Modeling influence in multi-agent preference aggregation over combinatorial domains

Alberto Maran[1], Nicolas Maudet[2], Maria Silvia Pini[1], Francesca Rossi[1], and K. Brent Venable[3]

1: University of Padova, Italy, email: amaran@studenti.math.unipd.it, pini@dei.unipd.it, frossi@math.unipd.it

2: LIP6 and UPMC, France, email: nicolas.maudet@lip6.fr

3: Tulane University and IHMC, USA, email: bvenable@ihmc.us.

Abstract. We consider multi-agent settings where a set of agents want to take a collective decision, based on their preferences over the possible candidate options. While agents have their initial inclination, they may interact and influence each other, and therefore modify their preferences, until hopefully they reach a stable state and declare their final inclination. At that point, a voting rule is used to aggregate the agents' preferences and generate the collective decision. Recent work has modeled the influence phenomenon in the case of voting over a single issue. Here we generalize this model to account for preferences over combinatorially structured domains including several issues. We propose a way to model influence and to aggregate preferences, by interleaving voting and influence convergence, when agents express their preferences as CP-nets. We also provide results about the resistance of this setting to bribery and manipulation.

1 Introduction

In a multi-agent context where a set of agents declares their preferences over a common set of candidates, it is often the case that such agents interact and exchange information before voting.

In this paper, we assume that the information agents exchange is the mere observation of others' votes. Agents may revise their vote on the basis of the observed votes of others. This occurs, for example, in political elections, where an agent may be influenced by the opinion of esteemed people. Another possible interpretation of this setting, which may fit different situations, is that it offers a model of inter-agent conditional behaviour, without necessarily modifying their preferences.

The concept of influence has been widely studied in psychology, economics, sociology, and mathematics [3, 4, 10]. Recent work [8] has modelled the influence phenomenon as an iterative process, where agents may change their inclination more than once based on the changes in the preferences of other influential agents. Some influence schemes may converge, while others may loop forever.

Also manipulation and bribery in voting may be regarded as a type of influence, although they does not involve an iterative process. In so-called *bribery* problems [5], an external agent (the briber) has typically a limited budget he can spend to modify the vote of other agents. In *manipulation* [14], one or more voting agents misreport their preferences in order to get a more preferred collective decision.

We define a framework for modeling multi-agent preferences and inter-agent influences. In our setting, a set of agents needs to select a common decision from a set of possible decisions, over which they express their preferences. The decision set has a combinatorial structure, that is, it can be seen as the combination of certain issues, where each issue has a set of possible instances. Such a scenario is typical of several application domains ranging from configuration problems to recommender systems.

Usually preferences over combinatorially structured domains need to be expressed compactly and CP-nets are a successful framework that allows one to do this [1]. They exploit the independence among some features to give conditional preferences over small subsets of them.

Following the line of work in [13], we show how to smoothly embed influence functions in such a multi-agent CP-net profile. In particular, we provide results on the resistance of the considered framework to bribery and manipulation. When agents charge the briber a fixed cost, we show that the presence of influence makes bribery computationally difficult, even in a very restrictive setting. However, in some cases it can be easy to perform bribery when we interleave influence and preference aggregation, and influential agents charge the briber a cost for each feature for which they are asked to change their inclination. We also show that the way we deal with non-convergent influences may affect the resistance to manipulability.

2 Background: Influence functions and CP-nets

In [8, 9, 6, 7] a framework to model influences among agents in a social network environment is studied. Each agent has two possible actions to take and it has an inclination to choose one of the actions. Due to influence by other agents, the decision of the agent may be different from its original inclination. The transformation from the agent's inclination to its decision is represented by an *influence function*. In many real scenarios, influence among agents does not stop after one step but it is an iterative process.

An influence function can be modelled via a graph where nodes are states and arcs model state transitions via the influence function. Starting from an initial state, via the influence function we

may pass from state to state until stability holds (that is, in the graph formulation, we are in a state represented by a node with a loop), or we may also not converge. Here are some examples from [8]: The **Fol** influence function considers two agents, each of which follows the inclination of the other one. It converges to stability only when the initial inclination models consensus. In the **Gur** function, instead, one of the agents is a guru and all other agents follow him. Such a function has two stable states, which represent consensus, and to which any initial inclination will converge. Finally, the **Conf3** influence function models a community with a king, a man, a woman, and a child following a Confucian model: the man follows the king, the woman and child follow the man, and the king is influenced by others only if he has a positive inclination, in which case he will follow such an inclination only if at least one of the other people agrees with him. This influence function always converges to one of two stable states, which both represent consensus, depending on the initial state.

