EMERGENCE OF COOPERATION THROUGH MUTUAL PREFERENCE REVISION

Pedro Santana[1] and Luís Moniz Pereira[2]

[1] IntRoSys S.A.
[2] Universidade Nova de Lisboa,
Quinta da Torre, Campus FCT-UNL
2829-516 - Portugal

Abstract. This paper proposes a method allowing an agent to perform in a socially fair way by considering other agents' preferences. Such a balanced action selection process is based on declarative diagnosis, which enables the removal of contradictions arising as all agents' preferences are confronted within the deciding agent. Agents can be negatively affected when some of their preferences are not respected for the sake of a global compromise. The set of preferences to be yielded by agents in order to remove all contradictions in a balanced way (i.e. the diagnosis that better manages how each agent is to be affected) is determined by minimising a cost function computed over application independent features. By respecting the resulting non-contradictory preferences set, the deciding agent acts cooperatively.
Key words: Preference revision, multi-agent systems, cooperative behaviour.

1 INTRODUCTION

Preference criteria are subject to be modified when new information is brought to the knowledge of the individual, or aggregated when we need to represent and reason about the simultaneous preferences of several individuals. As a motivating example, suppose you invite three friends Karin, Leif and Osvald to go and see a movie. Karin prefers thrillers to action movies. Leif, on the other hand, prefers action movies to thrillers. Finally, Osvald is like Leif and prefers action movies to thrillers. Suppose you need to buy the tickets. Which movie do you choose?

One way to consider preferences in Multi-Agent scenarios, is to devise a strategy for removal of contradictions, which may arise when the preferences of all agents are put together. For instance, if *agent A prefers a to b* and agent B the other way around, then we have a contradiction (a synonym of conflict in this context). Removing the referred contradiction is another way of saying that at least one of the agents will have to *relax* its own preferences so a trade-off is achieved.

This paper proposes a methodology along this line of reasoning, that can be used as an agent mental process to allow the agent to perform in a fair manner. For instance, an agent engaged in a task that requires to make some choices, like selecting a TV programme, may choose to act fairly by considering others' preferences as well. Another possible application for the method set forth herein is to help a *broker* (i.e. an agent

responsible for distributing work) to allocate agents to those tasks that better suit them; in this case preferences can involve skills as well.

Other work on preferences revision can be found in [1, 2], where the authors study the preservation of properties by different composition operators of preference relations definable by first-order formulas. In terms of preferences aggregation [3] handles different types of relationships that take into account the relative importance of the interacting agents, whereas [4] extended CP nets to handle preferences of several agents based on the notion of voting. In [5], a stimulating survey of opportunities and problems in the use of preferences, reliant on AI techniques, is presented.

2 APPROACH OVERVIEW

This paper presents some of the concepts presented in [6], which proposes an adapted version of the contradiction removal method defined for the class of normal logic programs plus integrity constraints, as proposed in [7], which transforms a given program into its revisable form. Then, it follows that each of the stable models of the revisable program specifies which preferences minimally need to be yielded or added for that program model to be consistent. Finally, from all preference revision minimal stable models, the one that is *fairest* (i.e. seeking highest cooperation while minimising losses) is selected and proffered to the end-user.

The proposed method involves the following steps: (1) set the preferences for each agent; (2) integrate all agents' preferences into a single *merged program*; (3) extend the *merged program* into a *revisable program* form; (4) determine the minimal stable models of the *revisable program* so as to generate all possible revision hypotheses; (5) from the set of minimal stable models select the most *fair* one, taking into account present and past iterations; and (6) repeat the whole process as desired.

3 BACKGROUND CONCEPTS

3.1 STABLE MODELS

Let $\mathcal{L}$ be a first order language. A literal in $\mathcal{L}$ is an atom A in $\mathcal{L}$ or its default negation $not\, A$. A Normal Logic Program (NLP) P over $\mathcal{L}$ (sometimes simply called program) is a finite set of rules of the form $H \leftarrow B_1, B_2, \cdots, B_n, not\, C_1, not\, C_2, \cdots, not\, C_m$, with $(n \geq 0, m \geq 0)$ comprising positive literals H, and B_i, and default literals not C_j. $\mathcal{L}_P$ denotes the language of P.

