# Bribery in Voting Over Combinatorial Domains Is Easy

**Nicholas Mattei**
Deptartment of Computer Science
University of Kentucky, USA
nick.mattei@uky.edu

**Maria Silvia Pini, Francesca Rossi,** and **K. Brent Venable**
Department of Pure and Applied Mathematics
University of Padova, Italy
{mpini,frossi,kvenable}@math.unipd.it

### Abstract

We investigate the computational complexity of finding optimal bribery and manipulation schemes in voting domains where the candidate set is the Cartesian product of a set of variables and agents' preferences are represented as compact CP-nets. We find that this change in the domain structure, which may lead to an exponential number of candidates in the size of the input, causes many existing computational results for bribery to break down. We provide new algorithms and complexity results which show that, in most cases, bribery in combinatorial domains is easy. This also holds for some cases of $k$-approval, where bribery is difficult in traditional domains.

## Introduction

We study the computational complexity of the bribery problem in voting domains where the set of candidates has a combinatorial structure and is exponentially large in the size of the input.

It is often natural to express group decision problems as the combination of a sequence of decisions. This method is used in many settings, from the United States Congress (specifically, votes for amendments to a bill) to a group of friends deciding what appetizer, main course, desert, and wine should be served for a group meal (Lang and Xia 2009; Brams, Kilgour, and Zwicker 1998). In all these cases, agents express preferences and vote on parts of the overall decision to be taken. Moreover, agents may have dependent preferences within this construction: the choice of wine may depend on the choice of main course for the meal. We consider a scenario where agents use the CP-net formalism. This allows the agents to compactly represent their preferences over many issues that may have conditional dependencies (Boutilier et al. 2004).

Bribery and manipulation directly question the security and robustness of an election system. They are ways that agents (outside or inside an election) can affect the election's outcome. In particular, the bribery problem is when an outside agent with a limited budget attempts to affect the outcome of an election by paying some of the agents to change their preferences. This problem was introduced to computational social choice by Faliszewski et al. (2009; 2008) and has become a metric by which the security of election rules is judged. If it is computationally difficult to find an optimal bribery in elections decided by a certain voting rule, we can say that the rule is resistant to bribery. Manipulation, instead, is when one or more voters attempt to vote strategically in order to affect the result of the election. Its complexity was first studied by Bartholdi et al. (1989) with further by Conitzer et al. (2007) and is another metric for election security. Both of these problems have been studied extensively in computational social choice. However, the literature is sparse when elections have combinatorially structured domains.

When voting is structured as the combination of several decisions, one natural method to determine a winner is to decide on an issue-by-issue basis, while the other natural approach is to aggregate the agents' votes over the set of all combinations of values of issues. In this paper we consider both approaches. In particular, we study elections via sequential (that is, issue-by-issue) majority (SM), plurality (OP), veto (OV), and $k$-approval (OK).

Unlike previous studies on bribery, CP-nets do not lend themselves to the convention that the cost of bribery is related to the number of swaps in the candidate ordering as proposed by Elkind et al. (2009). In fact, we don't want to work on the outcome ordering, since it is exponentially large, but on the CP-net. Since a single swap in a CP-net can lead to a large change in the outcome ordering, we consider three different bribery cost schemes: $C_{\text{EQUAL}}$ (any amount of change in a CP-net costs the same), $C_{\text{FLIP}}$ (the cost is the number of flips in the CP-net), and $C_{\text{LEVEL}}$ (the cost is the number of flips weighted by their position within the CP-net). Additionally, the use of acyclic CP-nets to denote preferences leads to instances where some voters, depending on their preference dependencies, cannot be bribed to vote for any arbitrary candidate. This restriction, along with the exponential number of candidates, breaks some of the existing algorithms for bribery and manipulation.

We show that bribery for OP and OV is always easy, while bribery for OK is easy when $k$ is a power of 2. This last result is particularly surprising at first sight, since bribery for $k$-approval in a non-combinatorial setting is difficult. However, this result is due to the fact that a profile of compatible acyclic CP-nets is not fully expressive, and there is a strong relationship among the preference orderings of such CP-nets. For SM, we prove that bribery is easy except when we use $C_{\text{EQUAL}}$. However, when voters are weighted, bribery

for SM is always difficult. We also provide results about manipulation and show that it is easy for SM and OP. Additionally, we provide results on bribery with non-binary issues. In this last setting, voting on a single issue can be performed by any rule (not just majority), so the complexity of both the bribery and the manipulation problem depends on the complexity of the voting rules used for the issues.

## Basic Notions

### CP-Nets

A major challenge for computational social choice is describing agent preferences in domains where there is a large number of candidates. AI provides several formalisms to do this including CP-nets (Boutilier et al. 2004), the one we consider in this paper.

CP-nets (Boutilier et al. 2004) are a graphical model for compactly representing conditional and qualitative preference relations. CP-nets are sets of *ceteris paribus* preference statements (cp-statements). For instance, the statement *"I prefer red wine to white wine if meat is served."* asserts that, given two meals that differ *only* in the kind of wine served *and* both containing meat, the meal with red wine is preferable to the meal with white wine.

Formally, a CP-net has a set of issues $F = \{x_1, \ldots, x_n\}$ with finite domains $\mathscr{D}(x_1), \ldots, \mathscr{D}(x_n)$. For each issue $x_i$, we are given a set of *parent* issues $Pa(x_i)$ that can affect the preferences over the values of $x_i$. This defines a *dependency graph* in which each node $x_i$ has $Pa(x_i)$ as its immediate predecessors. Given this structural information, preferences are specified over the values of $x_i$ for *each complete assignment* on $Pa(x_i)$. These preferences take the form of a total order over $\mathscr{D}(x_i)$. An *acyclic* CP-net has an acyclic graph.

