How to Change a Group's Collective Decision?

Noam Hazon[1] **Raz Lin**[1]
[1]Department of Computer Science
Bar-Ilan University
Ramat Gan Israel 52900
{hazonn,linraz,sarit}@cs.biu.ac.il

Sarit Kraus[1,2]
[2]Institute for Advanced Computer Studies
University of Maryland
College Park MD 20742
sarit@umiacs.umd.edu

Abstract

Persuasion is a common social and economic activity. It usually arises when conflicting interests among agents exist, and one of the agents wishes to sway the opinions of others. This paper considers the problem of an automated agent that needs to influence the decision of a group of self-interested agents that must reach an agreement on a joint action. For example, consider an automated agent that aims to reduce the energy consumption of a nonresidential building, by convincing a group of people who share an office to agree on an economy mode of the air-conditioning and low light intensity. In this paper we present four problems that address issues of minimality and safety of the persuasion process. We discuss the relationships to similar problems from social choice, and show that if the agents are using Plurality or Veto as their voting rule all of our problems are in P. We also show that with K-Approval, Bucklin and Borda voting rules some problems become intractable. We thus present heuristics for efficient persuasion with Borda, and evaluate them through simulations.

1 Introduction

Persuasion is a tool that enables social influence in many crucial areas of human interaction in both personal and business relationships. Persuasion is a key component that shapes marketing campaigns [Tybout, 1978], litigation [Bell and Loftus, 1985], and domestic and international political policy [Cobb and Kuklinski, 1997]. As computers make decisions with people in increasingly complex settings, a need has arisen for the ability of computers to influence people to adopt strategies that will benefit themselves and the group.

This paper considers an automated agent that needs to influence the decision of a group of self-interested agents that must reach an agreement on a joint action. Traditionally, most of the work on group persuasion appears in economic and political science literature (see [Lupia, 1992] for a brief survey)[1]. Previous work in these areas usually assumes that the members of the group have a specific (and restricted) type of preferences, and tends to avoid computational aspects. The work in social choice literature studies manipulation and bribery and considers richer (and more complex) preference models, but an explicit model of persuasion has not been considered in this context (we discuss the connection to manipulation and bribery in Section 3).

In this paper we model persuasion in the context of social choice. Specifically, we assume that there is one agent, the *sender* thereafter, that needs to influence the decision of a group of self-interested agents. The group members reach an agreement through a voting procedure, which is a common mechanism by which a collective decision is made. In a voting procedure, participants (voters) express their preferences via votes, and a voting rule then defines the social outcome chosen as a function of the votes cast.

In every election, there is a need for a reliable entity, responsible for handling the elections; we assume that the sender is that entity. During the election, each voter submits her vote to the sender, that determines the winner. Thus, the sender will have complete information on the voters' preferences. Since the voters will agree to use only a reliable entity as the election organizer, we assume that the sender is compelled (say, by law) to report the true winner of the election, according to the pre-specified voting rule. However, the sender may still affect the elections in the following way. After all the voters submit their votes, the sender may suggest that some of them change their minds and commit to different votes. These voters are not allowed to vote again or change their votes arbitrarily. Instead, they are given the opportunity to approve or disapprove the vote suggested to them by the sender. The voters are not compelled to accept the sender's suggestion, and they may want to retain their current vote. However, as a reliable entity, the sender sends a suggestion only to voters who benefit from her suggestions.

Consider the example of the agent that tries to reduce the energy consumption of a nonresidential building where human occupants do not have direct financial incentive to save energy. If the agent is able to convince some of the occupants to agree to her suggestion, then the overall energy consumption of the building will be reduced. Suppose that

[1]We note that "persuasion" is also a common term in the field of argumentation, where there is a dialog between the persuader and the listener. We follow the works in economic which use this term in any situation where an agent with some preferences tries to influence the decision of other agents.

there are 4 options, and 9 people, with the following preferences. Two people prefer $3 \succ 4 \succ 1 \succ 2$, two people prefer $4 \succ 2 \succ 1 \succ 3$, two people prefer $2 \succ 3 \succ 1 \succ 4$ and three people prefer $1 \succ 2 \succ 3 \succ 4$. Suppose also that 1 is the option with the lowest energy consumption and 2 is the most wasteful option. Using the Borda rule, which will be defined later, the chosen alternative is 2. However, if the agent persuades the first two people to approve the vote $3 \succ 1 \succ 4 \succ 2$ instead of their original vote, option 1 will be selected, which is more energy efficient and is also preferred over option 2 by the people that were convinced. We note that since the people do not know the preferences of the others, they will not have an incentive to submit a non-truthful vote in order to manipulate the system. In addition, as a reliable entity, the agent's decision making should be operated with transparency, so she cannot decide on an alternative by her own, ignoring the preferences of the people, and she cannot change their votes without their approval.

We propose four variants of the group persuasion problem. The basic problem is PERSUASION, where we ask the sender to find a set of voters together with suggestions that will benefit her and these voters. Since sending messages to many voters may be costly (in terms of time or money), we are also interested in an optimization version, where the sender is interested in influencing the election while sending the least number of messages. The corresponding decision problem is k-PERSUASION. Since the voters are not compelled to accept the sender's suggestions (even though they are guaranteed a more favorable alternative should they all accept the suggestions), the sender or the voters may attain an even worse outcome if not all the voters accept the senders suggestions. The set of suggestions that can never lead to a worse outcome is called *safe*, and we introduce the safe variants of the PERSUASION and k-PERSUASION problems, SAFE-PERSUASION and k-SAFE-PERSUASION, respectively. Our suggested problems are very similar to other problems in social choice. Specifically, PERSUASION is very similar to the unweighted coalition manipulation problem (UCM) [Conitzer and Sandholm, 2002; Conitzer *et al.*, 2007] and k-PERSUASION is very similar to the Bribery problem [Faliszewski *et al.*, 2006]. We discuss the relationships and point out the differences in Section 3, but we would like to emphasize two important differences here. In UCM and Bribery the assumption is that the manipulative voters always vote according to the suggested manipulation, and there is no requirement that they will benefit from it. Therefore, many works in UCM and Bribery considered the context where one would like to prevent the option to affect the voters, and thus hardness of computation is advantageous. In contrast, in our case the sender is allowed to send a suggestion only to voters that will benefit from it, voters can accept or decline the sender's suggestions, and we sometimes also require that the suggestions will be safe for the voters. These assumptions are more suitable to the context where the sender's preferences represent some desirable properties, and one would like to enable the sender to influence the decision of the voters. Thus, we searched for efficient algorithms for this purpose. In addition, in UCM and Bribery it is not so easy for the manipulators to attain complete information on the voter's preferences. In our setting, since the sender is the election organizer, this assumption is well-justified (see also [Xia, 2012] which renders this distinction).

