theorem 17 (Dichotomy Result 2)**.** Let $a = (a_1, ..., a_m)$ be a scoring vector. If $a_2 = ... = a_m$ then *a-weighted-manipulation* is in P. In all other cases, this problem is NP-complete.[2]

It is interesting to note that the above theorem gives the plurality rule a special place among scoring protocols, being the only one for which the a-weighted-manipulation problem is efficiently solved.

Corollary 2. Let $a = (a_1, ..., a_m)$ be a scoring vector. If $a_1 = ... = a_m$ then a-weighted-$bribery is in P. In all other cases, this problem is NP-complete. [3]

The above corollary is a good example of something broadly known about reductions. When problem A reduces polynomially to a problem B, namely $A \leq^p B$, this means that B is *at least* as hard as A, i.e. it may be of the same or greater hardness. Here for example we have theorem 13 and at the same time a-weighted manipulation is in P for the case of $a_2 = ... = a_m$, while a-weighted-$bribery is in NP for the same case. This is not at all a contradiction.

Theorem 18 (Dichotomy Result 3)**.** Let $a = (a_1, ..., a_m)$ be a scoring vector. If $a_2 = ... = a_m$ then *a-weighted-bribery* is in P. In all other cases, this problem is NP-complete.

The above theorem again gives the plurality rule a special place among scoring protocols, being the only one for which the a-weighted-bribery problem is efficiently solved. It is interestin that as FHH note, this theorem replaces the word "manipulation" with the word "bribery" in theorem 17. However its proof is a far from a trivial transformation.

[2]To be precise this Theorem was independently found in all three of [?], [?], and [?], with slight differences in wording or generality.

[3]This corollary is proven combining theorems 11, 13, and 17

3.3.4 Game Theoretic Considerations of Manipulation

We end the section on manipulation & bribery by mentioning some extremely interesting observations made in [**?**]. We have defined the problem of E-(constractive)-manipulation which roughly is: given a candidate c, can we make him/her a winner? Conitzer, Sandholm, and Lang however note that, what is considered to be a successful manipulation, is not uniquely defined. For example, one may consider a manipulation successful if it causes some candidate to win that is preferred (gives higher utility) by each one of the manipulators to the candidate who would win if the manipulators voted truthfully. Another may consider a manipulation successful if it causes some candidate to win that gives a higher *sum* of utilities to the manipulators than the candidate who would win if the manipulators voted truthfully and so on. So does it make any sense that we defined our decision problem in such a way?

Conitzer, Sandholm, and Lang answer affirmatively! E-manipulation can be seen game-theoretically as the case, where a candidate, say c, gives utility 1 to each of the manipulators, while every other candidate gives them a utility of 0. This way, *no matter which notion of manipulation we adopt*, the only sensible goal for the manipulators game-theoretically is what we have examined: to make c win.

Having said that, Conitzer, Sandholm and Lang note the following:

- Hardness results for our manipulation decision problem imply hardness of manipulation under any game theoretic notion of manipulation, since E-manipulation is a special case of every manipulation notion we may adopt[4].

- Easiness results for E-manipulation transfer as well to any game-theoretic notion of manipulation we may adopt. We can solve the problem of successful manipulation by simply determining, for each candidate in a subset of possible winners, whether that candidate can be made to win, and whether his winning is a successful manipulation. So the complexity exceeds that of E-manipulation by at most a constant factor.

In essence, what they claim is that, as with winner determination, the way we defined our manipulation decision problem captures the complexity of the whole manipulation problem. By "capture" we mean exactly that the easiness and hardness results of the decision problem hold for the manipulation problem no matter what notion of manipulation we adopt.

[4]Actually the same is true for the destructive manipulation decision problem. It can be seen game-theoretically as the case, where a candidate, say c, gives utility 0 to each of the manipulators, while every other candidate gives them a utility of 1. This way, the only sensible goal for the manipulators game-theoretically is to preclude c from winning. Hence, E-destructive-manipulation is a special case of every manipulation notion we may adopt

Another very interesting note they make which may come quite as a surprise is that destructive manipulation is related to some extent with the winner determination problem! Particularly, it is related to the complexity of determining whether enough votes have been elicited to determine the outcome of the election; enough votes have been elicited if we can find that there is no way to make the conjectured winner not win by casting the yet unknown votes.

3.4 Control

When[5] an advertisement for candidate c appears on television, it may be simultaneously trying to get voters who most favour d to switch to c, and to get people who already most prefer c but weren't planning on voting to make the effort to go and vote. The first case can be described as a real-world instance of the manipulation problem we discussed earlier. What about the second case? Well, it can be seen as an instance of a somewhat different setting, named *control* of the voting system. The advertisement tries to *add voters* to the election

In abstract terms manipulation and bribery are both cases where only the voters' preferences change; the *structural properties* of the election are not changed. Structural changes, refer to such actions as *adding candidates*, *deleting candidates*, *adding voters*, *deleting voters*, *partitioning candidates*, and *partitioning voters*. The term **Control** is used to describe issues related to influencing an election's outcome by exploiting its structure.