Consider for example a CP-net whose issues are $A$, $B$, $C$, and $D$, with binary domains containing $f$ and $\overline{f}$ if $F$ is the name of the issue, and with the following cp-statements: $a \succ \overline{a}$, $b \succ \overline{b}$, $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) : c \succ \overline{c}$, $(a \wedge \overline{b}) \vee (\overline{a} \wedge b) : \overline{c} \succ c$, $c : d \succ \overline{d}$, $\overline{c} : \overline{d} \succ d$. Here, statement $a \succ \overline{a}$ represents the unconditional preference for $A = a$ over $A = \overline{a}$, while statement $c : d \succ \overline{d}$ states that $D = d$ is preferred to D=$\overline{d}$, given that $C = c$.

We say that a CP-net is *compact* if the maximum number of parents of a feature is bounded from above by a constant. This implies that the number of outcomes is exponential in the size of the CP-net.

The semantics of CP-nets depend on the notion of a worsening flip. Given a complete assignment of all issues, called an *outcome*, a *worsening flip* is a change in the value of an issue to a less preferred value according to the relevant cp-statement for that issue. For example, in the CP-net above, passing from $abcd$ to $ab\overline{c}d$ is a worsening flip since $c$ is better than $\overline{c}$ given $a$ and $b$. One outcome $\alpha$ is *better* than another outcome $\beta$ (written $\alpha \succ \beta$) if and only if there is a chain of worsening flips from $\alpha$ to $\beta$. This definition induces a preorder over the outcomes, which is a partial order if the dependency graph of the CP-net is acyclic.

In general, finding the optimal outcome of a CP-net is NP-hard (Boutilier et al. 2004). However, in acyclic CP-nets, there is only one optimal outcome, which can be found in linear time by sweeping through the CP-net following the directed arcs of its dependency graph and assigning the most preferred values in the cp-statements. For instance, in the CP-net above, we would choose $A = a$ and $B = b$, then $C = c$, and then $D = d$.

In an acyclic CP-net it is also easy, given an outcome, to find the next best outcome in a linearization of the induced outcome ordering (Brafman et al. 2010). Thus, finding the top $k$ solutions is easy if $k$ is bounded.

It should be noted that, given a set of issues, domains, and an outcome, there is always a CP-net that has such an outcome as the optimal one. However, given issues, domains, and an outcome ordering, there could be no CP-net whose induced outcome ordering coincides with the given ordering. In fact, CP-nets cannot model all outcome orderings. For example, two outcomes differing for the value of one issue must be necessarily ordered in the preorder induced by a CP-net.

### Voting

Voting theory (Arrow, Sen, and Suzumura 2002) is a wide research area that considers scenarios where a collection of voters (that we sometimes call agents) vote by expressing their preferences over a set of candidates (that we sometimes call outcomes), and a voting rule decides the winner. Voting theory provides many voting rules to aggregate agents' preferences. Each rule takes, as input, a partial or complete preference ordering of the agents and gives, as output, the winner (the outcome that is best according to the rule). If there are only two candidates, the best rule, according to many criteria, is majority voting (May 1952). When there are more than two candidates, there are many voting rules one could use (Plurality, Borda, STV, approval, etc.), each with its advantages and drawbacks. In this paper we consider the following rules:

**Plurality:** the candidate ranked in first place receives one point. The candidate(s) with the most points win the election. When there are two candidates, plurality coincides with majority.

**Veto:** Each voter chooses a candidate to veto. The candidate(s) with the least number of vetoes wins.

**k-approval:** Each voter approves of $k$ out of $m$ candidates ($k \leq m$) and disapproves of the remaining candidates. Candidate(s) with the most points win the election.

Voting theory provides an axiomatic characterization of voting rules in terms of desirable properties such as: the absence of a dictator; unanimity; anonymity; neutrality; monotonicity; independence of irrelevant alternatives; and resistance to manipulation and bribery. In this paper we focus on bribery, with some results for manipulation.

Given a profile, a briber is an outside agent that wants to affect the result of the election by convincing some agents to change their votes so that a preferred candidate wins (Faliszewski, Hemaspaandra, and Hemaspaandra 2009). The briber can convince agents by paying them and is usually subject to a limitation of its budget. Later in the paper we formally define the kind of bribery problem we consider.

Manipulation is a closely related problem to bribery. Manipulation occurs when a subset of the agents can get a better outcome by misreporting their preferences. More formally, in the *Constructive Coalitional Manipulation (CCM)* problem (Conitzer, Sandholm, and Lang 2007) we are given a set of agents X with fixed votes, a set Y of agents with unfixed votes, and a candidate $p$. We ask if there is an assignment of votes to the agents of Y such that running the election on the profile $X \cup Y$ results in $p$ winning the election. Certain forms of the manipulation problem can therefore be seen as bribery problems where the agents which can modify their vote are fixed and the budget is unlimited.

While every reasonable voting rule is manipulable (Arrow, Sen, and Suzumura 2002), a rule may be said to be resistant to manipulation if it is computationally difficult to decide how to manipulate. This has led to intensive studies of the computational properties of both bribery and manipulation for a variety of voting rules and information assumptions including manipulation results for CP-nets (Conitzer, Lang, and Xia 2009; Faliszewski, Hemaspaandra, and Hemaspaandra 2009; Conitzer, Sandholm, and Lang 2007; Xia and Conitzer 2010).

### Voting with CP-nets

The setting we consider consists of a set of $n$ agents expressing their preferences over a common set of candidates. The candidate set has a combinatorial structure: there is a common set of $m$ binary issues and the set of candidates is the Cartesian product of their domains. Each candidate is an assignment of values to all issues, thus we have $2^m$ candidates.

Each agent expresses its preferences over the candidates via a compact acyclic CP-net. Moreover, there exists a total ordering $O$ over the issues such that, in each CP-net, each issue must be independent of all issues following it in the ordering $O$.