In this paper, we investigate the algorithmic aspects of the four persuasion problems. We show that all problems are easy when using Plurality or Veto voting rules. Although PERSUASION is easy for K-Approval and Bucklin, we show that k-PERSUASION and k-SAFE-PERSUASION are hard (in terms of computational complexity) with these rules. With Borda, all of our problems are hard. We thus propose heuristics for k-PERSUASION and k-SAFE-PERSUASION with Borda, and evaluate their performance through simulations. Our heuristics are not complete, but they are proven correct, e.g., if the heuristic for SAFE-PERSUASION finds a set of suggestions, it is guaranteed to be safe and benefit the sender and the voters.

2 Preliminaries and Definitions

We have a set of *actions* (also referred to as *alternatives*) $A = \{a_1, \ldots, a_m\}$ and a set of *voters* $V = \{1, \ldots, n\}$. Each voter i is represented by her *preference* R_i, which is a total order over A; we will also refer to total orders over A as *votes*. For readability, we will sometimes denote the order R_i by $\succ_i$ (and thus $a \not\succ_i a$). Given a preference R_i, we will denote the alternative that is ranked in the j-th position in R_i as $R_i(j)$, and by $rank(a, R_i)$ the position of alternative a in R_i. The vector $\mathcal{R} = (R_1, \ldots, R_n)$ is called a *preference profile*. We have one sender, s, which also has preferences over the available actions, denoted R_s.

We will use voting rules to determine which action will be selected. A *voting rule* $\mathcal{F}$ is a mapping from the set of all preference profiles to the set of actions. The voting rules that we consider assign scores to all alternatives; one winner is then selected among the alternatives with the highest score using a tie-breaking rule. To simplify the analysis, we assume that the tie-breaking rule is *lexicographic*, i.e., given a set of tied alternatives, it selects one that is maximal with respect to a fixed ordering $\succ$.

We will now define the voting rules considered in this paper. Given a vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ with $\alpha_1 \geq \cdots \geq \alpha_m$, the *score* $s_\alpha(a)$ of an alternative $a \in A$ under a *positional scoring rule* F_α is given by $\sum_{i \in V} \alpha_{rank(a,R_i)}$. Many classic voting rules can be represented using this framework. Indeed, *Plurality* is the scoring rule with $\alpha = (1, 0, \ldots, 0)$, *Veto* (also known as *Antiplurality*) is the scoring rule with $\alpha = (1, \ldots, 1, 0)$, and *Borda* is the scoring rule with $\alpha = (m-1, m-2, \ldots, 1, 0)$. Further, K-*Approval* is the scoring rule with α given by $\alpha_1 = \cdots = \alpha_{\mathrm{K}} = 1$, $\alpha_{\mathrm{K}+1} = \cdots = \alpha_m = 0$.

The *Bucklin rule* can be viewed as an adaptive version of K-Approval. We say that w, $1 \leq w \leq m$, is the *Bucklin winning round* if for any $j < w$ no alternative is ranked in the top j positions by at least $\lceil n/2 \rceil$ voters, and there exists some alternative that is ranked in the top w positions by at least $\lceil n/2 \rceil$ voters. We say that the alternative a's *score in round* j is its j-Approval score, and its *Bucklin score* $s_B(a)$ is its w-Approval score, where w is the Bucklin winning round. The *Bucklin winner* is the alternative with the highest Bucklin score. Note that the Bucklin score of the Bucklin winner is at least $\lceil n/2 \rceil$. If the context is clear, we refer to the score of an

alternative a simply by $s(a)$.

Now, the sender sends suggestions only to voters who will benefit from her suggestions. Let S_i be the suggestion that was sent to voter i. If no suggestion was sent to a voter i, then $S_i = R_i$. The vector $\mathcal{S} = (S_1, \ldots, S_n)$ represents the suggestions that were sent to the voters. We use $V^{\leftarrow}$ to denote the set of all voters who received a suggestion from the sender, and $V^{\checkmark} \subseteq V^{\leftarrow}$ denotes the set of all voters who accepted the suggestion. For any $V' \in \{V^{\leftarrow}, V^{\checkmark}\}$, let $\mathcal{R}_{-V'}$ be the profile obtained from $\mathcal{R}$ by replacing R_i with S_i for all $i \in V'$. We are now ready to define our basic problem.

Definition 1. *In the* PERSUASION *problem we are given a set A of alternatives, a set V of voters specified via their preferences, and a preference list of a sender R_s. We are asked whether there exist a subset of voters, $V^{\leftarrow} \subseteq V$, together with suggestions $\mathcal{S}$, such that $\mathcal{F}(\mathcal{R}_{-V^{\leftarrow}}) \succ_i \mathcal{F}(\mathcal{R})$ for all $i \in V^{\leftarrow} \cup \{s\}$.*

That is, in the PERSUASION problem we assume that all the voters accept their suggested votes, and we would like to find suggestions such that the chosen alternative will be preferred over the current winner by the sender and by all the voters who received a message from the sender. If the answer for PERSUASION is "yes", we will sometimes call the set of suggestions $\mathcal{S}$ a *successful persuasion* (and similarly for the rest of the problems). The corresponding optimized version is k-PERSUASION which is defined the same as PERSUASION, with the added requirement that $|V^{\leftarrow}| \leq k$ for a given positive integer k.