It is rather intuitive to perceive the setting of control in the following way: suppose a chairperson supervises the election and is responsible to guarantee that the procedure is followed. The problem of *control* is whether he/she puts in practise the rules of the election in such a way that a more desirable outcome is given. For example the voting protocol may instruct that the candidates are first divided in two groups and a final vote is taken after a first elimination procedure. The chair may find such a division so that a preferable candidate is proclaimed winner in the end. Most importantly the chair will have a preferable outcome doing a legal move! In manipulation or bribery the manipulators perform an "illegitimate" move by trying to change the way they vote, by acting contrary to what the voting procedure instructs. In control the setting is different because the chair may be able to "manipulate" the procedure to his/her interest and still doing nothing "illegal", just by exploiting the structural defects of the voting protocol, just by exploiting a procedural matter.

The question put forward in the setting of *control* is whether the structure of the voting procedure allows, makes it difficult or makes it impossible for the chair to exploit it in order to have a preferable outcome.

[5]The example is given by FHH in [**?**]

Before we move forward, we note that research in the setting of *control*, focuses in the case of the unique winner, contrary to the setting of manipulation & bribery.

As with manipulation and bribery there are two variations of control; the consructive and the destructive case. There is only some slight difference here. The constructive case is whether a candidate can be made the *unique* winner, while the destructive case is whether he can be either a non-winner or a non-unique winner. Especially this last case needs some attention because it is maybe counter-intuitive: being "just" one of the winners is not considered a desired outcome here. Most results for the constructive case were presented by Bartholdi, Tovey, Trick(BTT) in [**?**] in the beginning of 1990's, while for the destructive case, by HHR in [**?**] much later in 2005.

It is important to note that we assume that the chair knows beforehand the (true) votes that the voters will cast. This is a "worst case assumption" (or "best case assumption for the chair") which means that even under this favourable setting, the chair may still find it computationally hard to pull a successful *control*. Such an assumption strengthens our hardness results but makes weaker our easiness results. This is fine, since our main goal is to prove the hardness to control a voting system and not the easiness. (we had made the same assumption for the case of manipulation).

The voting procedures examined are Plurality rule, Condorcet rule and Approval rule. There are a number of results proven in [**?**] or [**?**]. We do not think there is much point to simply reproduce all the proofs here or even state the results one after the other, so the approach we follow is the following: we define all the problems as decision problems and we discuss mainly what is the real-world incentive behind the problems and the implications the results may have. We will give some of the proofs but mainly we focus on the results per se, which we will see in a condensed table form towards the end of the section.

Definition 1. A voting system is said to be *immune* to control in a given model of control if the model regards constructive control and it is never possible for the chair by using his/her allowed model of control to change a given candidate from being not a unique winner to being the unique winner, or the model regards destructive control and it is never possible for the chair by using his/her allowed model of control to change a given candidate from being the unique winner to not being a unique winner. This means that the voting rule has some structural property that resists to that type of control. If a system is not immune to a type of control, it is said to be *susceptible* to that type of control. A voting system is said to be (computationally) *vulnerable* to control if it is susceptible to control and the corresponding language problem is computationally easy (i.e., solvable in polynomial time).[6] A voting system is said to be *resistant* to control if it is susceptible to control but the corresponding

[6]HHR examine also the case of a *certifiably-vulnerable* system, when one can even produce in polynomial

language problem is computationally hard (i.e., NP-complete)

BTT didn't study what happens when ties occur during the (possible) subelections (before the final election) when partitioning candidates or voters. HHR however studied two rather natural tie-handling procedures ties-eliminate(TE) and ties-promote(TP)

Definition 2. Ties-eliminate(TE) means that, all candidates who tie in the subelection are eliminated from the final election. **Ties-promote(TP)** means that all candidates who tie for winner move forward (TP).

Note that these models do not apply to Condorcet voting, under which when a winner exists s/he is unique; so the TE/TP distinction is made only for plurality and approval voting.

Name: Control by adding Candidates
Input:A set C of qualified candidates and a distinguished candidate $c \in C$, a set B of possible spoiler candidates, and a set V of voters with preferences over $C \bigcup B$.
Question(constructive): Is there a choice of candidates from B whose entry into the election would assure c is the unique winner?
Question(destructive): Is there a choice of candidates from B whose entry into the election would assure that c is not the unique candidate?

Name: Control by deleting Candidates
Input: A set C of candidates, a distinguished candidate $c \in C$, a set V of voters, and a positive integer $K \leq\mid C \mid$.
Question:(constructive) Are there K or fewer candidates (except for c)whose disqualification would assure the (unique) election of c?
Question:(destructive) Are there K or fewer candidates (except for c)whose disqualification would assure that c is not the unique winner?