CP-nets [1] (for Conditional Preference networks) 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 cp-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 features $F = \{x_1, \ldots, x_n\}$ with finite domains $\mathcal{D}(x_1), \ldots, \mathcal{D}(x_n)$. For each feature x_i, we are given a set of *parent* features $Pa(x_i)$ 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. An *acyclic* CP-net is one in which the dependency graph is acyclic. Given this structural information, the agent explicitly specifies her preference over the values of a variable x for *each complete assignment* on $Pa(x)$. This preference is assumed to take the form of total or partial order over $\mathcal{D}(x)$. A cp-statement has therefore the general form $x_1 = v_1, \ldots, x_n = v_n : x = a_1 \succ \ldots \succ x = a_n$, where $Pa(x) = \{x_1, \ldots, x_n\}$, $D(x) = \{a_1, \ldots, a_n\}$, and $\succ$ is a total order over such a domain. The set of cp-statements regarding a certain variable X is called the *cp-table* for X.

Consider a CP-net whose features are A, B, C, and D, with binary domains containing f and $\overline{f}$ if F is the name of the feature, and with the preference statements as follows: $a \succ \overline{a}$, $b \succ \overline{b}$, $(a \wedge b) : c \succ \overline{c}$, $(\overline{a} \wedge \overline{b}) : c \succ \overline{c}$, $(a \wedge \overline{b}) : \overline{c} \succ c$, $(\overline{a} \wedge b) : \overline{c} \succ c$, $c : d \succ \overline{d}$, $\overline{c} : \overline{d} \succ d$. 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 $C = c$.

The semantics of CP-nets depends on the notion of a worsening flip. A *worsening flip* is a change in the value of a variable to a less preferred value according to the cp-statement for that variable. 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 α is *better* than another outcome β (written $\alpha \succ \beta$) iff there is a chain of worsening flips from α to β. This definition induces a preorder over the outcomes, which is a partial order if the CP-net is acyclic. In general, finding the optimal outcome of a CP-net is NP-hard [1]. However, in acyclic CP-nets, there is only one optimal outcome and this can be found in linear time by sweeping through the CP-net dependency graph, and assigning the most preferred values in the cp-tables. For instance, in the CP-net above, we would choose $A = a$ and $B = b$, then $C = c$, and then $D = d$. In the general case, the optimal outcomes coincide with the solutions of a set of constraints obtained replacing each cp-statement with a constraint [2]: from the cp-statement $x_1 = v_1, \ldots, x_n = v_n : x = a_1 \succ \ldots \succ x = a_n$ we get the constraint $v_1, \ldots, v_n \Rightarrow a_1$. For example, the following cp-statement (of the example above) $(a \wedge b) : c \succ \overline{c}$ would be replaced by the constraint $(a \wedge b) \Rightarrow c$.

CP-nets have been used as a compact way to represent the preferences of a set of voters. A sequential single-feature voting protocol can find a winner in polynomial time and has several other desirable properties [11].

3 Modelling preferences and influences

We follow the setting we defined in [13]. We consider a set of n agents expressing their preferences over a set of candidates with a combinatorial structure: there is a set of m binary features and each candidate is an assignment of values to all features. The approach we propose can be generalized to non-binary features. Each agent expresses its preferences over the candidates via an acyclic CP-net and the dependency graphs of such CP-nets must all be compatible with a linear order O over the features: for each voter, the preference over a feature is independent of features following it in O (known as O-legality in [11]). Notice that CP-nets respecting this property do not necessarily have the same dependency graph. Given n agents, m binary features, and a linear ordering O over the features, we call a collection of n acyclic CP-nets over the m features which are compatible with O a *p*rofile. A profile models the initial inclination of all agents, that is, their opinions over the candidates before they are influenced by each other. To avoid confusion we call variables the binary entities of each CP-net. Thus the whole profile has $m * n$ variables.