A *profile* $(P, O)$ is a collection $P$ of $n$ CP-nets over $m$ common issues and a total ordering $O$ over the issues which satisfies the above property. This is called an $O$-legal profile by Lang (2007). Notice that the CP-nets appearing in such profiles do not necessarily have the same dependency graph.

To define a voting scheme over $O$-legal profiles Lang (2007) proposes a sequential approach where, at each step, agents vote over a single issue by using a chosen voting rule. This voting scheme is based on a total order over the issues that is compatible with the (acyclic) dependency graphs of the CP-nets: in each CP-net, each issue must be independent of all issues following it in the ordering. After each vote, a value is selected for the considered issue, and this choice is returned to all the agents who include this information in their CP-net. After as many steps as the number of issues, the winner outcome is built by collecting all the winning values for each issue. In this paper we consider a sequential approach but we also examine a one-step approach which computes the winner by considering all the CP-nets at once.

## Bribery Problem Definition

The bribery problem we consider has three parameters: the way a winner is chosen from the given profile, the allowed bribery actions, and the cost scheme for such actions.

### Winner Determination

In this paper we consider profiles consisting of CP-nets, as defined in Section . Such CP-nets have the same issues but possibly different dependency graphs and cp-statements. We assume that all issues are binary. To determine the winner outcome, we consider two approaches:

**Sequential Majority (SM):** This is basically Lang's sequential procedure (Lang 2007) when instantiated on binary issues. We vote for each issue in the ordering $O$ using majority (since issues are binary). The value chosen for an issue is then returned to all agents who then fix the issue to that value in their respective CP-nets. When voting has completed on all issues, the winner outcome consists of the issue instantiation chosen by majority at each of the voting steps.

**One-step Plurality (OP):** We ask each agent to provide the optimal outcome in his CP-net and we use plurality to decide which of these outcomes is the winner.

We will also consider other variants of the second method, where instead of plurality we use the veto rule and the $k$-approval rule. We denote them by OV (for One-step Veto) and OK (for One-step $k$-approval). Using these voting rules means that agents needs to provide the worst outcome and the top $k$ outcomes, respectively.

It is worthwhile to clarify our notion of an agent: an agent is only a CP-net handler. He has the ability to specify a CP-net over a given set of binary issues and to compute the optimal outcome, the top $k$ outcomes, and the worst outcome of his CP-net. He does not handle the explicit partial order over all the outcomes of the CP-net, but just the CP-net, which is a compact representation of the ordering.

### Bribery Actions

In most bribery frameworks agents express their preferences over the candidates via a total order, and the briber asks agents to make any changes within this total order. In our domain, agents have a CP-net instead of an explicit outcome ordering. If the briber could ask for any change in the ordering induced by the CP-net, some of these changes could be computationally hard or impossible to accomplish via changes in the CP-net. Therefore, we define the bribery actions as changes made directly to the cp-statements within the CP-net of an agent. Since a cp-statement gives a total order for the domain of an issue the briber asks for a new total order over the domain. In this paper we consider binary issues, so changing a cp-statement means flipping the positions of the two values of an issue.

In a CP-net, a cp-statement is associated to a certain issue, and issues are of two kinds: independent and dependent. We distinguish bribery actions on these two kinds of issues:

**Independent Variable Bribery (IV):** The briber sets a new total order in a cp-statement related to an independent issue.

**Dependent Variable Bribery (DV):** The briber sets a new total order in a cp-statement related to a dependent issue.

We will also consider the combination of both such bribery actions, that we will denote by IV+DV. Notice that with these bribery actions no dependency can be added in

the CP-nets. Thus, the new CP-nets are still compatible with one another and with the ordering $O$ over the issues.

## Cost Schemes

In classical bribery domains, the pricing of the bribery actions is either fixed (that is, the same price for any new total order), or it depends on the number of swaps to be performed on the current total order to obtain the desired one. We believe that this second approach, the swap bribery approach introduced by Elkind et al. (2009), is a better model of the amount of resources needed to achieve the change asked for by the briber. In one of our cost schemes we use the swap model, although swaps are in the cp-statements and not directly on the outcome ordering.

When preferences are modeled via CP-nets, the structure of the CP-net should come into play. To define this more precisely, let us partition the issues of a CP-net into levels, which we define recursively as:

$$level(x) = \begin{cases} 1 & \text{If } x \text{ is an independent issue.} \\ i+1 & \text{If all parents of } x \text{ are in levels } \{1,\ldots,i\} \\ & \text{and there is a parent in level } i. \end{cases} \quad (1)$$

It is possible to see that changing a cp-statement associated to an issue in a lower level causes a larger overall change in the outcome ordering. Therefore, such a change should cost more. For example, consider a CP-net with two issues, A and B, where B depends on A, and with the following cp-statements: $a \succ \overline{a}$, $a : b \succ \overline{b}$, $\overline{a} : \overline{b} \succ b$. The outcome order induced by this CP-net is: $ab > a\overline{b} > \overline{a}\overline{b} > \overline{a}b$. If we change the cp-statement over A to $\overline{a} \succ a$, we get $\overline{a}\overline{b} > \overline{a}b > ab > a\overline{b}$. If we measure the difference between two orderings by the number of outcome swaps needed to pass from one ordering to the other then in this example the difference is 4. If instead we change the second cp-statement over B to $\overline{a} : b \succ \overline{b}$, thus making B independent of A, we get a partial order with $ab$ at the top, $\overline{a}\overline{b}$ at the bottom, and the other two outcomes incomparable in the middle. Since this is a partial order, we count the difference with the original ordering by the maximum number of outcome swaps to obtain a linearization of the partial order. In this case, 2.

In this paper we consider three cost schemes, starting from the simplest one (that closely mimics traditional bribery) and moving towards more complex bribery pricing schemes that take into consideration the CP-net formalism.