The problem of control by adding or deleting candidates has a very clear real-world incentive. For example in national elections, there are many cases in which small parties decide to run as candidates for power or not short before the elections. If control by adding or deleting candidates is an easy problem, such small parties may decide (be instructed) to run as voters or not depending on whether this serves the goals of the controller, that may be another party to be proclaimed a winner. A, concrete real-world example is during the last

time the actual action of the chair to execute control in the "best" way (namely, by adding or deleting the smallest number of candidates or voters for add/delete problems.

presidential elections in USA, where Hillary Clinton “was deleted” as a candidate.

Name: Control by Partitioning Candidates
Input: A set C of candidates, a distinguished candidate $c \in C$, and a set V of voters.
Question:(constructive) Is there a partition of the candidates into C_1, C_2 ($C_1 \bigcup C_2 = C$ and $C_1 \bigcap C_2 = \varnothing$) so that c is the unique winner in sequential elections? (That is, $c = E(V, E(V, C1) \bigcup C2)$.[7])
Question:(destructive) Is there a partition of the candidates into C_1, C_2 ($C_1 \bigcup C_2 = C$ and $C_1 \bigcap C_2 = \varnothing$) so that c is not the unique winner in sequential elections?

Name: Control by Run-off partitioning Candidates
Input: A set C of candidates, a distinguished candidate $c \in C$, and a set V of voters.
Question:(constructive) Is there a partition of the candidates into C_1, C_2 ($C_1 \bigcup C_2 = C$ and $C_1 \bigcap C_2 = \varnothing$) so that $c = E(V, E(V, C_1) \bigcup E(V, C_2)$.($c$ is the unique winner of the election in which those candidates surviving subelections $(V, C_1), (V, C_2)$ have a *run-off* election between them over voter set V).
Question:(destructive) Is there a partition of the candidates into C_1, C_2 ($C_1 \bigcup C_2 = C$ and $C_1 \bigcap C_2 = \varnothing$) so that c is not the unique winner of the election in which those candidates surviving subelections $(V, C_1), (V, C_2)$ have a *run-off* election between them over voter set V.

The cases of partitioning Candidates are maybe the only scenarios we see that do not seem to have a very clear real world incentive. It is interesting to note that *constructive control by partition of candidates*[8] is equivalent to the path-independence criterion we discussed in section 2.2. This means that a voting rule is immune to constructive control via Candidate partition iff it complies with the path-independence criterion. Indeed the only protocol immune to this type of control is the approval protocol in the TP model.

Now let us turn our attention to the cases of control where the chair makes moves concerning the voters, not the candidates.

Name: Control by adding Voters
Input: A set C of candidates and a distinguished candidate $c \in C$; a set V of registered voters and an additional set V' of voters who are unregistered but could still register in time for the election; a positive integer $K \leq \mid V' \mid$.
Question:(constructive) Are there K voters from V' whose registration would assure that

[7]We note as $E(V, C)$ an election under some voting rule, with a set of voters V and a set of candidates C
[8]under the TP model

c is the unique winner?
Question:(destructive) Are there K voters from V' whose registration would assure that c is not the unique winner?

Name: Control by deleting Voters
Input: A set C of candidates and a distinguished candidate $c \in C$; a set V of registered voters; a positive integer $K \leq \mid V' \mid$.
Question: Are there K or fewer voters whose disenfranchisement would assure that c is the unique winner?
Question: Are there K or fewer voters whose disenfranchisement would assure that c is not the unique winner?

The case of control via adding/deleting voters has a clear real world incentive. The laws about who has the right to vote are changeable. So if the party with the majority of members in the parliament believes it can make a special group to vote for it, for example a minority, it may pass a law that gives those people the right to vote, i.e. add candidates to the procedure.

Name: Control by partitioning Voters
Input: A set C of candidates and a distinguished candidate $c \in C$; a set V of voters.
Question:(constructive) Is there a partition of the voters into V_1, V_2 ($V_1 \bigcup V_2 = V$ and $V_1 \bigcap V_2 = \varnothing$)so that $c = E(V, E(V1, C) \bigcup E(V2, C))$ (hierarchical elections assure the unique victory of c)?
Question:(destructive) Is there a partition of the voters into V_1, V_2 ($V_1 \bigcup V_2 = V$ and $V_1 \bigcap V_2 = \varnothing$)so that hierarchical elections assure the non-unique victory of c?

Again the case of control via partitioning voters is very commonly seen in the real world. Dividing the voters in voting peripheries is such a case. The dominant party may try to redistribute the voters in voting peripheries. Its goal may be to make sure that (for example based on historical data of the voters of each place) it has enough supporters to hold the seat of as many peripheries as possible, but not so many supporters as to “waste” their votes by winning with too much support.

