Influencing and aggregating agents' preferences over combinatorial domains

Nicolas Maudet[1], **Maria Silvia Pini**[2], **Francesca Rossi**[2], **K. Brent Venable**[2]
1: LAMSADE, Univ. Paris Dauphine, France
Email: Nicolas.MAUDET@dauphine.fr
2: Department of Pure and Applied Mathematics,
University of Padova, Italy
Email: {mpini,frossi,kvenable}@math.unipd.it

Abstract

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 opinions before voting. In this initial phase, agents may influence each other and therefore modify their preferences, until hopefully they reach a stable state. Recent work has modelled 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. When agents express their preferences as CP-nets, we show how to model influence functions and to aggregate preferences by possibly interleaving voting and influence convergence.

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 opinions before voting. For example, in political elections, polls provide a representative sample of the opinion of the voters, and some influential people may declare their vote inclination. Moreover, in social networks, people often exchange their opinions before taking a decision.

In this initial phase, agents may influence each other and therefore modify their preferences. For example, in political elections, a voter may be influenced by the opinion of esteemed people. In a work environment, the participants to a project meeting may have to take one or more decisions about the project plan and may be influenced by the opinion of experts of the field.

The concept of influence has been widely studied in psychology, economics, sociology, and mathematics [DeGroot, 1974; P. DeMarzo, 2003; Krause, 2000]. Recent work has modelled the influence phenomenon in the case of taking a decision over a single issue [Grabisch and Rusinowska, 2010]. In this influence framework, 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 the original inclination. The transformation from the agent's inclination to its decision is represented by an influence function. It is also interesting to draw connection to the recent work on (some kind of) manipulation in computational social choice. In so-called *bribery* problems [Faliszewski *et al.*, 2009], an agent has typically a limited budget he can spend to modify the vote of other agents. In the *safe manipulation* setting [Slinko and White, 2008], it is assumed that an influential agent can be imitated in his vote by a proportion of followers. These are clearly specific notions of influence, but restricted in the sense that a single influencing agent is considered, and that the process is simply one-shot. In many real scenarios, influence among agents does not stop after one step but it is an iterative process.

Here we generalize these models to account for preferences over combinatorially structured domains including several issues. In fact, often a set of agents needs to select a common decision from a set of possible decisions, over which they express their preferences, and such a 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. Consider for example a car: usually it is not seen as a single item, but as a combination of features, such as its engine, its shape, its color, and its cost. Each of these features has some possible instances, and a car is the combination of such feature instances. If a family needs to buy a new car, each family member may have his own opinion about cars, and the task is to choose the car that best fits the preferences of everybody.

Usually preferences over combinatorially structured domains are expressed compactly, otherwise too much space would be needed to rank all possible alternatives. CP-nets are a successful framework that allows one to do this [Boutilier *et al.*, 2004]. They exploit the independence among some features to give conditional preferences over small subsets of them.

CP-nets have already been considered in a multi-agent setting [Rossi *et al.*, 2004; Lang and Xia, 2009; Purrington and Durfee, 2007; Xia *et al.*, 2008]. Here we adapt such frameworks to incorporate influences among agents. We allow influences to be over the same issue or also among different issues. We show how to model influence functions and we observe that influence and conditional preferential dependency in CP-nets have the same semantic model. This allows us to naturally embed influences in a multi-agent CP-net profile.

We then propose a way to aggregate preferences by possibly interleaving voting and influence convergence.

2 Background

2.1 Influence functions

In [Grabisch and Rusinowska, 2010] a framework to model influences among agents in a social network environment is defined. 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.

Any influence function over n agents can be modelled via a matrix with 2^n rows and 2^n columns, where each row and column correspond to a certain state (a vector containing the agents' inclinations). A 1 in the cell (S, T) of the matrix means that from state S we pass to state T via the influence function. Alternatively, the influence function can be modelled via a graph where nodes are states and arcs model state transitions via the influence function. If we adopt the iterative model of influence, 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.