As noted above, in the PERSUASION problem we assume that $V^{\leftarrow} = V^{\checkmark}$. Since the voters are not compelled to accept the suggestions (even though they are guaranteed a more favorable alternative should they all accept the suggestions), the sender or the voters in $V^{\leftarrow}$ may attain an even worse outcome if not all the voters accept the sender's suggestions. The set of suggestions that can never lead to a worse outcome is called *safe*, and we now introduce the safe variants of the PERSUASION and k-PERSUASION problems.

Definition 2. *In the* SAFE-PERSUASION *problem we are given a set A of alternatives, a set V of voters specified via their preferences, and a preference list of a sender R_s. We are asked whether there exist a subset of voters, $V^{\leftarrow} \subseteq V$, together with suggestions $\mathcal{S}$, such that $\mathcal{F}(\mathcal{R}_{-V^{\leftarrow}}) \succ_i \mathcal{F}(\mathcal{R})$ for all $i \in V^{\leftarrow} \cup \{s\}$ and for any $V^{\checkmark} \subseteq V^{\leftarrow}$, either $\mathcal{F}(\mathcal{R}_{-V^{\checkmark}}) \succ_i \mathcal{F}(\mathcal{R})$ for all $i \in V^{\checkmark} \cup \{s\}$ or $\mathcal{F}(\mathcal{R}_{-V^{\checkmark}}) = \mathcal{F}(\mathcal{R})$. In the k-SAFE-PERSUASION problem we also require that $|V^{\leftarrow}| \leq k$ for a given positive integer k.*

Given any preference profile and two distinct alternatives $c, a^* \in A$, we state that c is *stronger* than a^* if $s(c) > s(a^*)$ or $s(c) = s(a^*)$ but the tie-breaking rule favors c over a^*, and c is *weaker* than a^* if c is not stronger than a^*. We define the set of *potential voters* as $\mathcal{P}(\mathcal{R}, c, a^*) = \{i \in V : a^* \succ_i c\}$. In addition, if a voter i switches from R_i to S_i and thus the total score of an alternative a increases/decreases, we state that i increases/decreases the score of a.

We say that $A \leq_m^p B$ (A many-one polynomial time reduces to B) if there is a polynomial-time computable function f such that $x \in A \Leftrightarrow f(x) \in B$.

3 Persuasion Problems versus Manipulation, Bribery and Safe Manipulation

Before we proceed to analyze our problems with the different voting rules, we would like to show the links and point out the differences between our problems and other computational problems that were studied in computational social choice. The PERSUASION problem is very similar to the unweighted coalition manipulation problem (UCM), introduced by [Conitzer and Sandholm, 2002; Conitzer *et al.*, 2007]. However, in UCM we are allowed to add voters (or, equivalently, to change the votes of specific voters). In PERSUASION, we are allowed to make a suggestion to *every* voter who will benefit from the suggestion. Nevertheless, it is easy to see that PERSUASION $\leq_m^p$ UCM, for any voting rule.

Proposition 1. *For a given voting rule $\mathcal{F}$, if there is a polynomial time algorithm for UCM, there is a polynomial algorithm for* PERSUASION.

Proof. The idea is simple: if the current winner is alternative c, iterate over all the alternatives $a \in A$ such that $a \succ_s c$ (all the alternatives that are preferred by the sender over the current winner), to cover all the possible options to persuade the voters. Let a^* be such an alternative, and set $V^{\leftarrow} = \mathcal{P}(\mathcal{R}, c, a^*)$. Now, create a new preference profile $\mathcal{R}'$ by removing all the voters from $V^{\leftarrow}$ and run the algorithm for UCM on $\mathcal{R}'$ (set the number of manipulators to $|V^{\leftarrow}|$). It is easy to see that there is a successful manipulation for a^* in $\mathcal{R}'$ if and only if there is a set of suggestions $\mathcal{S}$ to the voters in $V^{\leftarrow}$ that will make a^* the winner. Since the algorithm iterates over at most m options, it runs in polynomial time. □

The k-PERSUASION problem is very similar to the Bribery problem [Faliszewski *et al.*, 2006; 2009]. However, there is one main difference. In Bribery we are allowed to change the preferences of all the voters, even ones that prefer the current winner to the alternative that we attempted to make as the winner of the elections. In our case, the sender is allowed to make a suggestion only to voters who will benefit from her suggestion. Due to this difference, it seems that there is no general reduction from k-PERSUASION to Bribery or the other way around. In addition, [Faliszewski *et al.*, 2009] showed that unless $P = NP$ there is no general reduction from UCM to Bribery. In our case, we note that PERSUASION $\leq_m^p$ k-PERSUASION and SAFE-PERSUASION $\leq_m^p$ k-SAFE-PERSUASION.

The k-PERSUASION problem is also very similar to the "minimum manipulation coalition size" problem, which aims at finding the smallest number k such that changing k votes can change the current winner [Nitzan, 1985; Chamberlin, 1985; Pritchard and Wilson, 2009]. Similar to k-PERSUASION, all voters who change their votes must prefer the new winner to the old one. However, in this problem there are no specific target alternatives which the coalition will try to make the winner. Rather, this is a destructive version of manipulation, where the coalition's objective is to ensure that the current winner will not be the winner. In addition, the analysis of this problem concentrated on the possibility of manipulation or the distribution over the required coalition size when the number of voters increases to infinity, given a known distribution on the votes. In our analysis, we demonstrate how

to find the set of suggestions for a given problem (or prove hardness).