We summarize the results related to control in the following table, provided by HHR in order to demonstrate that there is no voting rule that resists to all types of control!

	Plurality		**Condorcet**		**Approval**	
control by	**construct.**	**destruct.**	**construct.**	**destruct.**	**construct.**	**destruct.**
adding-candidates	R	R	I	V	I	V
deleting-candidates	R	R	V	I	V	I
partition-of-candidates	TE: R TP: R	TE: R TP: R	V	I	TE: V TP: I	TE: I TP: I
run-off-partition-of-candidates	TE: R TP: R	TE: R TP: R	V	I	TE: V TP: I	TE: I TP: I
adding-voters	V	V	R	V	R	V
deleting-voters	V	V	R	V	R	V
partition-of-voters	TE: V TP: R	TE: V TP: R	R	V	TE: R TP: R	TE: V TP: V

Key: I = immune, R = Resistant, V = vulnerable, TE = ties eliminate, TP = ties-promote

Resistance results are proven as usually by some reduction from a known NP-complete problem. Vulnerability results by giving an algorithm. We discuss here only the immunity results since we have not seen such result before.

Immunity of a voting system against some type of control means it has some structural/inherent property that makes it impossible to exploit it. Immunity results stem from the mathematical properties of the Social Choice Function, it is therefore only natural that immunity results are closely related to the rationality criteria we gave in section 2.2.

It stems directly from the definition of (unique) WARP that each voting rule that satisfies it, is immune to constructive control by adding candidates. It is self-proving to claim that each voting procedure that satisfies (unique) WARP is also immune to destructive control by deleting, partitioning, or partitioning with run-off candidates. Furthermore, path-independence is equivalent to constructive control by partition (and very closely related to constructive control by partition and run-off) under the TP rule. That is how we get all of our immunity results of the above table.

Until now we have presented results that study the complexity of manipulation, bribery and control of various existing voting protocols. It is apparent that no known protocol exists that resists to all cases of manipulation, bribery, and control. Even protocols that seem to resist to some type(s) may be vulnerable to some other type(s). So the natural “next step” to take in research is whether there is something we can do to close these loopholes that every protocol has. In the next two sections we present results in this very interesting direction: *how to make existing voting protocols hard to manipulate or control.*

3.5 Universal voting protocol tweaks to make manipulation hard

This section is named after the paper of Conitzer and Sandholm [**?**]. In this paper the authors present a uniform method to modify existing voting protocols in order to make (constructive) manipulation of the modified protocol computationally hard. This modification is simply to add a pre-round. The way the pre-round is organised yields different complexities of the manipulation problem.

Given a protocol E, the new protocol obtained by adding a pre-round to it proceeds as follows:

- The candidates are paired. If there is an odd number of candidates, one candidate gets a "bye" (= an immediate ticket to the next round).
- In each pairing of two candidates, the candidate losing the pairwise election(in terms of majority) between the two is eliminated. A candidate with a "bye" is never eliminated.
- On the remaining candidates, E is executed to produce a winner. For this, the implicit votes over the remaining candidates are used. (For example, if a voter voted $a \succ b \succ c \succ d \succ e$, and b and c were eliminated, the voters implicit vote is $a \succ d \succ e$.)

According to how the *schedule* of the pre-round (who faces whom) is determined, different complexities of the tweaked protocol may be achieved. There are three cases:

- the schedule is decided deterministically *before* the votes are collected
- the schedule is decided randomly *after* the votes are collected
- the schedule is decided *while* the votes are been collected (the two procedures are interleaved)

The complexities of constructive manipulation that are achieved with each of these pre-rounds are NP, $\#P$, and $PSPACE$ respectively. Therefore we can say that, as we make the schedule procedure of the pre-round more sophisticated, the problem of manipulation becomes computationally more complex.

In this section we will present the first case, of deciding the schedule deterministically. Having already discussed the issue of control of the voting procedure of the chair who puts it in practise, we might have concerns that the design of the schedule might allow the possibility of control. In the first case, the chair has no knowledge of the voters' votes , since the schedule is decided before the votes are elicited. So in this case there is no problem of control. In the second case however, since the votes are known, the procedure must be based on a random process, since the chair may be corrupt and want to control the procedure. However, in this second case we end up with an equivalent problem: it may be difficult to check that the chair used indeed (enough) randomness. This problem (among others) is tackled by Lipmaa and Elkind in [**?**], who even in this case make the schedule deterministic, extracting the needed randomness by the votes and not by the chair.

Finally a word of note about the results we present. Conitzer and Sandholm prove their hardness results under the assumption that the number of candidates is unbounded, that there is only one manipulator (rather than a coalition) and that all voters have equal weight. The first assumption is not desired but they couldn' t do without it. The other two however constitute a special case of the general setting where there is a group of manipulators and voters have weights. Therefore hardness results for this special case imply hardness for the general case.