**$C_{\text{EQUAL}}$**: A unit cost allows to make any number of changes in a CP-net.

**$C_{\text{FLIP}}$**: The cost of changing a CP-net is the number of flips performed in total in its cp-statements.

**$C_{\text{LEVEL}}$**: The cost of changing a CP-net is the number of flips performed in total in its cp-statements, each weighted according to the level of the relevant issue. More precisely: $\sum_x flip(x) \times (k+1-level(x))$, where $x$ ranges over the issues and k is the number of levels in the CP-net, and $flip(x)$ is the number of flips performed in cp-statements associated to $x$. We expect the weights to be greater at the lower (more influential) levels (Boutilier, Bacchus, and Brafman 2001).

Each of the above cost schemes can be generalized to account for agents with different bribing costs by associating a certain cost to each agent and multiplying the cost of the bribery actions with the agent's cost. In the following we only consider the generalized versions of the cost schemes. We denote with $Q_i \in \mathbb{Z}^+$ the bribing cost of agent $i$.

In many variations of both the bribery and manipulation problems it is standard to attach weights to the voters (Faliszewski, Hemaspaandra, and Hemaspaandra 2009; Conitzer, Sandholm, and Lang 2007). These weights are usually regarded as either representing a group of voters who all think alike or instances, such as shareholder votes for publically traded companies, where not all voters are considered equal. We assume that each agent $i \in n$ is assigned a weight, $w_i \in \mathbb{Z}^+$, that represents the weight of that agent.

## Combinatorial Bribery

We now formally state the bribery problem:

**Name:** ($D$,$A$,$C$)-Bribery.

**We are given:**

–A profile $(P,O)$ where $P$ is a collection of $n$ compact CP-nets with $m$ binary issues and $O$ is a total ordering of the $m$ issues;

–a budget $B \in \mathbb{Z}^+$;

–an outcome $p$, that is a complete assignment to all $m$ issues;

–and bribing cost vector $\vec{Q}$.

**Question:** Is there a way for an outside actor to make $p$ win in profile $(P,O)$ with winner determination rule $D \in \{SM, OP, OV, OK\}$, by using bribery actions according to $A \in \{IV, DV, IV+DV\}$, and by paying according to scheme $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}\}$ and bribing cost vector $\vec{Q}$, without exceeding $B$?

We now study the computational complexity of this problem, considering all possible combinations of winner determination rules, bribery actions, and cost schemes.

In this paper we assume, as in other works on bribery such as Elkind et al. (2009) and Faliszewski et al. (2009), a non-unique winner model; we are only asked to elevate $p$ into the winning set. Some of our reductions can be extended to any unique winner model but we focus on the general case.

# Winner Determination Is Easy

The computational complexity of the bribery problem is interesting only when it is easy to determine the winner of an election, since otherwise bribery would be trivially hard. We thus show that this is the case for all the winner determination approaches we consider.

**Theorem 1** *Winner determination is in* P *for SM, OP, OV, and OK (when k is bounded).*

**Proof.** Sequential Majority (SM) works issue-by-issue, and at each issue it applies the majority voting rule. As there are a limited number of issues, the overall procedure is in P.

For One-step Plurality (OP), we need to compute the optimal outcome for each CP-net, which is polynomial since we

are dealing with acyclic CP-nets. Then, plurality is applied, and this again is a polynomial step.

The winner according to One-step Veto (OV) can be achieved by reversing each total order in all cp-statements of the CP-nets. In fact, by doing this, the optimal outcome of the new CP-net is the worst outcome of the original one. Notice that in an acyclic CP-net we have exactly one worst outcome. This reversal step takes polynomial time in the size of the CP-nets since they are compact. After the reversal step, we just choose an outcome which appears the smallest number of times (possibly zero) as the optimal outcome of the reversed CP-nets. This again takes polynomial time with respect to the size of the input.

One-step $k$-approval (OK) needs to get the top $k$ outcomes from each CP-net. If $k$ is bounded, this is easy since the CP-nets are acyclic (Brafman et al. 2010). Therefore, providing the ballots is polynomial, as is applying $k$-approval. ❑

## Changing a Vote Is Easy

In the classical bribery scenario, the cost of bribing a single agent so that he votes for another candidate is straightforward to compute since agents specify their preferences explicitly In our scenario, the use of CP-nets makes this computation not as trivial. However, we show that it is still easy to do, no matter which bribery actions or cost schemes we use. The main difference is that it is sometimes impossible to achieve the desired vote change because of restrictions on the bribery actions.

**Theorem 2** *Given a CP-net and an outcome p, determining if the CP-net can be changed to make p its optimal outcome, and, if so, determining the minimum price to perform such a change, can be computed in polynomial time.*

**Proof.** Assume we can use all bribery actions, $IV + DV$, therefore, any CP-net can be changed to a vote for $p$.

With $C_{\text{EQUAL}}$, it is trivial to compute the cost, as it does not depend on changes to the CP-net: if $p$ is already the top outcome of the CP-net, the cost is 0, otherwise it is 1.

With $C_{\text{FLIP}}$, we need to compute the minimum number of flips to be done on the CP-net. There are some flips that are necessary to make $p$ the top outcome. In order to find them, we start from the independent issues and we flip their cp-statements which do not have the corresponding value of $p$ as their preferred value. We then proceed to each dependent variable, as soon as its parents are processed, and do the same in its relevant cp-statement. At the end, $p$ is the top outcome in the resulting CP-net. The number of flips performed gives us the minimum price to make $p$ win. If we use $C_{\text{LEVEL}}$, the flips are the same as above but the total cost takes into account the levels.

Notice that this algorithm gives us the minimum set of flips, but the same result could be achieved also by performing other useless flips. However, what we are interested in is computing the minimum cost to change the CP-net.