Let us consider some examples of influence functions, as defined in [Grabisch and Rusinowska, 2010]:

- The **Fol** influence function considers two agents, each of which follows the inclination of the other one. This influence function converges to stability only when the initial inclination models consensus between the two agents. If we start from another state, influence iteration never stops.
- On the other hand, in the **Id** influence function, where each of agent follows only its own inclination, all states are stable.
- Another example is the influence function modelling the presence of a guru, called **Gur**, where one of the agents is the guru and all other agents follow him. Such a function has two states, which both represent consensus. Given any initial inclination, the iteration will converge to one of the stable states.
- A final example, that we will consider also later in the paper, is the **Conf3** influence function, that models a community with 4 people which follow a Confucian model. The four people are a king, a man, a woman, and a child. 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. In [Grabisch and Rusinowska, 2010] it is shown that this influence function always converges to one of two stable states, which both represent consensus, depending on the initial state.

2.2 CP-nets

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 (cp)* preference 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 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. Given this structural information, the agent explicitly specifies her preference over the values of x_i for *each complete assignment* on $Pa(x_i)$. This preference is assumed to take the form of total or partial order over $\mathcal{D}(x_i)$. An *acyclic* CP-net is one in which the dependency graph is acyclic.

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: $a \succ \overline{a}$, $b \succ \overline{b}$, $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) : c \succ \overline{c}$, $(a \wedge \overline{b}) \vee (\overline{a} \wedge b) : \overline{c} \succ c$, $c : d \succ \overline{d}$, $\overline{c} : \overline{d} \succ d$. 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 $C = c$.

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 [Boutilier *et al.*, 2004]. 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, assigning the most preferred values in the preference 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 constraint problem obtained replacing each cp-statement with a constraint [Brafman and Dimopoulos, 2004]. For example, the following cp-statement (of the example above) $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) : c \succ \overline{c}$ would be replaced by the constraint $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) \Rightarrow c$.

In the context of preference aggregation, CP-nets have been used as a compact way to represent the preferences of each voter. In particular, in [Lang and Xia, 2009] the authors showed that a sequential single-feature voting protocol can find a winner object in polynomial time. Moreover, such an approach has several other desirable properties, when the CP-nets satisfy a certain condition on their dependencies called *O-legality*. In [Lang and Xia, 2009], the CP-nets must be acyclic, and their dependency graphs must all be compatible with a given graph ordered according to the feature ordering in the voting procedure. In other words, there is a linear order O over the features such that for each voter the preference over a feature is independent of features following it in O.

3 Modelling influence

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 features and the set of candidates is the Cartesian product of their domains. Thus each candidate is an assignment of values to all features.

For the sake of simplicity of the technical developments of this paper, we assume features to be binary (that is, with two values in their domain). However, the approach we propose can be generalized to non-binary features.

Each agent expresses its preferences over the candidates via an acyclic CP-net. Moreover, we assume that these CP-nets are compatible: given n CP-nets $N_1, \ldots, N_n$, they are said to be compatible if the union of their dependency graphs, that we call $Dep(N_1, \ldots, N_n)$, does not contain cycles. Notice that compatible CP-nets do not necessarily have the same dependency graph.

Definition 1 *Given n agents and m binary features, a profile is a collection of n compatible CP-nets over the m features.*

We note that our notion of profile coincides with the notion of O-legal profile in [Lang and Xia, 2009].

Given a profile P with CP-nets $N_1, \ldots, N_n$, we will abuse the notation and often write $Dep(P)$ to mean $Dep(N_1, \ldots, N_n)$.

A profile models the initial inclination of all agents, that is, their opinions over the candidates before they are influenced by each other.

Since the set of features is the same for all agents, but each agent may have a possibly different CP-net, to avoid confusion we call variables the binary entities of each CP-net. Thus, given a profile with m features, for each feature there are n variables modelling such a feature, one for each CP-net. Thus the whole profile has $m * n$ variables.

3.1 Conditional influence

A straightforward way to include influences into profiles is to have influence functions act on each single feature, as in [Grabisch and Rusinowska, 2010]. That is, the preferences of an agent over a certain feature may be influenced by the preferences of one or more other agents over the same feature.