Models are 2-valued and represented as sets of those positive literals which hold in the model. The set inclusion and set difference are with respect to these positive literals. Minimality and maximality too refer to this set inclusion.

Definition 1. *[8] Let P be a NLP and I a 2-valued interpretation. The* GL-transformation of P modulo I *is the program* $\frac{P}{I}$*, obtained from* P *by (1) removing from* P *all rules which contain a default literal* $not\, A$ *such that* $A \in I$ *and (2) removing from the remaining rules all default literals. Since* $\frac{P}{I}$ *is a definite program, it has a unique least model* J*. Define* $\Gamma_P(I) = J$*. Stable models are the fixpoints of* Γ_P*, and they do not always exist (namely when a finite program contains loops over an odd number default negations).*

3.2 PREFERENCE RELATION

Given a set N, a preference relation $\succ$ is any binary relation on N. Given two elements a and b in N, $a \succ b$ means that a is preferred to b. We assume that N contains at least two elements. We restrict $\succ$ to satisfy the properties of a strict partial order, which involves:

$$\begin{aligned} Irreflexivity &: \forall x, x \not\succ x \\ Asymmetry &: \forall x \forall y, x \succ y \Rightarrow y \not\succ x \\ Transitivity &: \forall x \forall y \forall z, (x \succ y \wedge y \succ z) \Rightarrow x \succ z \end{aligned}$$

Let us call the above properties integrity constraints, which are employed to detect preference contradictions (others could be added); they can be described in logic programming as follows:

$$\bot \leftarrow p(x,x). \qquad \bot \leftarrow p(x,y), p(y,x). \qquad \bot \leftarrow p(x,y), p(y,z), not\, p(x,z).$$

where x, y, and z are variables, $\bot$ represents a contradiction, when present in a model of a program, and $p(a,b)$ represents a preference of type $a \succ b$. An agent i can define its preferences by adding facts of type $p_i(x,y)$. As previously stated, the preferences of all agents have to be merged into a *merged program*. As a consequence, we have to add the following clause to the integrity constraints set: $p(x,y) \leftarrow p_i(x,y)$. Thus, there will exist a p predicate rule for each corresponding p_i predicate. If one of the integrity constraints rules succeeds, then there is a contradiction in the *merged program*.

3.3 DECLARATIVE DIAGNOSIS

In this section we adopt the definitions set forth in [6]. Given a contradictory program P, i.e. with a contradictory stable model, to revise (i.e. eliminate) the contradiction symbol ($\bot$) from its models we need to modify P by adding and removing rules. In this framework, the diagnostic process reduces then to finding such combinations of rules. To specify which rules in P may be added or removed, we assume given a set C of predicate symbols in $\mathcal{L}_\mathcal{P}$. C induces a partition of P into two disjoint parts: a changeable one P_c and stable one P_s. P_c contains the rules in P defining predicate symbols in C, while P_s contains the rules in P defining predicate symbols not belonging to C. P_c is the part subject to the diagnosis process.

Definition 2. *Let P be a program and C a set of predicate symbols in $\mathcal{L}_\mathcal{P}$. Let D be a pair $\langle U, I\rangle$ where U is a set of atoms, whose predicate symbols are in C and $I \subseteq P_c$. Then, D is a diagnosis for P iff $(P - I) \bigcup U \not\models \bot$. The pair $\langle\{\},\{\}\rangle$ is called empty diagnosis.*

Intuitively, a diagnosis specifies the rules to be added and removed from the changeable part of P to revise its contradiction $\bot$. In order to minimise the number of changes, one should consider minimal diagnosis.