The notion of safety in the context of coalitional manipulation was introduced by [Slinko and White, 2008]. In their setting a potential manipulator v announces how she intends to vote, and some of the other voters whose preferences coincide with those of v may follow suit. A manipulative vote is called safe if for some number of followers it improves the outcome from v's perspective, and can never lead to a worse outcome. Extensions of this model were proposed by [Hazon and Elkind, 2010], where the followers' preferences may differ from those of the leader. However, both in the original model and in its extensions, the leader is restricted to propose one manipulative vote to all the voters. Our work can be seen as a generalization of this work, since we allow the sender to suggest different votes to the voters. In addition, we are the first to provide a heuristic which guarantees safety.

4 Plurality and Veto

In this section we show that all of our problems can be decided in polynomial time for Plurality and Veto voting rules. Since PERSUASION $\leq_m^p$ k-PERSUASION and SAFE-PERSUASION $\leq_m^p$ k-SAFE-PERSUASION, it is sufficient to show that the optimization versions, i.e., k-PERSUASION and k-SAFE-PERSUASION, are tractable. We begin with Plurality, which is the most widely-used voting rule in political elections at the moment. We say that voter i votes for alternative a if i ranks a first.

Theorem 2. k-PERSUASION *and* k-SAFE-PERSUASION *with Plurality are in* P.

Proof. Let c be the current winner. We note that the sender cannot send any suggestion to voters that vote for c. Thus, the score of c cannot be decreased. We iterate over all the alternatives $a \in A$ such that $a \succ_s c$, to cover all the possible options to persuade the voters. Let a^* be such an alternative, and let $g_{ca^*} = |s(c) - s(a^*)|$. If,
(1) $g_{ca^*} \leq k$ in the case that the tie-breaking rule favors a^* over c, or $g_{ca^*} < k$ otherwise, and
(2) There are at least g_{ca^*} voters from $\mathcal{P}(\mathcal{R}, c, a^*)$ who do not vote for a^*,
then there is a successful persuasion; the sender suggests to these voters to vote for a^*. It is easy to verify that if one of the conditions does not hold there is no set of k suggestions that can make a^* the winner, and checking these conditions can be done in polynomial time. For k-SAFE-PERSUASION, we already noted that the score of c cannot be decreased. In addition, we increase the score of a^* and the score of all other alternatives does not increase. Therefore, the solution found by the algorithm is also a safe k-persuasion. □

With Veto, the analysis is slightly different. We say that voter i vetoes alternative a if i ranks a last, i.e., $R_i(m) = a$.

Theorem 3. k-PERSUASION *with Veto is in* P.

Proof. Let c be the current winner. The suggested algorithm iterates over all the alternatives $a \in A$ such that $a \succ_s c$, to cover all the possible options to persuade the voters. Let a^* be such an alternative. We observe that the sender cannot send any suggestion to voters that veto a^*. Thus, the score of a^* cannot be increased. The algorithm tries to decrease the score of all the alternatives that are stronger than a^* in the following way. It iterates over all the voters from $\mathcal{P}(R, c, a^*)$. Given a voter $i \in \mathcal{P}(R, c, a^*)$, if the vetoed alternative in R_i, $R_i(m)$, is stronger than a^* or weaker than a^* but has the same score of a^*, the algorithm skips to the next voter. Otherwise, the algorithm suggests to i to veto an alternative that is currently stronger than a^* (instead of $R_i(m)$), and updates the scores accordingly. If after k suggestions there are still alternatives stronger than a^* then there is no k-persuasion for a^*. An inductive argument shows that this algorithm always finds a successful k-persuasion if it exists, and it clearly runs in polynomial time. □

Not all successful k-persuasions found by the algorithm in the proof of Theorem 3 are safe. However, only two small modifications are needed for the algorithm to guarantee that a k-safe-persuasion (if it exists) will be found.

Theorem 4. k-SAFE-PERSUASION *with Veto is in* P.

Proof. Let c be the current winner, and suppose searching for a k-persuasion that will make the alternative a^* the winner. If there is at least one alternative b such that $c \succ_s b$ and b is stronger than a^*, then there is no successful k-persuasion for a^* which is also safe. Indeed, suppose there is a successful k-persuasion which sends suggestions to $V^{\leftarrow}$ such that $\mathcal{F}(\mathcal{R}_{-V^{\leftarrow}}) = a^*$. Since every voter decreases the score of exactly one alternative and $V^{\leftarrow}$ is a successful k-persuasion, there exists a subset of voters, $V_b \subseteq V^{\leftarrow}$, that veto b till b is weaker than a^*. If we set $V^{\checkmark} = V^{\leftarrow} \setminus V_b$ we find that $\mathcal{F}(\mathcal{R}_{-V^{\checkmark}}) = b$, thus the k-persuasion is not safe. Similarly, in any successful k-persuasion there is a subset of voters $V_c \subseteq V^{\leftarrow}$ that veto c till c is weaker than a^*. If there is an alternative b such that $c \succ_i b$ for a voter $i \in V_c$ and b is stronger than a^*, then i cannot be part of V_c in any successful k-persuasion for a^* which is also safe, due to the same reasoning. Thus, in order to find a k-safe-persuasion we use the algorithm from the proof of Theorem 3 with the following two modifications. Before iterating over all voters from $\mathcal{P}(\mathcal{R}, c, a^*)$ we add a step which checks if an alternative b exists such that $c \succ_s b$ and b is stronger than a^*. If so, the algorithm skips to check the next possible $a^* \in A$. Also, when iterating over the voters from $\mathcal{P}(\mathcal{R}, c, a^*)$, before making a suggestion to voter i the algorithm checks if an alternative b exists such that $c \succ_i b$ and b is stronger than a^*. If so, the algorithm skips to the next voter. Therefore, if the modified algorithm finds a k-persuasion it is also a k-safe-persuasion, and an inductive argument shows that this algorithm always finds a successful k-safe-persuasion, if it exists. □

5 K-Approval, Bucklin and Borda

Unlike Plurality and Veto, in K-Approval, Bucklin and Borda voting rules a single voter can decrease and increase the score of multiple alternatives at once. Some of our problems thus become intractable. We begin with K-Approval. It is known that UCM for K-Approval is in P [Lin, 2011]. We thus immediately attain a polynomial time algorithm for PERSUASION with K-Approval according to Proposition 1. However, k-PERSUASION and k-SAFE-PERSUASION become NP-hard.