If we are only allowing one kind of bribery actions, IV or DV, we may stop the procedure as soon as a required flip cannot be done, since this means that $p$ cannot be the optimal outcome for the CP-net with the allowed change actions. ❑

## Sequential Majority

In this section we consider the computational complexity of bribery when we determine the winner via sequential majority. The first result shows that, when we use the cost schemes $C_{\text{FLIP}}$ or $C_{\text{LEVEL}}$, the bribery problem is computationally easy.

**Theorem 3** *(SM,IV,C)-Bribery, (SM,DV,C)-Bribery, and (SM,IV+DV,C)-Bribery are in* P *when* $C \in \{C_{\text{FLIP}}, C_{\text{LEVEL}}\}$.

**Proof.** We first show a polynomial time algorithm for solving $(SM, IV + DV, C_{\text{FLIP}})$-Bribery.

Given a profile $(P, O)$, we consider the issues according to the ordering given by $O$. Beginning with the first issue in an ordering, we can compute the minimum number of flips to be performed in the agents' CP-nets in order to achieve a majority for the same value that appears in $p$ for that issue. Since we are using SM, we just need to select the agents to bribe by starting from the cheapest ones (according to $\vec{Q}$), until we achieve a majority for the considered issue.

In fact, by SM, the resulting preferred value for this issue will be propagated in all CP-nets (not just those where this value was most preferred). In the next level, and in all subsequent levels, we perform the same computation. At the end, when all issues have been considered, the overall number of flips (required to get the correct majority for each issue) will determine the cost of bribing so that $p$ wins. The total cost is $\leq B$ if and only if the bribery problem has a solution.

If we use the cost scheme $C_{\text{LEVEL}}$ instead of $C_{\text{FLIP}}$, the choice of which agents to bribe for each issue is dictated not only by the bribing cost vector, but also by the level of the issue in the CP-nets. Notice that the same issue could appear in different levels in the agents' CP-nets.

A very similar algorithm can also be used for IV and DV bribery actions with the restrictions on which cp-statements can be flipped. With IV, for each issue, we are allowed to perform a flip only in those CP-nets that have the particular issue as independent. With DV, we can only flip issues in CP-nets that have the particular issue as dependent. ❑

The above theorem considers only the costs schemes $C_{\text{FLIP}}$ and $C_{\text{LEVEL}}$. The reason is that, with these cost schemes, we compute the cost by counting the number of flips (either with a weight or not), so the flips we perform for one issue do not pose any restriction to the flips we perform for another issue. With $C_{\text{EQUAL}}$, instead, the cost for changing a CP-net is always 1, no matter how many flips are needed. Therefore, the decision on which CP-nets to change for a particular issue is strictly related to this same decision for another issue: if we change the same CP-nets, the cost will not increase. So, to compute the minimum bribery cost, the computation we do for one issue depends on the computation for a different issue. This makes the bribery problem computationally hard, as the following theorem formally shows.

Before stating and proving the theorem, we need to consider the OPTIMAL LOBBYING (OL) problem (Christian et al. 2007).

**We are given:** An $n \times m$ $0/1$ matrix $E$ and a $0/1$ vector $\vec{x}$ of length $m$ where each column of $E$ represents an issue and

each row of $E$ represents a voter. $E$ is a binary approval matrix with 1 corresponding to a "yes" vote and $\vec{x}$ is the target group decision.

**Parameter:** A positive integer, $k$: the number of agents to be influenced.

**Question:** Is there a choice of $k$ rows of the matrix $E$ such that these rows can be edited so that the majority of votes in each column matches the target vector $\vec{x}$?

This problem is shown to be $W[2]$-complete (Christian et al. 2007). We give a polynomial reduction from OL to our bribery problem, thus showing that it is NP-complete (Downey and Fellows 1999). In fact, since OL is $W[2]$-complete for parameter $k$, all the bribery problems in the following proof are $W[2]$-hard with parameter $B$.

**Theorem 4** *$(SM,IV,C_{\text{EQUAL}})$-Bribery, $(SM,DV,C_{\text{EQUAL}})$-Bribery, and $(SM,IV+DV,C_{\text{EQUAL}})$-Bribery are* NP*-complete.*

**Proof.** For all bribery problems, membership in NP is an immediate consequence of Theorem 1 and a guess and check algorithm. To show completeness, we provide polynomial reductions from OL. We start with $(SM,IV,C_{\text{EQUAL}})$-Bribery.

Given an instance $(E,\vec{x},k)$ of OL, we can construct an instance of $(SM,IV,C_{\text{EQUAL}})$-Bribery containing CP-nets with only independent variables. The set of issues, $m$, is equal to the number of columns in $E$. For each row of $E$, we create a voter with the preferences over the $m$ variables as described in the row of $E$. Finally, we set the price of bribery for each voter to be 1, the budget $B = k$, the weights of the agents all equal, and the preferred outcome $p = \vec{x}$.

Thus, $p$ wins the election if and only if there is a selection of $k$ rows of $E$ such that $\vec{x}$ becomes the winning agenda of the OL instance. Therefore $(SM,IV,C_{\text{EQUAL}})$-Bribery is NP-complete. The same reduction works for IV+DV.

For DV, we choose an instance of the bribery problem as above, except that there is one more issue and each voter therefore has an additional independent variable on which all others depend. The preference of all voters on the new variable is 1¿0. For the other variables, the corresponding row of E will provide the preference associated to the value 1 of the new variable, while for the value 0 we give the opposite preference. The last difference is that now $p$ is $1\vec{x}$. With this mapping, $p$ wins the election if and only if the given OL instance has a positive answer. ❑

We now turn our attention to weighted agents. The winner of the election will be computed by a weighted majority for each issue. We prove that bribery with the $C_{\text{LEVEL}}$ and $C_{\text{EQUAL}}$ cost schemes is computationally difficult, while it is easy with $C_{\text{FLIP}}$. We denote by weighted-X the bribery problem X with weighted agents.