We said we present the NP-hardness proof of the case of adding a deterministic pre-round which is announced before the votes are elicited. The way they prove their result is the following:

- They give a sufficient condition under which adding a pre-round will make the "tweaked" protocol NP-Hard. This sufficient condition is a reduction from SAT that satisfies some constraints.
- They then prove that if such a reduction can be constructed, a pre-round can be added in such a way, so as the manipulator pulls-off a successful manipulation *if and only if* a feasible solution to the SAT problem is found. Therefore, constructive manipulation is NP-hard.
- The last step is to prove that certain voting protocols indeed satisfy the sufficient condition, i.e. a reduction from SAT which satisfies certain constraints can be constructed.

Theorem 19 (The sufficient condition)**.** *Given a voting protocol E, suppose that it is possible,* for any Boolean formula ϕ in conjunctive normal form (i.e., a SAT instance), *to construct in polynomial time a set of votes* over a candidate set containing at least $\{p\} \bigcup C_L$, $C_L = \{c_l : l \in L\}$, where $L = \{\bigcup\{+v, -v\} : \forall v \in V\}$, where V is the set of variables occurring in ϕ, *with the following properties:*

Property 1a If we remove, for each $v \in V$, one of c_{+v} or c_{-v}, p would win an election under protocol E against the remaining candidates *if and only if* for every clause $k \in K$ (where K is the set of clauses in ϕ), there is some $l \in L$ such that c_l has not been removed, and l occurs in k. This should hold even if a single arbitrary vote is added.

Property 1b For any $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election after these votes.

Then it can be proven that constructive-manipulation after adding a deterministic pre-round is NP-hard(and NP-complete if E is deterministic and can be executed in polynomial time).

Proof. Let the candidate set be the set of all candidates occurring in the votes constructed from ϕ (the "original candidates"), plus one dummy candidate for each of the original candidates besides those in C_L. To each of the constructed votes, add all the dummy candidates at the bottom; *let the resulting set of votes be the set of the non-manipulators votes. A single manipulators vote is yet to be added.* Let the schedule for the pre-round be as follows: for each v, c_{+v} and c_{-v} face each other in the preround; and every other original candidate faces (and, because of the dummy candidates position in the votes, defeats) a dummy candidate. Thus, the set of candidates that make it through the pre-round consists of, for each $v \in V$, one of c_{+v} and c_{-v}; and all the other original candidates. The manipulators vote will decide the winner of every c_{+v} and c_{-v} match-up, because *by property 1b*, all these pairwise elections are currently tied. Moreover, it is easy to see that the manipulator can decide the winner of each of these match-ups independently of how it decides the winners of the other match-ups. Thus, we can think of this as the manipulator giving the variables truth-values: v is set to true if c_{+v} survives, and to false if c_{-v} survives. *By property 1a* it then follows that p wins *if and only if* the manipulators assignment satisfies all the clauses, i.e. is a solution to the SAT instance. Hence there is a successful constructive manipulation if and only if there is a solution to the SAT instance, and it follows that constructive manipulation in by adding a deterministic pre-round is NP-hard. (It is also in NP if P is deterministic and can be executed in polynomial time, because in this case, given a vote for the manipulator, it can be verified in polynomial time whether this vote makes p win). □

As we said the next step is to prove that a certain voting procedure satisfies the sufficient condition. Conitzer and Sandholm provide proofs for Plurality, Borda, Maximin, and STV rules. We only give the proof to the Plurality rule.

Theorem 20. Plurality rule satisfies the sufficient condition of Theorem 19

Proof. Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \bigcup C_L\}$, plus a set of candidates corresponding to the set of clauses K of ϕ, $C_K = \{c_k : k \in K\}$.(C_L, C_K are the "original candidates" we mentioned earlier). Then, let the set of votes be as follows: $4|K| + 2$ votes ranking the candidates $p \succ C_L \succ C_K$; for each $k \in K$, $4|K|$ votes ranking the candidates $c_k \succ \{c_{cl} \in C_K : cl \neq k\} \succ C_L \succ p$; and for each $k \in K$, 4 votes ranking the candidates $4\{c_l \in C_L : l \in k\} \succ c_k \in \{c_l \in C_L : l \neq k\} \succ \{c_{cl} \in C_K : cl \neq\} g \succ p$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the C_L!). *Hence, property 1b is satisfied by such a reduction.* We now show property 1a is satisfied. We first observe that regardless of which of the candidates corresponding to literals are removed, p will get $4|K| + 2$ votes. Now,if for each $k \in K$, at least one candidate cl with $l \in k$ remains, then each of the c_k will get precisely $4|K|$ votes. Because each remaining c_l can get at most $4|K|$ votes as well, p will win (if part). on the other hand, if for some $k \in K$, all the candidates c_l with $l \in L$, $l \in k$ are removed, then c_k will get at least $4|K| + 4$ votes and p will not win (only-if part). In both cases there is a margin of at least 2, so a single additional vote will not change this. *Thus, property 1a is satisfied.* □