Theorem 5. k-PERSUASION *and* k-SAFE-PERSUASION *with* K-*Approval* [2] *are* NP*-hard, for each* K ≥ 5.

The proof for K-Approval is very similar to the proof for Bucklin (it is even simpler) which we demonstrate below, thus we omit this proof due to space constraints. As proved by [Xia *et al.*, 2009], UCM for Bucklin is also in P, and we again immediately attain a polynomial time algorithm for PERSUASION with Bucklin according to Proposition 1. We now show that k-PERSUASION and k-SAFE-PERSUASION become NP-hard with Bucklin.

Theorem 6. k-PERSUASION *and* k-SAFE-PERSUASION *with Bucklin are* NP*-hard.*

Proof. Our reduction is from the exact cover by 3-sets problem (X3C), defined as follows. Given a ground set $G = \{g_1, \ldots, g_{3t}\}$ and a collection $\mathcal{X} = \{X_1, \ldots, X_n\}$ of subsets of G with $|X_j| = 3$ for $j = 1, \ldots, n$. It is a "yes"-instance if G can be covered with exactly t sets from $\mathcal{X}$, and a "no"-instance otherwise. We can assume without loss of generality that $n > t$: otherwise either an exact cover is impossible or the instance is easily solvable. Given an input $(G, \mathcal{X})$ we construct an instance of our problem as follows. The set of alternatives consists of G, two extra alternatives $\{a^*, c\}$, and a set $D = D_{\mathcal{X}} \cup D_{nd}$ of dummy alternatives, $|D_{\mathcal{X}}| = 3t + 1$ and $|D_{nd}| = 3t + 2$. For each i, $1 \leq i \leq 3t$, let x_i be the number of sets X_j that contain g_i. We construct the following two sets of voters.

For each set $X_i \in \mathcal{X}$, we construct a voter i with preferences: $a^* \succ_i c \succ_i X_i \succ_i D_{\mathcal{X}} \succ_i (G \setminus X_i) \cup D_{nd}$. The alternatives in X_i, $D_{\mathcal{X}}$ and $(G \setminus X_i) \cup D_{nd}$ are placed in arbitrary order by i. Let $V_{\mathcal{X}}$ denote the set of all such voters.

We construct a second set of $3n - 2t$ voters where n voters rank c first, and the other $2n - 2t$ voters rank $d_c \in D_{nd}$ first. Then, $2n - x_1 - t + 1$ voters rank g_1 second, and the other $n + x_1 - t - 1$ voters rank $d_1 \in D_{nd}$ second. Then, $2n - x_2 - t + 1$ voters rank g_2 third, and the other $n + x_2 - t - 1$ voters rank $d_2 \in D_{nd}$ third. In general, for each i, $1 \leq i \leq 3t$, $2n - x_i - t + 1$ voters place g_i in the $(i+1)$-th position, and the other $n + x_i - t - 1$ voters place $d_i \in D_{nd}$ in the $(i+1)$-th position. Then, $2n - 2t$ voters place $d_p \in D_{nd}$ in the $(3t+2)$-th position, c in the $(3t+3)$-th position and a^* in the $(3t + 4)$-th position. The other n voters (which have already ranked c first) place a^* in the $(3t + 2)$-th position. The rest of the alternatives are placed by all the voters in arbitrary order after a^*. Note that for each voter i in this set, $c \succ_i a^*$. We complete the reduction by setting $k = t$ and the preference of the sender to $a^* \succ_s c \succ_s G \cup D$. Note that the Bucklin score of every winner in this setting is at least $2n - k + 1$.

In the current setting c wins in round 2 with $2n$ points, and all other alternatives attain less than $2n$ points in the first two rounds. Since c is ranked second by the sender, the only possible k-persuasion is to make a^* the winner by suggesting to at most k voters from $V_{\mathcal{X}}$ to change their vote. We note that this is possible if and only if the sender sends suggestions that correspond to an exact cover by 3-sets of G. Details are left out due to space restrictions. As for k-SAFE-PERSUASION, we observe that the instance is built such that only when $V^{\checkmark} = V^{\leftarrow}$ the $3t + 2$-Approval score of c is $2n - k$ as required. If $V^{\checkmark} \subset V^{\leftarrow}$, then $\mathcal{F}(\mathcal{R}_{-V^{\checkmark}}) = c = \mathcal{F}(\mathcal{R})$. Therefore, any successful k-persuasion for the instance is also a successful k-safe-persuasion. □

[2] The k in k-PERSUASION and the K in K-Approval are different parameters.

As for Borda, the complexity of UCM for Borda was recently resolved by [Davies *et al.*, 2011] and [Betzler *et al.*, 2011], who show that UCM is NP-complete for Borda. We show that all of our problems are NP-hard with Borda. As noted above, it is sufficient to show that PERSUASION and SAFE-PERSUASION are NP-hard.

Theorem 7. PERSUASION *and* SAFE-PERSUASION *with Borda are* NP*-hard.*

In essence, we use the same construction of [Betzler *et al.*, 2011], but for the PERSUASION problem we need to fill the preferences of W (i.e., the set of manipulative votes in the definition of UCM for Borda by [Betzler *et al.*, 2011]). Let c be the winner of the resulting elections. We make sure that:
(1) For the sender, $c^\star \succ_s c \succ_s (C \cup D) \setminus \{c, c^\star\}$.
(2) For every voter $i \in V$, $c \succ_i c^\star$.
(3) For every voter $j \in W$, $c^\star \succ_j c$, and $\mathcal{F}(\mathcal{R}_{-\{R_j\}}) = c$.
The full proof is omitted due to space constraints.