While influence functions in [Grabisch and Rusinowska, 2010] allow only for positive influence, we adopt a more general notion of influence, which changes the opinion of an agent but not necessarily making it the same as the opinion of the influencing agents. Thus, being influenced just means that an agent modifies his opinion w.r.t. his current inclination. For example an agent could say that ”if Bob likes white wine, I would like to take white wine as well”, or ”if Alice doesn’t like pasta, I would like to take pasta”.

Moreover, we allow for conditional influence that holds only in a specific context, where the context is the assignment of some variables. For example, an agent could say ”if we decide to drink wine, I will follow Bob’s preferences, otherwise I will follow my inclination”.

Besides this form of influence over the same feature, we also want to allow influence to come from the preferences of other agents over different features. For example, 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”.

In [Grabisch and Rusinowska, 2010] an influence function is a set of statements, or equivalently a matrix or a graph, saying how agents are influenced by each other. We will model each influence function via one or more conditional influence statements.

Definition 2 *A conditional influence statement (ci-statement) on variable X has the form*

$$X_1 = v_1, \ldots, X_k = v_k :: o(X)$$

where $o(X)$ is an ordering over the values of variable X. Variables $X_1, \ldots X_k$ are the influencing variables and variable X is the influenced variable.

A ci-statement $X_1 = v_1, \ldots, X_k = v_k :: o(X)$ models the influence on variable X of an assignment to the set of influencing variables $X_1, \ldots, X_k$. A ci-table is a collection of ci-statements with the same influencing and influenced variables, and containing at most one ci-statement for each assignment of the influencing variables.

As in CP-nets dependencies are graphically denoted by hyperarcs, we also use hyperarcs to graphically denote ci-tables. Such hyperarcs go from the influencing variables to the influenced variable. To distinguish them from the dependencies, we call them ci-arcs.

Definition 3 *An i-profile is 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.*

Moreover:

- *The ordering O of the features must be such that $Dep(P)$ has only arcs from earlier variables to later variables. This ordering partitions the set of variables into m levels. Variables in the same level correspond to the same feature.*
- *The ci-tables of an i-profile must be such that each variable can be influenced by variables in her level or in earlier levels, but not in the same ci-statement.*

Notice that, because of the restriction we impose on ci-tables, ci-arcs in an i-profile can create cycles only within variables of the same level.

Example 1 *Consider the i-profile of Figure 1. There are three agents and thus we have three CP-nets. In this example the three CP-nets have the same dependency structure (thus they are obviously compatible). There are two binary features: X and Y, with values, respectively, x and $\bar{x}$, and y*

and $\bar{y}$. The ordering O is $X \succ Y$. Thus the i-profile has six variables denoted by X_1, X_2, X_3, Y_1, Y_2, and Y_3. 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-dependencies are denoted by solid-line arrows and ci-statements are denoted by dotted-line arrows. As it can be seen, agent 3 is influenced (positively) on feature X by agent 2.

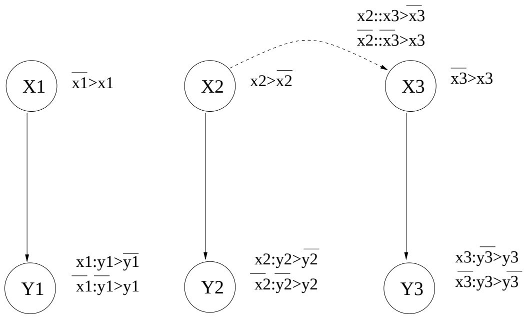

Figure 1: Example of an i-profile.

3.2 Modelling influence functions

Consider the **Conf3** influence function. There is a binary issue to be decided upon, and four people that express their opinions: a king, a man, a woman, and a child. 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 at least one the other people agrees with him. As shown in [Grabisch and Rusinowska, 2010], this influence function converges to one of two stable states, which both represent consensus, depending on the initial state.

To model this function, we may use a single binary feature X and 4 binary variables X_k, X_m, X_w, and X_c. Each variable X_i, with $i \in \{k, m, w, c\}$, has two values denoted by x_i and $\bar{x}_i$.