3.6 Making Control hard by combining existing voting protocols

The same direction of research with the one by Conitzer and Sandholm presented previously follows the paper of E. Hemaspaandra, L. Hemaspaandra, J. Rothe [?], only this time for the setting of control. In abstract terms what they do is put together existing voting rules in such a way, so that the resulting voting rule inherits all of their strengths. Namely the resulting, hybrid protocol, is resistant or even in some cases immune to various types of control if at least one of the protocols combined has this property. Furthermore, the winner is still easily determined if winner determination is easy for all voting protocols that are combined. We can therefore say that *the hybrid protocol maintains the simplicity of the multiple voting protocols it combines while at the same time it inherits their resistance (or even immunity in some cases) to control.*

Particularly HHR showed that combining Plurality and Condorcet rules the hybrid protocol that is produced resists to all types of constructive manipulation. Because of non-existence of voting protocols that combined would give resistance to all types of destructive manipulation, they built some artificial voting rules that in combination achieve this.

Although we are not getting in the details of their proving procedures (which are quite intricate we can say) we would like to stress out the importance of their result. Up until now the field of voting rules was dominated by impossibility results. Arrow's theorem (paradox) is about the inherent unfairness of all voting rules and Gibbard-Satterthwaite theorem is about the inherent manipulability of all voting rules. This result by HHR is one of the few possibility results in the field, stating that control can always be made computationally intractable (at least in terms of worse-case complexity)!

3.7 Steps towards average-case complexity analysis of manipulation

In this section we focus on the setting of manipulation where as we have already seen computational complexity is desired. Until now we have presented results that examine voting rules using the traditional complexity measure in computer science, worst-case complexity. Although many results have proven that various manipulation contexts are computationally hard in the worst-case, all scientists stated their justified concerns as to whether worse-case analysis of manipulation makes even sense. Proving that manipulation in a particular context is hard in the worst case, although demonstrating some measure of resistance to manipulation, it - by no means - preclude it. Therefore we now turn our attention to what seems to be the current direction of research, i.e. what happens "on average". If it is allowed to us we would comment that researchers have been settling until now with worst-case complexity analysis because they could not produce results for the average-case. But the average-case was their desired way of analysis right from the beginning. That is why all results concerning worst case complexity were presented with the concern that worst-case complexity was not a sufficient guarantee of hardness.

3.7.1 How the size of the manipulators' set affects manipulability of the voting procedure

Let us think of national elections, as a typical case of voting. It would be absurd to think that if only a single voter acted strategically she could have any impact on the outcome. So it seems more appropriate to study the case where the set of manipulators consists of more than one voters. The question now is "how many more makes sense?". If there are too few they may still be powerless to make any difference on the outcome; if there are too many (almost as many as the truthful voters), then the problem may be trivial in the sense that the manipulators can always affect the outcome. Such questions are studied by A. D. Procaccia and J. S. Rosenschein in [?] for the case of scoring protocols. We will omit the proof details although they are not very hard to follow. First let us introduce some notation:

- $|M| \rightarrow$ the number of manipulators
- $|N| \rightarrow$ the number of non-manipulators

- $D^i \rightarrow$ non-manipulator's v_i distribution over all the $|C|!$ possible votes
- $S_{i,k} \rightarrow$ the random variable induced by the distribution D^i, which gives the score given to candidate k by voter v_i

When the manipulators cannot affect the instance of the problem, we say that this instance is *closed.*

The first case that is studied is what happens when *the fraction of manipulators is "small".* In quantitative terms we consider "small" to be the case where $|M| = o(\sqrt{|N|})$.

Theorem 21. In the case the fraction of manipulators is "small", under two rather lenient conditions it can be proven that with overwhelming probability (probability converging to 1 as number of voters grows) the instance is *closed.*

The two conditions are:

- D^i being independently distributed
- for all v_i, c_k, c_l there exists $d > 0$ such that $Var[S_{i,k} - S_{i,l}] > d$

Why are these conditions lenient? The first condition implies that the non-manipulative voters vote independently. This is a quite natural condition since each voter has his own/independent preferences. The second implies that there is no voter for whom two candidates have always the same difference in scores. This second condition is a condition of a "minimum randomness requirement" in the votes and is therefore too quite natural to hold for any distribution that makes sense.

Therefore, they proved that when the fraction of manipulators is "small", under each distribution that satisfies two quite natural conditions, the manipulators are most of the times powerless to do anything to affect the outcome of the election.

The authors consider also the case when the fraction of manipulators is "large", or better when it is "large" but not "too large". In quantitative terms this case is when it holds that $|M| = \omega(\sqrt{|N|})$ but at the same time $|M| = o(|N|)$. In this case The results are not that strong but they are interesting.