6 Heuristics for Borda

The Borda voting rule or one of its modifications are used by many organizations and competitions, but unfortunately all the persuasions problems with Borda are NP-hard. In this section we provide heuristics for k-PERSUASION and SAFE-PERSUASION when using the Borda voting rule.

6.1 k-Persuasion

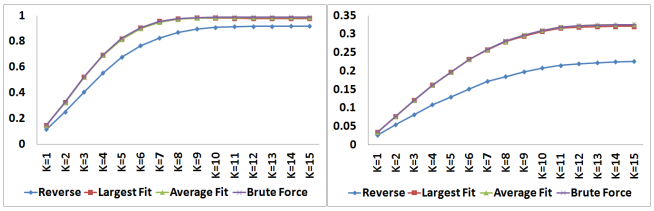
The challenge in providing a successful k-persuasion is double: we need to decide which k voters to include in $V^{\leftarrow}$ and what suggestions to provide. Our heuristic deals with the first challenge. It assigns scores to the voters and then greedily chooses the k voters with the highest score. Let c be the current winner, and a^* the alternative that we will try to render the winner. Given a voter $i \in V$, the score of i is $rank(a^*, R_i) - 1 + m - rank(c, R_i)$ (where $rank(a, R_i)$ is the position of alternative a in R_i). That is, the heuristic score reflects the ability of a voter to affect the elections by increasing the score of a^* and decreasing the score of c. We also tested a more involved heuristic, which takes into account the position of other alternatives, but since it did not perform better than the simple version we do not present it here.

After the selection of k voters we need to generate a set of suggestions. We thus use the current available heuristics for UCM for Borda. The first heuristic, introduced by [Zuckerman *et al.*, 2009] is REVERSE (we follow the terms provided by [Davies *et al.*, 2011]). This heuristic fills the votes of each voter in turn. It ranks the desired alternative a^* first, and the remaining alternatives are placed in reverse order of their current Borda scores. The heuristic was shown by [Zuckerman *et al.*, 2009] to actually be an approximation algorithm, which guarantees that a manipulation that uses at most one more manipulator than is optimal will be found. A different approach was proposed by [Davies *et al.*, 2011], which is

inspired by bin packing and multiprocessor scheduling. They suggested two heuristics, LARGEST FIT and AVERAGE FIT, and experimentally showed that these heuristics significantly outperform REVERSE. Specifically, with AVERAGE FIT they were able to find optimal manipulations in almost all the randomly generated elections they tested.

In our experiments we tested our heuristic for selecting the k voters, combined with the aforementioned heuristics for generating suggestions. We used the same repository of votes of [Davies *et al.*, 2011], which generated either uniform random votes or votes drawn from a Polya-Eggenberger urn model [Berg, 1985]. The urn model tries to capture varying degrees of social homogeneity by introducing correlation between votes. At first, all the possible $m!$ votes are placed in an urn. Then, votes are drawn from the urn at random, and are placed back into the urn along with b other votes of the same type. In our setup $b = m!$ so that there is a 50% chance that the second vote will be the same as the first.

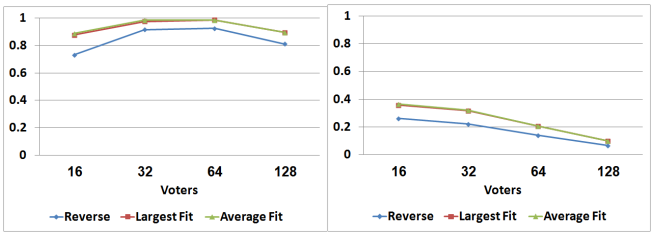
In the first set of experiments we checked how k would affect the performance of the heuristic. We fixed the number of voters $n = 32$ and the number of alternatives $m = 8$. Since these values were relatively small, we were also able to compare our heuristic to a Brute-Force approach, which checks every possible subset of k voters, runs REVERSE, LARGEST FIT and AVERAGE FIT to fill in the voters' preferences, and chooses the best one. We generated 1000 instances and for each instance and each value of k we checked which alternatives could be made the winners by sending suggestions to at most k voters. Figure 1 depicts the results, where the y-axis is the percentage of alternatives that could be made the winners, and each point is an average over 1000 instances. Surprisingly, our heuristic with either LARGEST FIT or AVERAGE FIT achieves the same performance compared to the Brute-Force approach, even though the running time of our heuristic is 500 times faster than the Brute-Force approach (on average). With REVERSE our heuristic's performance is consistently behind, and it does not close the gap even when we increase k to its maximum value. As indicated by [Davies *et al.*, 2011], in the urn model many more voters are required to generate a successful manipulation. This explains the difference between the uniform and the urn model in the percentage of alternatives that can be made winners.



(a) Uniform distribution (b) Urn distribution

Figure 1: Success ratio with k-PERSUASION

In the next set of experiments we set $k = 12$ and the number of alternatives $m = 8$ and varied the number of voters. The results are presented in Figure 2. As before, the y-axis is the percentage of alternatives that could be made the winners and each point is an average over 1000 instances. We can see that our heuristic with either LARGEST FIT or AVERAGE FIT performs better than with REVERSE in all the scenarios. Since k is a fixed number, when the number of voters increases there are less alternatives that can be made winners.