Faliszewski et al. show NP-completeness of a problem called plurality-weighted-$bribery (Faliszewski, Hemaspaandra, and Hemaspaandra 2009). In such a problem each candidate $c_i$ is associated with a price $\$_i \in \mathbb{Z}^+$ and weight $w_i \in \mathbb{Z}^+$, the voting rule used is plurality. The problem is to determine if a favorite candidate $x$ can be made a winner by bribing voters within a budget $k$. In fact, Faliszewski et al. show this problem to be NP-complete for only two candidates; we will use this result in the following theorem.

**Theorem 5** *Weighted-(SM,A,C)-Bribery where $A \in \{IV, IV+DV\}$ and $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}\}$ is* NP*-complete.*

**Proof.** Membership in NP is an immediate consequence of Theorem 1 and a guess and check algorithm. We can reduce the plurality-weighted-$bribery problem over two candidates to an instance of our problem where we have the same number of voters; the same costs and weights; and the same budget as the original problem. Moreover, there is only one issue and, thus, the voters have a single independent variable with two values corresponding to the two candidates, one of which, say $p$, corresponds to $x$. On a profile of this kind all the cost schemes act the same and so do the bribery actions $IV$ and $IV+DV$. ❑

The following theorem follows, we omit the proof.

**Theorem 6** *Weighted-(SM,DV,C)-Bribery where $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}\}$ is* NP*-complete.*

However, if we use the cost scheme $C_{\text{FLIP}}$ and we assume that the bribing costs are the same for all voters, the bribery problem becomes easy.

**Theorem 7** *Weighted-(SM,IV,$C_{\text{FLIP}}$)-Bribery, weighted-(SM,DV,$C_{\text{FLIP}}$)-Bribery, and weighted-(SM,IV+DV,$C_{\text{FLIP}}$)-Bribery are in* P *when the bribing costs in $\vec{Q}$ are all the same.*

**Proof.** We can use the same argument as in the proof of Theorem 3. Since voters are weighted, we only need to achieve a weighted majority for each individual issue. Since all the bribing costs at each level are the same with the $C_{\text{FLIP}}$ pricing function, the cost to bribe any voter at any single level is equal for all voters. We can therefore bribe the voters from heaviest to lightest (in terms of their weight). This will be an optimal sequence of bribes at each individual level and lead to the optimal bribery scheme overall.

This is true no matter which bribery actions are allowed, since using IV or DV just restricts the issues over which it is possible to perform a flip in the CP-nets. ❑

## One-step Plurality

The model presented in this paper with cost function $C_{\text{FLIP}}$ and $C_{\text{LEVEL}}$ is similar to the nonuniform bribery model presented by Faliszewski (Faliszewski 2008). Faliszewski shows that bribery in single issue elections with nonuniform cost functions is in P through the use of flow networks. The algorithm requires the enumeration of all possible elements of the candidate set as part of the construction of the flow network. In our model, the number of candidates is exponential in the size of the input, so we cannot use that construction directly. However, we show that a similar technique to that of Faliszewski can be used while not requiring the enumeration of an exponential number of candidates (Faliszewski 2008).

**Theorem 8** *(OP,IV,$C_{\text{FLIP}}$)-Bribery is in* P *when all variables are independent in all CP-nets of the profile.*

**Proof.** If the number of candidates, which is $2^m$, is polynomial in the number of voters ($n$), we can the proof by Faliszewski (Faliszewski 2008). So we assume that we have a number of candidates such that $2^m > 2n$.

For a candidate $v$ and a positive integer $d$, let $neb(v,d)$ be the set of candidates that can be obtained starting from $v$ by changing $d$ issues. Let $G$ be the set of candidates who receive 1 or more votes. Let $score(v)$ denote the number of votes for candidate $v$. Note that the number of candidates with nonzero scores is at most $n$.

For all $c \in neb(p,1) \cap G$ at cost $|neb(p,1) \cap G|$, we can safely change them to votes for $p$. If, after this step we have $\forall g \in G : score(g) \leq score(p)$ and $|neb(p,1) \cap G| \leq B$ then we accept. Likewise, if $B > (\frac{n}{2} - 1) \cdot m$, then we can afford bribe half the voters to vote for $p$ and we accept.

We now consider all $r \in \{1,\ldots,n\}$ and ask if $p$ can be made a winner with exactly $r$ votes without exceeding $B$. If there is at least one $r$ such that this is possible, then we accept, otherwise we reject. We show that, for each $r$, the corresponding decision problem can be solved in polynomial time. This means that the overall bribery problem is in P.

To solve the decision problem for a certain $r$, we reduce this problem to a minimum-cost flow problem (Ahuja, Magnanti, and Orlin 1993). In detail, we build a network with a source $s$, a sink $t$, and three "layers" of nodes.

The first layer of nodes has one node for each voter $v_1,\ldots,v_n$. We also add $n$ edges $(s,v_i)$, all with capacity 1 and cost 0.

The second layer of nodes represents a subset of the candidates. In order to define the subset, we need to introduce the notion of a *safe* candidate. A candidate $g$ will be considered safe if $score(g) \leq score(p) - 1$. For each voter not voting for $p$ in the given profile, we add $n$ nodes corresponding to its top outcome and additional $n-1$ safe candidates that are closest to the optimal outcome for that voter. Notice that we are sure there are at least $n-1$ safe candidates, since $2^m > 2n$ and at most $n-1$ candidates have some vote.

To find such safe candidates, we exploit the voter's CP-net by visiting its outcome ordering starting from the top outcome and moving down in one linearization of the ordering, skipping those candidates that are not safe. Since the CP-nets are acyclic, each step is polynomial (Brafman et al. 2010). Moreover, at most we need to perform $2n-1$ steps to find $n-1$ safe candidates, since between any two safe candidates there can be at most $n-1$ unsafe ones.