The ci-tables representing the influences are:

$King$	Man
$\bar{x}_k - -- :: \bar{x}_k \succ x_k$	$x_k :: x_m \succ \bar{x}_m$
$x_k \bar{x}_m \bar{x}_w \bar{x}_c :: \bar{x}_k \succ x_k$	$\bar{x}_k :: \bar{x}_m \succ x_m$
$x_k x_m - - :: x_k \succ \bar{x}_k$	
$x_k - x_w - :: x_k \succ \bar{x}_k$	
$x_k - -x_c :: x_k \succ \bar{x}_k$	

$Woman$	$Child$
$x_m :: x_w \succ \bar{x}_w$	$x_m :: x_c \succ \bar{x}_c$
$\bar{x}_m :: \bar{x}_w \succ x_w$	$\bar{x}_m :: \bar{x}_c \succ x_c$

A general mapping from any influence function to a set of ci-statements can easily be defined. In general, this mapping will produce between 1 and $n \times 2^n$ ci-statements if we have n agents. In the above example we have exploited the fact that the influence function has a compact formulation in terms of as many influence statements as the number of people involved, and thus we have obtained a much smaller number of ci-statements.

Given an influence function f, we will call $ci(f)$ the ci-statements modelling f.

3.3 Ci- or cp-statements?

It is interesting to notice that ci-statements can be interpreted as cp-statements. In fact, if we see the statements $ci(f)$ as cp-statements, their optimal outcomes coincide with the stable states of the influence function f.

As it is known [Brafman and Dimopoulos, 2004], the optimal outcomes of a set of cp-statements are the solutions of a set of constraints, where each constraint correspond to one of the cp-statements. Following this approach, the constraints corresponding to the statements above are:

- for the king:
  $\bar{x}_k - -- \Rightarrow \bar{x}_k$
  $x_k \bar{x}_m \bar{x}_w \bar{x}_c \Rightarrow \bar{x}_k$
  $x_k x_m - - \Rightarrow x_k$
  $x_k - x_w - \Rightarrow x_k$
  $x_k - -x_c \Rightarrow x_k$
- for the man:
  $x_k \Rightarrow x_m$
  $\bar{x}_k \Rightarrow \bar{x}_m$
- for the woman:
  $x_m \Rightarrow x_w$
  $\bar{x}_m \Rightarrow \bar{x}_w$
- for the child:
  $x_m \Rightarrow x_c$
  $\bar{x}_m \Rightarrow \bar{x}_c$

The only two solutions of this set of constraints are: (x_k, x_m, x_w, x_c) and $(\bar{x_k}, \bar{x_m}, \bar{x_w}, \bar{x_c})$, which are exactly the two stable states of the **Conf3** influence function.

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

In other words, influences and cp-dependencies are not different in their semantics. This is very useful, since it allows for a very simple integration of ci- and cp-statements in the same profile. However, we need to give them a different syntax since we must distinguish between the initial inclination of the agents, given by the cp-statements, and the influences, given by the ci-statements. In fact, influences modify the initial inclination by overriding the preferences, but the opposite does not hold. So it would be a mistake to just treat the ci-statements as additional cp-statements in the profile.

4 Aggregating influenced preferences

We will now propose a way to aggregate the preferences contained in an i-profile, while taking into account the influence functions. The main idea is to use a sequential approach where at each step we consider one of features, in the ordering stated by the i-profile. The method we propose includes three main phases: influence iteration within one level, propagation from one level to the next one, and preference aggregation. At the end, a winner candidate will be selected, that is, a value for each feature.

In the following subsections, we will describe each of these phases and how they can be combined.

4.1 Influence iteration

For each feature, we consider the influences among different variables modelling this feature and thus belonging to the same level. What we need to do is to find, if it exists, the stable state of such influences corresponding to the initial inclination of the agents. Such inclination is given by the cp-tables of these variables in the profile.

Consider the hypergraph corresponding to the ci-statements over variables representing the same feature. We consider this hypergraph to be cyclic if there are cycles of length at least 2. In fact, a cycle of length 1 models the fact that a variable is influenced by other variables and also by its current inclination.

Notice that, when we are at the first level, the variables are all independent in terms of cp-dependencies, so each agent has an inclination over the values of his variable which does not depend on any other variable.