(a) Uniform distribution (b) Urn distribution

Figure 2: Success ratio with k-PERSUASION

6.2 Safe-Persuasion

In order to describe the heuristic we introduce additional notations. As before, let c be the current winner, and a^* the alternative that we will try to render the winner. Let $B = \{a \in A \mid (c \succ_s a) \vee (c \succ_i a, i \in \mathcal{P}(\mathcal{R}, c, a^*))\}$. We denote by $maxA^*$ the score of a^* when all the voters from $\mathcal{P}(\mathcal{R}, c, a^*)$ rank a^* first. We define $target = maxA^*$ in the case that the tie-breaking rule favors a^* over c, and $target = maxA^* - 1$ otherwise. For every $b \in B$, let $aGap(b) = target - s(b)$ in the case that the tie-breaking rule favors c over b, and $aGap(b) = target - s(b) - 1$ otherwise. For every other alternative $a \notin B$, $aGap(a) = \infty$. Let $B^h = \{b \in B \mid b \text{ is stronger than } a^* \text{ in } \mathcal{R}\}$ and for each such alternative b we define $cGap(b) = s(c) - s(b)$ in the case that the tie-breaking rule favors c over b, and $cGap(b) = s(c) - s(b) - 1$ otherwise. Finally, let $A' = A \setminus (B^h \cup \{c, a^*\})$. In the description of the algorithm we abuse notation and use $s(a)$ to denote the score of alternative a in the preference profile $\mathcal{R}_{-\mathcal{P}(\mathcal{R},c,a^*)}$ (i.e., the preference profile that is obtained from $\mathcal{R}$ when replacing R_i with S_i for every voter $i \in \mathcal{P}(\mathcal{R}, c, a^*)$).

The algorithm works as follows. It ranks a^* first in the preferences of every voter from $\mathcal{P}(\mathcal{R}, c, a^*)$. Thus, a^* attains its maximum score, $maxA^*$. If as a result a^* is stronger than c, the task has been completed. The resulting persuasion is also safe, since the score of c does not decrease and the score of all the other alternatives does not increase. If a^* is still weaker than c, the voters need to decrease the score of c, but the algorithm does not allow the score of c to go below $target$. Thus, the algorithm iterates over the voters and places the alternatives in reverse order according to their current score (as REVERSE does). However, it keeps track of the alternatives from B, since they are the alternatives that can break the safety of the persuasion. The algorithm thus assigns for each alternative $b \in B$ the variable $aGap(b)$, which represents the maximum increase in score that b can attain. Whenever a voter increases the score of alternative b, $aGap(b)$ is updated, and if the filling of S_i results in an increase in b's score above $aGap(b)$ (negative $aGap$ in the algorithm) the algorithm resets all the alternatives to their original positions in S_i (except for a^*). One more requirement is necessary to ensure safety. The set B^h consists of the alternatives from B that are stronger than a^* in the original profile. Since the score

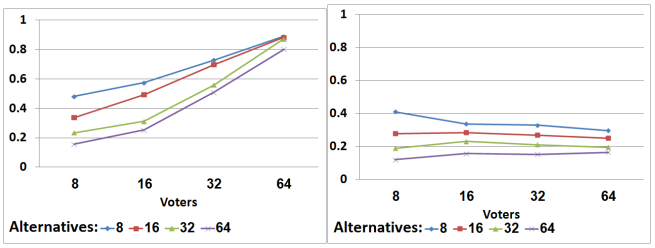
Algorithm 1 SafePersuasion($\mathcal{R}, c, a^*$)

Init $\mathcal{S} \leftarrow \mathcal{R}$, when all the voters from $\mathcal{P}(\mathcal{R}, c, a^*)$ rank a^* first
if $s(c) \leq target$ **then**
  return $\mathcal{S}$
for all $i \in \mathcal{P}(\mathcal{R}, c, a^*)$ **do**
  $backup[] \leftarrow$ the current $aGap$ and $cGap$ of all the candidates
  if $B^h \neq \emptyset$ **then**
    $dec \leftarrow m - \max_{b \in \{B^h \cup \{c\}\}} rank(b, S_i)$
    move all $b \in B^h$ and c down by dec positions in S_i
    if $s(c) < target$ **then**
      move all $b \in B^h$ and c up in S_i until $s(c) = target$
    else if c is not last in S_i **then**
      $dec \leftarrow \min\{\min_{b \in B^h} cGap(b), m - rank(c, S_i)\}$
      if $s(c) - dec < target$ **then**
        $dec \leftarrow s(c) - target$
      $e \leftarrow S_i(rank(c, S_i) + dec)$
      if $e \notin B^h$ **then**
        switch between e and c
      else
        if $\exists a$ such that a is positioned between c and e and $a \notin B^h$ and given such a with the highest rank, $cGap(e) - dec - (rank(e, S_i) - rank(a, S_i)) \geq 0$ **then**
          decrease $cGap(e)$ by $rank(e, S_i) - rank(a, S_i)$
        else if $\exists a$ such that a is positioned lower than e and $a \notin B^h$ **then**
          find such a with the lowest rank
        if a suitable a was found by one of the previous conditions **then**
          put e in $rank(a, S_i)$
          put a in $rank(c, S_i)$
          put c in $rank(c, S_i) + dec$
      if c's position has been changed **then**
        decrease dec from $cGap$ of all $b \in B^h$
  else
    $orig \leftarrow rank(c, S_i)$
    place c last in S_i
    while $aGap(S_i(rank(c, S_i) - 1)) = 0$ and $rank(c, S_i) > orig$ **do**
      move c up by one position in S_i
    if $s(c) < target$ **then**
      move c up in S_i until $s(c) = target$
  sort A' according to the current scores $s(\cdot)$
  place the alternatives from A' in all the positions in S_i that are not occupied by alternatives from $B^h \cup \{c, a^*\}$, in reverse order of their scores $s(\cdot)$
  for all $b \in B$ **do**
    if $rank(b, S_i) > rank(b, R_i)$ **then**
      $aGap(b) \leftarrow aGap - (rank(b, S_i) - rank(b, R_i))$
  if $\exists b \in B$, such that $aGap < 0$ **then**
    $S_i \leftarrow R_i$, when a^* is ranked first
    revert from $backup[]$ all the $aGap$ and $cGap$
  if $s(c) = target$ **then**
    return $\mathcal{S}$
return failure

of c decreases, their score should also decrease. Moreover, any voter that decreases the score of c decreases the scores of the alternatives from B^h as well. There is no need to always decrease the scores of c and the score of the alternatives from B^h to the same extent; thus for each alternative $b \in B^h$ the algorithm assigns the variable $cGap(b)$, which represents the maximum difference allowed between the decrease in the score of c and b. However, whenever the $cGap$ of one of the alternatives reaches zero, every decrease in the score of c is executed to the same extent to the scores of the alternatives from B^h. To summarize, the algorithm ensures that the following three conditions are always true:
(1) The score of c does not go below $target$ (unless it was like that at the beginning and we are done).
(2) For any subset of voters, any alternative $b \in B^h$ is weaker than c.
(3) Any subset of voters does not increase the score of alternative $b \in B$ above $target$.
We thus attain:

Theorem 8. *If Algorithm 1 returns $\mathcal{S}$, it is a successful safe-persuasion for a^*.*

We conducted experiments to test the performance of our algorithm. We varied the number of alternatives and the number of voters. However, we did not use all of our randomly generated instances, since in some of them even a regular persuasion is not possible. Thus, we first used REVERSE to select the instances with a successful persuasion. We were able to find 900 instances with the uniform distribution, and 600 instances with the urn model. In each of these instances, the preferences of the sender were randomly chosen. We then used our algorithm to find a successful safe-persuasion. Figure 3 depicts our results, where the y-axis is the percentage of instances with a safe-persuasion (of the all the instances with a successful persuasion). Not surprisingly, in the uniform distribution the percentages were much higher. In addition, when we increased the number of voters and decreased the number of alternatives there were more voters in $\mathcal{P}(\mathcal{R}, c, a^*)$ to choose from and less alternatives in B, resulting in an increase in the ratio. In the urn model, the increase in the number of voters had a much less effect, since the votes were correlated.



(a) Uniform distribution (b) Urn distribution

Figure 3: Success ratio with SAFE-PERSUASION

7 Conclusions and Future Work

Settings in which a group of self-interested agents need to reach an agreement on joint actions are common nowadays. The ability of an automated agent to influence their decision is an important aspect in the field of social choice and economics. This paper investigated four different problems that address the persuasion process in such settings, both theoretically and empirically. One of the innovations presented in this paper concerns its context. We provide the motivation and demonstrate a real world application that can utilize the phenomena of manipulation, which is traditionally considered an artifact of election systems. We also explored the safety of persuasion which is of essence, especially when trying to persuade a group of self-interested agents when the outcome is uncertain. Motivated by the simulation results, in future research in this field we plan to focus on extending the analysis to other voting rules. Moreover we intend to provide additional heuristics for more voting rules.

Acknowledgments

This work was supported in part by ERC grant #267523, the Google Inter-university center for Electronic Markets and Auctions and MURI grant number W911NF-08-1-0144.

References

[Bell and Loftus, 1985] B. E. Bell and E. F. Loftus. Vivid persuasion in the courtroom. *Journal of Personality Assessment*, 18:1171–1192, 1985.

[Berg, 1985] S. Berg. Paradox of voting under an urn model: the effect of homogeneity. *Public Choice*, 47(2):377–387, 1985.

[Betzler *et al.*, 2011] N. Betzler, R. Niedermeier, and G.J. Woeginger. Unweighted coalitional manipulation under the borda rule is np-hard. In *Proceedings of IJCAI*, pages 55–60, 2011.

[Chamberlin, 1985] J.R. Chamberlin. An investigation into the relative manipulability of four voting systems. *Behavioral Science*, 30(4):195–203, 1985.

[Cobb and Kuklinski, 1997] M. D. Cobb and J. H. Kuklinski. Changing minds: Political arguments and political persuasion. *American Journal of Political Science*, 41(1):88–121, 1997.

[Conitzer and Sandholm, 2002] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of AAAI*, pages 314–319, 2002.

[Conitzer *et al.*, 2007] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):1–33, 2007.

[Davies *et al.*, 2011] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for borda manipulation. In *Proceedings of AAAI*, pages 657–662, 2011.

[Faliszewski *et al.*, 2006] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. The complexity of bribery in elections. In *Proceedings of AAAI*, pages 641–646, 2006.

[Faliszewski *et al.*, 2009] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. How hard is bribery in elections? *Journal of Artificial Intelligence Research*, 35:485–532, 2009.

[Hazon and Elkind, 2010] N. Hazon and E. Elkind. Complexity of safe strategic voting. In *Proceedings of the 3rd International Symposium on Algorithmic Game Theory (SAGT-10)*, pages 210–221, 2010.

[Lin, 2011] A. Lin. The complexity of manipulating k-approval elections. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence (ICAART-11)*, pages 212–218, 2011.

[Lupia, 1992] A. Lupia. Busy voters, agenda control, and the power of information. *The American Political Science Review*, 86(2):390–403, 1992.

[Nitzan, 1985] S. Nitzan. The vulnerability of point-voting schemes to preference variation and strategic manipulation. *Public Choice*, 47(2):349–370, 1985.

[Pritchard and Wilson, 2009] G. Pritchard and M.C. Wilson. Asymptotics of the minimum manipulating coalition size for positional voting rules under impartial culture behaviour. *Mathematical Social Sciences*, 58(1):35–57, 2009.

[Slinko and White, 2008] A. Slinko and S. White. Non-dictatorial social choice rules are safely manipulable. In *Proceedings of the 2nd International Workshop on Computational Social Choice (COMSOC-2008)*, pages 403–413, 2008.

[Tybout, 1978] A.M. Tybout. Relative effectiveness of three behavioral influence strategies as supplements to persuasion in a marketing context. *Journal of Marketing Research*, 15(2):229–242, 1978.

[Xia *et al.*, 2009] L. Xia, M. Zuckerman, A.D. Procaccia, V. Conitzer, and J.S. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proceedings of IJCAI*, pages 348–353, 2009.

[Xia, 2012] L. Xia. Computing the margin of victory for various voting rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 982–999, 2012.

[Zuckerman *et al.*, 2009] M. Zuckerman, A.D. Procaccia, and J.S. Rosenschein. Algorithms for the coalitional manipulation problem. *Artificial Intelligence*, 173(2):392–412, 2009.