Theorem 22. Under two conditions, when the fraction of manipulators is "large", then any candidate can be made to win with overwhelming probability iff he belongs to *the set of candidates with maximal expected score (denoted as C').*

The problem is that this time one condition is not that lenient. The two conditions are:

- D^i being independently distributed
- D^i being identically distributed

It is easy to agree that the second condition is too stringent to hold in a real setting.

It is interesting to make the following note for the case of a large fraction of candidates:

- If $|C'| = 1$ then the probability of drawing a closed instance converges to 1 as the number of voters grows.
- If $|C'| \geq 2$, then the probability of drawing an open instance converges to 1 as the number of voters grows.

In the first case, the only voter that belongs in $|C'|$ will most probably be proclaimed winner and the manipulators will be unable to change the outcome, since in order for someone to be proclaimed winner he must be in $|C'|$ and the only such candidate is (with overwhelming probability) already the winner. In the second case the manipulators have (with overwhelmingly high probability) enough power to choose between the members of $|C'|$.

So Procaccia and Rossenschein proved that under some conditions about the distributions of the voters' votes they can determine a big fraction of the manipulation problem, just by examining the ratio between manipulators and non-manipulators. They therefore argue that their results strongly point toward the direction that the manipulation problem can be easily decided on average, or equivalently that the existance of voting rules that are hard to manipulate on average is quite improbable.

3.7.2 A quantitative version of the Gibbart-Satterthwaite Theorem

We have already discussed extensively the Gibbard-Satterthwaite theorem and its importance. We believe (and hope) we have made clear that the Gibbard-Satterthwaite theorem is of tremendous importance for the whole field. Therefore we find it only natural to conclude with a recent result, which is characterized as the *quantitative version of the Gibbard-Satterthwaite theorem.*

The theorem we present is found in a very interesting technical report by E. Friedgut, G. Kalai and N. Nisan [**?**] published on April 2008. We present its gist omitting the (quite extensive) proof. First let us give some definitions.

Definition 3 (manipulation power)**.** The manipulation power of voter i on a social choice function(voting rule) f denoted $M_i(f)$, is the probability that x'_i is a profitable manipulation of f by voter i at profile $x_1 ... x_n$, where $x_1 ... x_n$ and x'_i are *chosen uniformly at random* from all the $|C|!$ choices. By *profitable* we mean ofcourse that x'_i yields an outcome of higher utility for voter i than reporting his true preference order, x_i

Another interesting new element of the paper is that they quantify the *distance of a social choice function from a dictatorship.* Particularly:

Definition 4. Given a function f and a dictatorship g from a probability space X to a set Y we denote the distance between f and g as

$$\Delta(f,g) = Pr_{x \in X}[f(x) \neq g(x)]$$

If G is the family of dictatorships we define $\Delta(f,g) = \min_{g \in G} \Delta(f,g)$.

In our setting X will always be endowed with the uniform probability, so the distance between two functions is nothing else than the proportion of inputs on which the functions disagree.

Definition 5 (neutral SCF)**.** A social choice function is neutral if the names of the candidates "do not matter"; formally, if f commutes with permutations of the candidate set, i.e. $f(\sigma(x_1),...,\sigma(x_n)) = \sigma(f(x_1,...,x_n)$

Now we can present the main result:

Theorem 23. There exists a constant $C > 0$ such that for every $\epsilon > 0$, if f is a neutral social choice function among 3 alternatives for $|V|$ voters that is ϵ-far from dictatorship, then: $\sum_{i=1}^{|V|} M_i(f) \geq C\epsilon$

The above theorem immediately implies that there exists one voter, the one who has the maximum manipulation power, for whom $\max_i M_i(f) \geq \Omega(1/\sqrt{|V|})$. This means that there exists a voter with non-negligible manipulation power.

The striking importance of this theorem is stated as follows:
Even a randomly attempted manipulation has a non-negligible probability of being profitable. Therefore, the computational hardness of manipulation is trivial in the average case!

The theorem is quite striking but there are 2 remaining open problems:

- The theorem is stated only for the special case of three candidates and has to be generalized
- Compared to the Gibbard-Satterthwaite theorem which only requires the non-imposition property, this theorem requires the neutrality property of the SCF which is more strong a requirement (neutrality implies non-imposition).

Chapter 4

Conclusion

In this thesis we tried to present primarily some of the most important results in the field of Computational Social Choice. We showed how the problem of Voting can be seen as a problem of mechanism design. We stressed out the importance of the Gibbart-Satterthwaite theorem on the field of Social Choice Theory and that it gave rise to a new direction of research: studying social choice in computational terms. We studied the problems of winner determination, control, bribery and manipulation in various flavours, focusing heavily on the problem of manipulation. We presented results concerning algorithmic issues, worst-case computational complexity and saw what steps have been taken towards the average-case complexity analysis, which was really the researchers' desired measure of analysis from the beginning.