The notion of influence we adopt generalizes that in [8]: it can be either positive or negative, and it can occur across features. For example, an agent could say that "if Alice doesn't prefer pasta, I would like to take pasta". Or also, assume a set of friends needs to decide whether to go out together today or tomorrow, and if to have dinner or lunch. Then an agent could say "if Bob prefers to go out

tomorrow, I prefer to go for dinner". Moreover, we allow for conditional influence that holds only in a specific context, where the context is an ordering over the values of some variables. In our setting we model each influence function via one or more conditional influence statements. A *conditional influence statement (ci-statement)* on variable X has the form $o(X_1), \ldots, o(X_k) :: o(X)$, where $o(Y)$ is an ordering over the values of variable Y, for $Y \in \{X_1, \ldots, X_n, X\}$. Variables $X_1, \ldots X_k$ are the influencing variables and variable X is the influenced variable. A *ci-table* is a collection of ci-statements with the same influenced variable, and such that any pair of influencing contexts is mutually exclusive. Note that the semantics differs from the one used in cp-statements. Such a ci-statement must be interpreted as asserting the top choice for X in the specified influencing context, *disregarding the other variables.* [16]. Unlike a cp-table, a ci-table may not specify the values of the influenced variable for all possible assignments of the influencing variables. Since we are dealing with binary features, in what follows, we will just write the top elements of the orders in ci-statements (and in cp-statements).

An agent *positively influences* (resp. *negatively influences*) another agent when there are (resp. there are no) circumstances under which this other agent will adopt the same top ranked option as the influencing agent, and no (resp. there are) circumstances where he would deliberately pick a different one. If i always adopts the same top as j, we say that i follows j. Observe that influences can be 'mixed' when more than one influencing agent is considered.

A general mapping from any influence function to a set of ci-statements can easily be defined. Given an influence function f, we will call $ci(f)$ the ci-statements modelling f. Ci- and cp-statements are similar in syntax and are also linked in terms of their semantics: when the ci-statements $ci(f)$ of a function f are interpreted as cp-statements and turned into corresponding constraints, then we exactly obtain the stable states of the influence function f. This may seem surprising at first sight, given the different meaning of ci-statements and cp-statements.

Theorem 1. *Given an influence function f, consider the set of cp-statements S corresponding to the ci-statements $ci(f)$. Then the undominated outcomes of S coincide with the stable states of f.*

This result may suggest that ci-tables may be turned into cp-tables in the profile, thus getting rid of ci-statements. However, finding all the stable states of a function is not sufficient: what we need to know is whether a stable state can be reached from a given initial inclination. If no stable state can be reached, then we need to recognize such a situation and deal with it. To do so, we need to specify the dynamics of influence in our context.

3.1 Dynamics of influences

We adopt the following approach for modelling the dynamics of influences:

1. agents declare their initial inclination regarding the feature;
2. depending on the previous declarations, agents consider their ci-table and then simultaneously declare whether they stick to their opinion or change it (by influence);
3. if all agents stick to their opinion, the state is stable, otherwise the process is iterated.

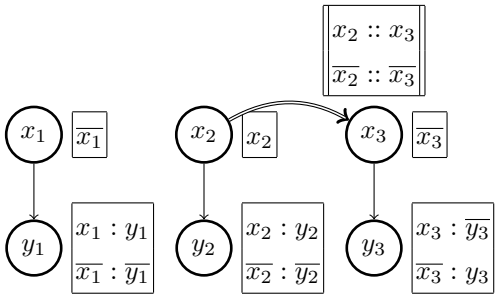
To find a stable state, or to find out that there is no stable state reachable from the initial inclination, we apply the above three steps iteratively. We first consider all variables regarding the same feature and we start with the assignment s of such variables modelling the initial inclination. We then move to another assignment s' by setting the value of each variable with an ingoing influence link to its most preferred value, given the values in s of its influencing variables. We then iterate this step until we either (i) reach a state which has been seen one step before (that is, a stable state), or (ii) reach a state already seen two or more steps before (that is, a cycle is detected). This check is achieved by keeping all the states seen so far (at most 2^n if we have n agents). If S is the list of all states seen so far, we denote by $S[k:]$ the list of items in S starting from the item in position k and going forward. If s' is the current state, and it is already in S in position k, then $S[k:]$ contains all states seen between two successive occurrences of s'. Thus, if s' is a stable state, $S[k:]$ contains just s'. On the other hand, if s' has been seen some steps before (and thus we are in a cycle), $S[k:]$ contains all the items in the cycle. If $S[k:]$ contains just one state, this stable state gives us the values to assign to the variables. If instead $S[k:]$ contains more than one state, we need to decide how to deal with the cycle. This is done by a function, which we call *Break-Cycle*, which chooses a state among the states in the cycle or the initial state s.