To find stability or to find out that there is no stable state, we employ an iterative algorithm (see Algorithm 1 below). This algorithm starts with the assignment s of all variables given by their initial inclination, which can be seen in their cp-statements, and moves to another assignment s' by setting the value of each variable to its most preferred value given the values in s of its influencing variables (this is achieved by function ci-flip in Algorithm 1). It then iterates this step until either it reaches a fixpoint or it sees an assignment twice. In the first case, the fixpoint gives us a stable state and the variables are fixed to such values. In the second case, it stops and reports a non-convergent influence for the variables of the considered level.

Algorithm 1: Influence iteration algorithm

```
$s = (s_1, \ldots, s_n)$ // the initial inclination
$s' = s$
**repeat**
    $s = s'$
    **for** *i=1 to n* **do**
        $s'_i$ = ci-flip($s, i$)
**until** $s = s'$ *or* $s'$ *already seen* ;
**if** $s = s'$ **then**
    return s
**else**
    return ”No convergence”
```

Notice that, if the ci-statements do not generate cycles, stability is always reached, since the structure is assimilable to an acyclic CP-net, which always has exactly one optimal outcome, thus by Theorem 1 the influence statements have exactly one stable state corresponding to the initial inclination.

4.2 Propagation

Once the variables of a certain level have been fixed to some values, by the influence iteration procedure outlined above, we can propagate to the next level this information by considering the ci- and cp-statements that go from the current level to the next one. Propagation through a ci- or cp-table is achieved by eliminating the conditional statements that refer to conditions not satisfied by the chosen assignment of the influencing or parent variables. The resulting table has exactly one value ordering, giving us the inclination of that variable.

Since influence overrides preference, we first look at the ci-tables and set the inclination of the influenced variables according to such tables. For the variables whose inclination has not been determined after this step, their inclination will be determined by their cp-tables.

After this, we are ready to handle the next level as we did for the first one, since all of its variables are now subject only to influence functions.

4.3 Preference aggregation

In the previous section we have described how to reach stability within one level and how to propagate the decision taken at one level to the next one. It remains to decide when to perform preference aggregation in order to obtain a winner from the profile.

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 aggregation is redundant, since all variables at the same level have the same value. Thus the most preferred outcome is the same for all agents, and this will be declared the winner (with any unanimous voting rule).

However, at each level we obtain a possibly different value for the variables modelling the same feature. Now we can either aggregate at each level, and then propagate the result to the next level, or we can aggregate only at the end of the procedure, when each agent will have a most preferred candidate.

If we decide to aggregate at each level, we will choose by majority (since variables are binary) which value to give to all variables of the considered level. Then we propagate such a choice to the next level and start again with an influence iteration. We call LA this method (for *Level Aggregation*).

Otherwise, we can leave the variable values in each level as they are after the influence iteration and proceed with the interleaving of propagation and convergence, until all levels have been handled. At this point, we have a most preferred candidate for each agent, and we can obtain a winning candidate by any voting rule that needs the top choices, such as plurality. We call FA this method (for *Final Aggregation*).

The two approaches yield different results as shown by the following example.

Example 2 *Let us consider the i-profile of Figure 1. After the influence iteration step at level 1 (that is, on feature X), the preference of agent 3 is $x_3 \succ \bar{x}_3$, while the preferences of the other agents are unchanged.*

Assume to adopt method LA. Then we now aggregate the votes over X by majority. This results in $X = x$ winning and *thus the variables of the first level are set to the following values: $X_1 = x_1$, $X_2 = x_2$, and $X_3 = x_3$. We then propagate such assignments to the next level and we get the following assignment for the variables corresponding to the Y feature: $Y_1 = y_1$, $Y_2 = y_2$, and $Y_3 = \bar{y}_3$. We now aggregate the votes over Y by majority, and the winning assignment is $Y = y$. Thus the overall winner of the procedure is $\langle X = x, Y = y \rangle$.*

Instead, if we follow the FA procedure, the assignments for X that are propagated are those after the influence iteration, that is, $X_1 = \bar{x}_1$, $X_2 = x_2$, and $X_3 = x_3$. This gives, through propagation, the following values for the variables corresponding to Y: $Y_1 = \bar{y}_1$, $Y_2 = y_2$, and $Y_3 = \bar{y}_3$. Thus we have the following three top candidates for the three agents: $C1 = (X = \bar{x}, Y = \bar{y})$, $C2 = (X = x, Y = y)$, and $C3 = (X = x, Y = \bar{y})$. Now we aggregate, for example by using plurality, with a tie-breaking rule where precedence is given by a lexicographical ordering where $\bar{x} \succ x$ and $\bar{y} \succ y$. According to this rule, the winner is $(X = \bar{x}, Y = \bar{y})$.