To this set of safe candidates we add, edges so that each voter can also vote for $p$ and the $n$ candidates who currently receive votes. Therefore, each voter can change their existing vote to $p$ (always safe). This second layer models the profile modified by the bribery: each voter can change its vote or also maintain the previous one.

For each node in the second layer, say $S_{ij}$, corresponding to voter $v_i$, we add an edge from $v_i$ to $S_{ij}$ with capacity $+\infty$ and cost equal to the cost to bribe $v_i$ to vote for the candidate corresponding to node $S_{ij}$.

In the third layer of the network, we add a node for each candidate who already appears somewhere in the network (up to $n^2 - n$). One of these nodes, say $f_1$, represents $p$. These will be the nodes that enforce the constraint that every candidate besides $p$ cannot receive more than $r$ votes. These nodes will have an edge from the nodes of the second layer representing the same candidate, where the edge has zero cost and infinite capacity. The output link from each node in the third layer to the sink has capacity $r$. The cost is 0 for the edge from $f_1$ to the sink, while it is a integer $M$ higher than any possible bribery for all other nodes of the third layer.

Notice that in the second layer we just add, for each voter, nodes corresponding to $n$ candidates, so we do not allow voters to move their vote to a candidate which is not safe and farther away from its top candidate in terms of cost. This allows us to avoid an exponential number of nodes.

If we had included nodes for all the candidates in the second layer, we would have used a network equivalent to the one used in the proof of Theorem 3.1 in (Faliszewski 2008), which shows that there is a minimum cost flow of value $n$ if and only if there is a way to solve the bribing problem. However, since we have a number of candidates which is exponential in the size of the input, we would not have a polynomial algorithm.

However, by including just the cheapest $n$ alternative candidates for each voter such a result still holds. In fact, assume there is a minimal flow in the larger network which goes through one of the added nodes. This means that a voter has been forced to vote for another candidate since all its top $n$ safe candidates had already $r$ votes each. However, this is not possible since we have only a total of $n-1$ votes that can be given by the other voters.

We will build, at worst, $n$ networks with $2n^2+n+2$ nodes and $2n^2+2n$ edges. We can solve a min cost feasible flow problem in time $\mathcal{O}(|V||E|\log(|V|))$. Therefore the overall running time of this method is polynomial. ❑

This leads to the following corollaries (proofs omitted).

**Corollary 9** *(OP,IV,C)-Bribery is in* P *for* $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}\}$.

**Corollary 10** *(OP,DV,C)-Bribery and (OP,IV+DV,C)-Bribery is in* P *when* $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}\}$.

## One-step Veto Rule

The following theorem follows a similar construction to that of Theorem 8. We omit the full proof.

**Theorem 11** *(OV,IV,C)-Bribery, (OV,DV,C)-Bribery and (OV, IV+DV,C)-Bribery is in* P *when* $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}\}$.

## One-step $k$-approval

In One-step $k$-approval (OK) we aggregate the $n$ profiles by taking the top $k$ outcomes from each agents' profile. Since CP-nets induce a partial order we mean the top $k$ according to some linearization. In the traditional bribery domain, $k$-approval, when $k \geq 3$, is NP-complete when all the bribery costs are equal (Faliszewski, Hemaspaandra, and Hemaspaandra 2009). However, when agents' preferences are expressed as CP-nets, bribery schemes for $k$-approval can be computed in polynomial time under certain conditions. We require the following lemma:

**Lemma 12** *Consider acyclic CP-nets and $k = 2^j$. Then the top $k$ outcomes of a CP-net are all the outcomes differing from the top one on the value of $j$ issues. Moreover, given two CP-nets in the same profile and with the same top element, they have the same top $k$ elements.*

**Proof.** Given a CP-net, consider any order of its issues which is compatible with its dependency graph, say $x_1 > \cdots > x_i > x_{i+1} > \cdots > x_m$. For a given top outcome (assignment to variables $x_1, \ldots, x_m$) the next outcome will be the one with $\bar{x_m}$ and variables $x_1, \ldots, x_{m-1}$ maintaining their assignments. This property is expandable up to any power of two. If $k = 2^j$, then the top $k$ outcomes have the same values for issues $x_1, \ldots, x_{m-j}$ and differ on the values of issues $x_{m-j+1}, \ldots, x_m$. Since we can use the same topological order for any two CP-nets in a profile, if two CP-nets have the same top outcome then they have the same top $k$ outcomes when $k$ is a power of 2. ❑

**Theorem 13** *(OK,IV,$C_{\text{FLIP}}$)-Bribery is in* P *when* $k = 2^j$.

**Proof.** When $k$ is a power of two, Lemma 12 tells us that, for a given top candidate, the rest of the candidates must follow in some order. Therefore, we treat the top $k$ outcomes as one bundle. This ends up restricting the number of candidates each voter can approve of into certain bundles.

Now that we can treat the top $k$ outcomes as a single item we can apply the algorithm described in the proof of in Theorem 8 in order to decide the cheapest bribery scheme to elevate the group that includes $p$ into the winning set. For each voter we will need to find $n$ safe groups and this can be done in polynomial time. ❑

Similar to Theorem 8, we state the following:

**Corollary 14** *(OK,A,C)-Bribery where* $A \in \{IV, DV, IV + DV\}$ *and* $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}\}$ *is in* P *when* $k = 2^j$.

A practical use of this result is for the elicitation problem for CP-nets: if the ordering over the issues is known, then we only need to elicit the top candidate in order to know a voters' top $k$ candidates, when $k$ is a power of 2.

## Manipulation and Non-binary Domains

We will now show that, in the setting we have considered, manipulation is easy for two of the voting procedures. We omit the proof for space.