3.2 Influenced CP-nets

In line with CP-nets graphical notation, we use hyperarcs (called ci-arcs) going from the influencing variables to the influenced variable to graphically model influences. Notice that in this paper we consider acyclic CP-nets, while ci-arcs may create loops due to the iterative nature of influences. We define an *I-profile* to be a triple (P, O, S), where P is a profile, O is an ordering over the m features of the profile, and S is a set of ci-tables. Ordering O partitions the set of variables into m levels. Variables in the same level correspond to the same feature. Moreover, we assume that each variable can be influenced only by variables in her level or in earlier levels, but not both in the same ci-statement. Because of this restriction, ci-arcs in an I-profile can create cycles only among variables

of the same level. Notice that variables may appear both in ci-tables and cp-tables: influences and conditional preferences may be in conflict. In this case, *influences override preferences*.

Consider the I-profile below. There are three agents and thus we have three CP-nets. There are two binary features: X and Y. The ordering O is $X \succ Y$. Each variable X_i (resp., Y_i), with $i \in \{1, 2, 3\}$, has two values denoted by x_i and $\bar{x}_i$ (resp., y_i and $\bar{y}_i$). Notice that values x_i for the variables X_i correspond to value x for X, and similarly for Y. The variables X_i belong to the first level while the variables Y_i belong to the second level. Cp-statements are denoted by single-line arrows while ci-statements are denoted by doubled-line arrows. As it can be seen, agent 3 is influenced (positively) on feature X by agent 2.



To aggregate the preferences contained in an I-profile, while taking into account the influence functions, we propose to use a sequential approach where at each step we consider one of features, in the ordering stated by the I-profile. Both approaches includes three main phases: (*i*) influence iteration within one level, (*ii*) propagation from one level to the next one, and (*iii*) preference aggregation. In both cases, at the end, a winner candidate will be selected, that is, a value for each feature.

If the influence statements within each level model an influence function which always converges to a *consensus* state, as it is the case for the **Gur** or the **Conf3** functions, then preference aggregation is redundant, since all variables at the same level have the same value. If we have a possibly different value for the variables modelling the same feature, we consider two ways of combining preference aggregation and propagation: *Level Aggregation (LA)* performs, at each level, first influence iteration and then preference aggregation (by majority since variables are binary). At the end, all variables in the considered level have the same value. We then perform preference propagation to the next level, in order to set the initial inclination of the variables of the next level. If we use *Final Aggregation (FA)*, at each level we perform only influence iteration. At the end, each variable in the considered level has a final inclination, that is, an ordering for its values. We then propagate this information to the next level. As for LA, the goal of this propagation step is to set the initial inclination of the

variables of the next level. After all levels have been processed this way, each agent has its most preferred candidate. At this point we perform preference aggregation.

The two approaches may yield different results but not if the only change concerns the ordering.

Theorem 2. *Consider two I-profiles (P, O, S) and (P, O', S). Their two winners coincide, if we use the same aggregation method (either LA or FA).*

4 Bribery and Manipulation

We now turn our attention to other forms of influence in our setting, specifically bribery and manipulation. In the traditional (single issue) voting setting, a briber is an outside agent with a limited budget that attempts to affect the outcome of an election by paying some of the agents to change their preferences [5]. This scenario is particularly interesting when there are influential agents which, once bribed, can lead to a change in the inclination of other agents at no additional cost. On one side, thus, the presence of influences may favor the briber by making bribery cheaper. However, we show that, from a computational point of view, influences make the problem difficult for the briber, even in a very restrictive setting. In the results that follow we assume that *influence overrides bribery*. It thus makes no sense to bribe influenced agents as it would be either unnecessary (as the agent will eventually agree at no cost) or wasteful if, after being bribed, the agent changes his inclination against the briber following the influences. We will also assume two possible types of bribing costs that agents can charge to the briber: an agent may charge a fixed cost c_i for making any change to his CP-net, or it may charge a cost for each feature where he is asked to change his inclination.

Theorem 3. *Let P be an I-profile with positive acyclic influences, and where each agent is influenced by at most one other agent. Assume that only non-influenced agents can be bribed and that each agent i is associated with a (possibly different) bribery cost c_i. Then, given a candidate p and a budget B, deciding whether p can be made the winner within B is NP-complete both for LA and FA.*

If we fix the agents' bribery costs to be all the same, hardness still holds for LA and FA.