Notice that the choice of the ordering does not matter, since, if we consider an i-profile (P, O, S), any other i-profile (P, O', S) will produce the same final result. In fact, different orderings of an i-profile with the same profile and the same ci-statements will order differently variables that are independent both in terms of preferences and influence functions.

However, as seen in the example above, in general the two procedures LA and FA return different winners. Moreover, some agents may be better off with one of the two procedures, while others may be better off with the other one. This is the case of agent 1, that gets its top candidate to win with FA, while it would get a worse candidate with LA. The opposite situation holds for agent 2.

5 Conclusions and future work

In this paper we have assumed that agents express their preferences via CP-nets. We also plan to consider settings where other formalisms for compact preference representation are used, such as soft constraints.

We plan to study the normative properties of procedures LA and FA, as well as to asses their behavior via experimental tests.

In [Grabisch and Rusinowska, 2010] there are also influence functions where influence is followed with a certain probability, otherwise the agent follows its inclination. We plan to study how to generalize our framework to allow for such influence functions.

In [M. Grabisch, 2003] influence is over the top choice among a set of possible actions, not just two. We plan to formalize the extension of our approach to this case. We also plan to allow for influences over the ordering of the actions, rather than just over the top element of such an ordering.

References

[Boutilier *et al.*, 2004] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)*, 21:135–191, 2004.

[Brafman and Dimopoulos, 2004] R.I. Brafman and Y. Dimopoulos. Extended semantics and optimization algorithms for cp-networks. *Computational Intelligence*, 20(2):218–245, 2004.

[DeGroot, 1974] M.H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69:118–121, 1974.

[Faliszewski *et al.*, 2009] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. How hard is bribery in elections? *J. Artif. Intell. Res. (JAIR)*, 35:485–532, 2009.

[Grabisch and Rusinowska, 2010] Michel Grabisch and Agnieszka Rusinowska. Iterating influence between players in a social network. Documents de travail du centre d'economie de la sorbonne, Universit Panthon-Sorbonne (Paris 1), Centre d'Economie de la Sorbonne, 2010.

[Krause, 2000] U. Krause. A discrete nonlinear and nonautonomous model of consensus formation. *Communications in Difference Equations*, 2000.

[Lang and Xia, 2009] Jerome Lang and Lirong Xia. Sequential composition of voting rules in multi-issue domains. *Mathematical social sciences*, 57:304–324, 2009.

[M. Grabisch, 2003] A. Rusinowska M. Grabisch. A model of influence wuth an ordered set of possible actions. *Theory and Decisions*, 69(4):635–656, 2003.

[P. DeMarzo, 2003] D. Vayanos P. DeMarzo. Persuasion bias, social influence, and unidimensional opinions. *Quarterly Journal of Economics*, 118:909–968, 2003.

[Purrington and Durfee, 2007] K. Purrington and E. H. Durfee. Making social choices from individuals' cp-nets. In *AAMAS*, page 179. IFAAMAS, 2007.

[Rossi *et al.*, 2004] F. Rossi, K.B. Venable, and T. Walsh. mcp nets: Representing and reasoning with preferences of multiple agents. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, pages 729–734, 2004.

[Slinko and White, 2008] Arkadii Slinko and Shaun White. Is it ever safe to vote strategically? Department of mathematics - research reports-563, 2008.

[Xia *et al.*, 2008] L. Xia, V. Conitzer, and J. Lang. Voting on multiattribute domains with cyclic preferential dependencies. In *AAAI*, pages 202–207. AAAI Press, 2008.