**Theorem 15** *Weighted-CCM and unweighted-CCM are in* P *for SM and OP.*

Up to this point we have required that all variables in the domain be binary. While this is a convenient and highly expressive model, we wish to relax this assumption. This allows the sequential composition of general voting rules (not just majority) as in other work on voting in sequential domains (Lang and Xia 2009).

In what follows, each issue has a domain with possibly more than two values and the CP tables specify strict total orderings on such domains. The briber will modify the linear orders, much like in the swap bribery domain described by Elkind et al. (Elkind, Faliszewski, and Slinko 2009).

Now that we can use different voting rules we will associate to a profile $(P, O)$ a vector $R$ of voting rules $< r_1, \ldots, r_m >$. At level $i$ we use voting rule $r_i$ to aggregate the preferences at that level and propagate the winning outcome at that level to the next level for all agents' CP-nets. Winner determination in this domain is polynomial if evaluation of the underlying voting rules are polynomial procedures.With this model we can make two observations about sequential voting rules:

| | SM | SMw | OP | OV | OK |
|---|---|---|---|---|---|
| $C_{\text{EQUAL}}$ | NP-c | NP-c | P | P | P* |
| $C_{\text{FLIP}}$ | P | NP-c | P | P | P* |
| $C_{\text{LEVEL}}$ | P | NP-c | P | P | P* |

Table 1: Complexity results about bribery presented in this paper. SMw stands for SM with weighted voters, P* stands for P when k is a power of 2, NP-c stands for NP-complete.

**Theorem 16** *If bribery or CCM is hard for any rule in* $< r_1, \ldots, r_m >$ *then bribery is hard for the sequential rule.*

This property can be shown by providing a reduction from the single issue bribery or CCM problem to the multi-issue setting via the construction of an appropriate profile and by exploiting the fact that CCM is a special case of bribery.

**Theorem 17** *If bribery or CCM is easy for every rule in* $< r_1, \ldots, r_m >$ *then bribery is easy for the sequential rule.*

**Proof.** This follows from the separability property shown in Theorem 3. Since the levels of the ordering $O$ can be treated independently the problem separates into a sequence of independent bribery problems. If each one of these subproblems are computationally easy, the entire sequence is computationally easy. ❑

## Conclusions and Future Work

We have investigated the computational complexity of bribery and manipulation schemes in voting scenarios with a combinatorial structure over the set of candidates and votes expressed via CP-nets. We generalized the traditional bribery problem to encompass these domains. To this aim, we considered four election rules, three bribery methods, and three cost structures. For most of the combinations of these parameters, bribery in this domain is computationally easy. Our results are summarized in Table 1.

We have also shown several interesting properties of acyclic CP-nets that can be leveraged for computational and preference elicitation reasons. In fact, when agents express their preferences as acyclic compatible CP-nets, there are cases of the bribery problem that become computationally *easier* than the single issue domains, since such CP-nets have a restricted expressive power in terms of the orderings they can induce.

We would like to generalize our work in order to study other scoring and voting rules. We are also interested in considering additional bribery actions: in this paper, the briber can only delete dependencies, while we would like to investigate the complexity of the bribery problem when the outside agent is also allowed to add dependencies within the CP-net.

**Acknowledgements** We are grateful for the support of NSF IIS-1107011 for funding the initial research visit to start this work as part of the IJCAI 2011 Doctoral Consortium. Nicholas Mattei is also supported by NSF EAGER grant CCF-1049360.

## References


Ahuja, R.; Magnanti, T.; and Orlin, J. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

Arrow, K. J.; Sen, A. K.; and Suzumura, K. 2002. *Handbook of Social Choice and Welfare.* North-Holland, Elsevier.

Bartholdi, J.; Tovey, C.; and Trick, M. 1989. The computational difficulty of manipulating an election. *Social Choice and Welfare* 6(3):227–241.

Boutilier, C.; Bacchus, F.; and Brafman, R. 2001. Ucp-networks: A directed graphical representation of conditional utilities. In *Proc. UAI 2001*, 56–64.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *JAIR* 21(1):135–191.

Brafman, R. I.; Rossi, F.; Salvagnin, D.; Venable, K. B.; and Walsh, T. 2010. Finding the next solution in constraint- and preference-based knowledge representation formalisms. In *KR 2010*. AAAI Press.

Brams, S.; Kilgour, D.; and Zwicker, W. 1998. The paradox of multiple elections. *Social Choice and Welfare* 15(2):211–236.

Christian, R.; Fellows, M.; Rosamond, F.; and Slinko, A. 2007. On complexity of lobbying in multiple referenda. *Review of Economic Design* 11(3):217–224.

Conitzer, V.; Lang, J.; and Xia, L. 2009. How hard is it to control sequential elections via the agenda. In *IJCAI09*, 103–108.

Conitzer, V.; Sandholm, T.; and Lang, J. 2007. When are elections with few candidates hard to manipulate? *JACM* 54(3):1–33.

Downey, R., and Fellows, M. 1999. *Parameterized Complexity*. Springer-Verlag.

Elkind, E.; Faliszewski, P.; and Slinko, A. 2009. Swap bribery. *Algorithmic Game Theory* 299–310.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. 2009. How hard is bribery in elections? *JAIR* 35:485–532.

Faliszewski, P. 2008. Nonuniform bribery. In *AAMAS08*, 1569–1572.

Lang, J., and Xia, L. 2009. Sequential composition of voting rules in multi-issue domains. *Mathematical Social Sciences* 57(3):304–324.

Lang, J. 2007. Vote and aggregation in combinatorial domains with structured preferences. In *IJCAI07*, 1366–1371.

May, K. 1952. A set of independent necessary and sufficient conditions for simple majority decisions. *Econometrica* 20,4:680–684.

Xia, L., and Conitzer, V. 2010. Strategy-proof voting rules over multi-issue domains with restricted preferences. *Internet and Network Economics* 402–414.