Theorem 4. *Let P be an I-profile and assume all agent have the same bribery cost. Then, given a candidate p and a budget B, deciding whether p can be made the winner within B is NP-complete for LA and FA.*

If we consider LA, it may be reasonable to consider the second bribery cost scheme, where agents charge a cost for each feature where they are asked to change their inclination. Assuming this

cost is the same for all the features at the same level, it is possible to show that bribery is actually in P even if we allow for a variety of different influences to take place.

Theorem 5. *Let P be an I-profile where influences are of type* **Gur***,* **Fol***,* **Conf3** *or they are acyclic positive or negative influences where each agent is influenced by at most one other agent. Assume that the cost of bribing an agent on a feature at a given level is the same for all agents. Then, given a candidate p and a budget B, deciding whether p can be made the winner within B is in P for LA.*

We now consider *manipulation* under LA, which occurs when one of the voters misreports its preferences in order to get a better collective choice. The problem is complicated by the fact that agents may be in turn influenced by the manipulating agent. This is reminscient of the *safe manipulation* setting [15], where an influential agent can be imitated in his vote by a *proportion* of followers.

Definition 1 **(X_i-manipulation).**
Let s be the vector of initial inclinations over feature X and and let X_i be the inclination of voter i in s. Moreover, let $n_{X_i}(s) = |\{j \in$ agents$|X_j = X_i\}|$, that is, the number of agents with the same inclination as i, f be an influence function, and $s' = \langle s_{-i}, \overline{X_i} \rangle$ be the vector of initial inclinations except that agent i misreports her inclination. We say that f is X_i-manipulable, i.e.*, agent i has an incentive to misreport her initial inclination if $n_{X_i}(IterInf(f, s)) < n_{X_i}(IterInf(f, s'))$.*

In words, voter i is manipulating if, by misreporting its preferences, the state that the influence propagation process reaches at stability has more variables in line with i's truthful view than the initial truthful vector (because majority is monotonic). We say that an influence function is *feature-manipulable* when there exists a feature X, an agent i, and some initial inclination for which the function is X_i-manipulable. Even with only positive influences, there are both examples of feature-manipulability and non-feature-manipulability.

Proposition 1. *Let f be an influence function without negative influences. Then (i) f is not feature-manipulable when* Break-Cycle *returns a state from $S[k:]$, and (ii) it is feature-manipulable when* Break-Cycle *returns the initial inclination.*

5 Future work

We plan to study the normative properties of LA and FA, as well as to asses their behavior via experimental tests. We also plan to study how to generalize our framework to allow for probability

distributions over influences, as in [8], as well as for influences over the top choice and the ordering among a set of possible actions, as in [7].

References

1. C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *JAIR*, 21:135–191, 2004.
2. R.I. Brafman and Y. Dimopoulos. Extended semantics and optimization algorithms for cp-networks. *Computational Intelligence*, 20(2):218–245, 2004.
3. M.H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69:118–121, 1974.
4. P. DeMarzo and D. Vayanos. Persuasion bias, social influence, and unidimensional opinions. *Quarterly Journal of Economics*, 118:909–968, 2003.
5. P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. How hard is bribery in elections? *JAIR*, 35:485–532, 2009.
6. M. Grabisch and A. Rusinowska. A model of influence in a social network. *Theory and Decisions*, 69(1):69–96, 2010.
7. M. Grabisch and A. Rusinowska. A model of influence with an ordered set of possible actions. *Theory and Decisions*, 69(4):635–656, 2010.
8. M. Grabisch and A. Rusinowska. Iterating influence between players in a social network. In *Proc. 16th Coalition Theory Network Workshop*, 2011.
9. C. Hoede and R. Bakker. A theory of decisional power. *Journal of Mathematical Sociology*, 8:309322, 1982.
10. U. Krause. A discrete nonlinear and nonautonomous model of consensus formation. *Communications in Difference Equations*, 2000.
11. J. Lang and L. Xia. Sequential composition of voting rules in multi-issue domains. *Mathematical social sciences*, 57:304–324, 2009.
12. N. Mattei, F. Rossi, K. B. Venable, and M. S. Pini. Bribery in voting over combinatorial domains is easy. In *Proc. AAMAS 2012*, pages 1407–1408, 2012.
13. N. Maudet, M. S. Pini, F. Rossi, and K. B. Venable. Influence and aggregation of preferences over combinatorial domains. In *Proc. AAMAS 2012*, pages 1313–1314, 2012.
14. P.Faliszewski and A. D. Procaccia. Ai's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.
15. Arkadii Slinko and Shaun White. Is it ever safe to vote strategically? Department of mathematics - research reports-563, 2008.
16. N. Wilson. Extending cp-nets with stronger conditional preference statements. In Deborah L. McGuinness and George Ferguson, editors, *AAAI*, pages 735–741. AAAI Press / The MIT Press, 2004.