We believe that in this thesis we have presented enough results to argue that computational aspects of voting do matter. We hope that this thesis will serve as a starting point for other people who want to study the field.

Bibliography

[1] Robert J. Aumann and Adam Brandenburger. Epistemic conditions for nash equilibrium. *Econometrica*, 63(5):1161–80, 1995.

[2] J.J. Bartholdi, C.A. Tovey, and M.A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, pages 392–412, 1988.

[3] J.J. Bartholdi, C.A. Tovey, and M.A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[4] J.J. Bartholdi, C.A. Tovey, and M.A. Trick. How hard is it to control en election? Technical report, 1990.

[5] Ioannis Caragiannis, Jason A. Covey, Michal Feldman, Christopher M. Homan, Christos Kaklamanis, Nikos Karanikolas, Ariel D. Procaccia, and Jeffrey S. Rosenschein. On the approximability of dodgson and young elections. In *SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1058–1067, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[6] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1), September 1971.

[7] Vincent Conitzer and Tuomas Sandholm. Universal voting protocol tweaks to make manipulation hard. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 781–788, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[8] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate? *J. ACM*, 54(3):14, 2007.

[9] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. In *STOC '06: Proceedings of the thirty-eighth*

annual ACM symposium on Theory of computing, pages 71–78, New York, NY, USA, 2006. ACM.

[10] Marquis de Condorcet. Essai sur l'application de l'analyse a la probabilite des decisions rendues a la pluralite des voix. Facsimile reprint of original published in Paris, 1972, by the Imprimerie Royale, 1785.

[11] John Duggan and Thomas Schwartz. Strategic manipulability without resoluteness or shared beliefs: Gibbard-satterthwaite generalized. *Social Choice and Welfare*, 17(1):85–93, 2000.

[12] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2001. ACM.

[13] Edith Elkind, , Edith Elkind, and Helger Lipmaa. Small coalitions cannot manipulate voting. In *The Commonwealth of Dominica*, pages 285–297. Springer-Verlag, 2005.

[14] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. How hard is bribery in elections? *CoRR*, abs/cs/0608081, 2006.

[15] Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. A richer understanding of the complexity of election systems. *CoRR*, abs/cs/0609112, 2006.

[16] Ehud Friedgut, Gil Kalai, and Noam Nisan. Elections can be manipulated often. Discussion paper series, Center for Rationality and Interactive Decision Theory, Hebrew University, Jerusalem, 2008.

[17] Allan Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, July 1973.

[18] Jerry Green and Jean-Jacques Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45(2):427–38, March 1977.

[19] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[20] L. A. Hemachandra. The strong exponential hierarchy collapses. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 110–122, New York, NY, USA, 1987. ACM.

[21] Edith Hemaspaandra and Lane A. Hemaspaandra. Dichotomy for voting systems. *CoRR*, abs/cs/0504075, 2005.

[22] Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Hybrid elections broaden complexity-theoretic resistance to control. *CoRR*, abs/cs/0608057, 2006.

[23] Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Anyone but him: The complexity of precluding an alternative. *Artif. Intell.*, 171(5-6):255–285, 2007.

[24] Christopher M. Homan and Lane A. Hemaspaandra. Guarantees for the success frequency of an algorithm for finding dodgson-election winners. *Journal of Heuristics*, 15(4):403–423, 2009.

[25] Ehud Kalai. Game theory: Analysis of conflict : By roger b. myerson, harvard univ. press, cambridge, ma, 1991. 568 pp. *Games and Economic Behavior*, 3(3):387–391, August 1991.

[26] H.W.jun. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[27] Hannu Nurmi. Voting procedures: A summary analysis. *British Journal of Political Science*, 1983.

[28] Christos H. Papadimitriou, Michael Schapira, and Yaron Singer. On the hardness of being truthful. In *FOCS*, pages 250–259. IEEE Computer Society, 2008.

[29] Christos H. Papadimitriou and Stathis Zachos. Two remarks on the power of counting. In *Proceedings of the 6th GI-Conference on Theoretical Computer Science*, pages 269–276, London, UK, 1982. Springer-Verlag.

[30] Ariel D. Procaccia and Jeffrey S. Rosenschein. Average-case tractability of manipulation in voting via the fraction of manipulators. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–1, New York, NY, USA, 2007. ACM.

[31] Ariel D. Procaccia and Jeffrey S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *J. Artif. Intell. Res. (JAIR)*, 28:157–181, 2007.

[32] Mark Allen Satterthwaite. The existence of strategy-proof voting procedures: A topic in social choice theory. Phd thesis.

[33] V. Vazirani. Approximation algorithms. pages 120–122. Springer-Verlag, 2001.

[34] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[35] Klaus W. Wagner. More complicated questions about maxima and minima, and some closures of np. *Theor. Comput. Sci.*, 51:53–80, 1987.