Appendix: Proofs

Theorem 1 *Given an influence function f, consider the set of cp-statements S corresponding to the ci-statements $ci(f)$. Then the undominated outcomes of S coincide with the stable states of f.*

Proof. We note that, since we may not have a ci-statement for every possible ordering over the influencing variables, N may not be a CP-net. Nevertheless, the cp-statements induce an ordering over the outcomes according to the semantics given in Section 2. Let $o = (X_1 = x_1, \ldots X_t = x_t)$ be an undominated outcome in this ordering. This means that flipping any value in o is never improving in any cp-statement. Thus there cannot be any ci-statement for which the assignment in o of the influenced variable would be changed given the assignments in o to its influencing variables. Thus o is a stable state for the influence function.

Now let d be a dominated outcome according to the ordering induced by N. If it is dominated, by the cp-statement semantics, it must be dominated by at least one outcome, say d' that is one flip away. Let X be the variable on which they differ. Since d' dominates d, there must be a cp-statement on X where the parents of X are assigned the values they have in d (and d'), and according to which the value X in d' is preferred to the one in d. Thus, applying the influence function to the state corresponding d would induce a change in the inclination on X. This allows us to conclude that this state is not stable

Theorem 2 *Consider two I-profile (P, O, S) and (P, O', S). Their two winners coincide, if we use the same aggregation method (either LA or FA).*

Proof. If we consider an I-profile (P, O, S), any other I-profile (P, O', S) will produce the same final result. Different orderings of an I-profile with the same profile and the same ci-statements will possibly order differently only variables that are independent both in terms of preferences and influence functions.

Theorem 3 *Let P be an I-profile where influences are all positive and acyclic, and each agent is influenced by at most one other agent. Assume that only non-influenced agents can be bribed and that each such agent i is associated with a (possibly different) bribery cost c_i. Then, given a candidate p and a budget B, deciding whether p can be made the winner within B is NP-complete both for LA and FA.*

Proof. It is in NP, since, given a bribery strategy (i.e., a set of agents to bribe), it is possible to test in polynomial time if the winner is p and if the budget does not exceed B. To show NP-completeness, we use a reduction from X3C. In X3C we are given an instance (B, S) of X3C,

where $B = \{b_1, \ldots, b_{3k}\}$ is some set and $S = S_1, \ldots, S_n$ is a family of 3-element subsets of B. We ask if there is a collection of exactly k sets in S so that their union is B. We create a bribery instance on an influenced profile as follows. We will have a single binary feature X with values x and $\bar{x}$, and $n + 3k + (n + 3k - 1)$ voters. The first n voters correspond to the sets in S, the next $3k$ voters correspond to elements $b_1, \ldots, b_{3k}$, and the last $(n + 3k - 1)$ voters are dummies. Voters in $S \cup B$ and $4k - 1$ dummy voters vote $\bar{x}$. The other voters vote x. Thus, we have:

- $n + 3k + 4k - 1 = n + 7k - 1$ votes for x;
- $2n + 6k - 1 - (n + 7k - 1) = n - k$ votes for x.

Further, the prices are set so that each voter in S costs 1, and all the other voters cost $k + 1$. The budget is set to k. The (positive) influences are defined as follows: an agent b_i is influenced by agent S_j if and only if b_i belongs to S_j. It is easy to see that it is possible to ensure that x wins by bribery, only if there is a way to pick k sets from S such that their union is B. If such sets exist, then after the bribery we have additional $4k$ votes for x (altogether there are $n + 3k$ votes for x, and x is chosen). On the other hand, if there is no way to pick such k sets, then by bribing k voters we cannot get more than $4k - 1$ voters to switch from $\bar{x}$ to x, and the bribery is impossible.

Theorem 4 *Let P be an I-profile where each agent is associated with the same bribery cost. Then, given a candidate p and a budget B, deciding whether p can be made the winner within B is NP-complete for LA and FA.*

Proof. The result for LA derives from the fact that this problem has been shown to be NP-complete for CP-nets with no influences [12]. The result for FA derives from the fact that the reduction of Theorem 3 works also if all the agents have the same prices: we still have to bribe voters in S because otherwise we would not be able to get enough votes for x.

Theorem 5 *Let P be an I-profile where influences are of type* **Gur***,* **Fol***,* **Conf3** *or they are acyclic positive or negative influences where each agent is influenced by at most one other agent. Assume that the cost of bribing an agent on a feature at a given level is the same for all agents. Then, given a candidate p and a budget B, deciding whether p can be made the winner within B is in P for LA.*

Proof. The main idea underlying this result is to transform the bribery problem at each level into a problem of bribery with weighted agents and a fixed set of costs. The overall algorithm applies, to each level, first bribery, then influence, and then aggregation. After each step at each level, intra-level propagation takes place. In what follows we assume the briber is pushing for value 1. The key step

is the computation of the minimal bribery cost at each level. If the sum of the costs on all levels does not exceed the budget, we accept, otherwise we reject. For each level we do the following. We first compute the bribery cost for each cluster. The cluster size is 2 in the case of function **Fol**, 4 for **Conf3** and can be anywhere between 2 and n for **Gur**. We assume that each agent belongs to a unique cluster. We note that it is possible to compute the minimum cost for bribing a cluster subject to one of the aforementioned functions in the linear time in the size of the cluster. For **Fol**, either both agents agree with each other and with the briber (cost 0), or they both disagree with the briber (cost 2) or only one of them disagrees and needs to be bribed (cost 1); for **Gur**, if the guru agrees we have cost 0, otherwise cost 1; for **Conf3**, if the king and one other agent of the cluster agree with the briber we have cost 0; if at least one agent, other than the king, agrees with the briber then we have cost 1 for bribing the king; otherwise, cost 2 for bribing the king and another agent. We then set the vote on all of those clusters with cost 0 to be in favor of the briber (i.e., equal to the projection of p on the current feature). If we have not reached the majority, we replace each **Fol** cluster with a single agent of weight 2 and cost either 1 or 2, depending on how many agents must be bribed. We replace every **Conf3** cluster with a single agent with weight 4 and cost either 1 or 2 and every **Gur** cluster with a single agent with weight equal to the number of agents in the cluster and cost 1. For each agent, not in any cluster and not influenced by other agents, we compute the number of votes he brings in favor of the briber both when he votes for and against p (this is due to the presence of negative influences). Let t be the maximum gain that such a agent can bring. We replace the agent and all the agents influenced (directly or indirectly) by him with a single agent with cost 1 and weight t. We then order all the new agents with weight greater than or equal to 3 and cost 1 in decreasing order of weight and we bribe them following such an ordering. If by doing so we don't exceed the budget or we do not obtain a majority, we continue by bribing the agents with weight 4 and cost 2, then the agents with cost 1 and weight 2, then the agents with cost 2 and weight 2, and finally the ones with cost 1 and weight 1. This until we either exceed the budget or we obtain a majority. The only caution that should be taken, is that if the budget is exceeded by bribing a voter of cost 2, then the following ones with cost 1 should be considered. Assuming the budget is not exceeded, influences are applied and the result is propagated to the next level where the procedure is iterated given the residual budget.

Proposition 1 *Let f be an influence function without negative influences. Then (i) f is not feature-manipulable when* Break-Cycle *returns a state from $S[k:]$, and (ii) it is feature-manipulable when* Break-Cycle *returns the initial inclination.*

Proof. We only show (ii). The first statement is easily shown because n_{X_i} must grow monotonically with each state. Consider the example with 5 agents A, B, C, D, and E with influences such that agent A is followed by B, B is followed by A, D is followed by C, and E is followed by D. Now compare the initial inclinations $s = \langle 0, 1, 1, 0, 0 \rangle$ and $s' = \langle 0, 0, 1, 0, 0 \rangle$. From s the algorithm ends up in a cycle (because of the mutual influence of A and B). Hence s is returned. But if we start from s', the stable state $s' = \langle 0, 0, 1, 1, 1 \rangle$ is attained. Thus agent B is better off misreporting her initial inclination.