Definition 3. *Let P be a program and $D = \langle U, I\rangle$ a diagnosis for P. Then, D is a minimal diagnosis for P iff there exists no diagnosis $D_2 = \langle U_2, I_2\rangle$ for P such that $(U_2 \bigcup I_2) \subseteq (U \bigcup I)$.*

Let us now clarify what P_c and P_s are. P_s (the stable partition of program P) refers to the integrity constraints and preferences that agents consider as non-negotiable (i.e. those preferences agents do no accept to discard at the cost of rejecting further cooperation). In this work we only considered negotiable preferences. On the other hand, P_c refers to the agents' preferences that will be subject to revision (i.e. that can be yielded). As an example, let us assume that there are two agents, one with a single preference and another one with two preferences; these define P_c:

$$P_c = \begin{pmatrix} p_1(a,b) \\ p_2(b,a) \\ p_2(c,b) \end{pmatrix} \; P_s = \begin{pmatrix} \bot \leftarrow p(x,x) \\ \bot \leftarrow p(x,y), p(y,x) \\ \bot \leftarrow p(x,y), p(y,z), notp(x,z) \\ p(x,y) \leftarrow p_1(x,y) \\ p(x,y) \leftarrow p_2(x,y) \end{pmatrix}$$

which contains those rules (i.e. preferences) that can be removed as well as those rules that have been added so as to guarantee consistency. In this case, one possibility would be to remove $p_1(a,b)$ because it contradicts $p_2(b,a)$ (non-reflexivity) whereas a rule $p(c,a)$ would be added to ensure transitivity. Another possibility, which is simpler and so preferable, would be to remove $p_2(b,a)$ and no further change would be required.

Now it is necessary to extend P so it has a suitable form for contradiction removal. To do that, we have to consider the *revisable set*, which includes default atoms, $not\ A$, whose CWA (Closed World Assumption) makes them initially true, can be revised by adding A to P. By so doing, one can disable rules having such defaults in their body, and, as a consequence, remove some contradictions. Also, the positive atom A, being initially false, can be made true by adding it as a fact. This will enable rules with A introduced in their body.

Definition 4. *The revisables of a program P is a subset of the set of atoms A (with $A \neq \bot$) for which there are no rules defining A in P.*

Definition 5. *Let P be a program and V a set of revisables of P. A set $Z \subseteq V$ is a revision of P with respect to V iff $P \bigcup Z \nvDash \bot$.*

Definition 6. *Let P be a program and C a set of predicate symbols in $\mathcal{L}_\mathcal{P}$. The* transformation *Γ that maps P into a program P' is obtained by applying to P the following two operations: (1) add $not\,incorrect(A \leftarrow Body)$ to the body of each rule $A \leftarrow Body$ in P_c and (2) add, for each predicate p with arity n in C, the rule $p(x_1, \cdots, x_n) \leftarrow uncovered(p(x_1, \cdots, x_n))$.*

In our example, the *revisable program* would be:

$$\Gamma(P) = \begin{pmatrix} \bot \leftarrow p(x,x) & p_1(a,b) \leftarrow not\,incorrect(p_1(a,b)) \\ \bot \leftarrow p(x,y), p(y,x) & p_2(b,a) \leftarrow not\,incorrect(p_2(b,a)) \\ \bot \leftarrow p(x,y), p(y,z), & p_2(c,b) \leftarrow not\,incorrect(p_2(c,b)) \\ \quad not\,p(x,z) & \\ p(x,y) \leftarrow p_1(x,y) & p_1(x,y) \leftarrow uncovered(p_1(x,y)) \\ p(x,y) \leftarrow p_2(x,y) & p_2(x,y) \leftarrow uncovered(p_2(x,y)) \end{pmatrix}$$

The transformation Γ preserves the truths of program P. Then, by adding instances of $incorrect/1$, one can eliminate unsound answers of predicate rules and, by adding instances of $uncovered/1$, one can complete predicates with missing answers.

4 PROPOSED APPROACH

The proposed approach considers that, in opposition to definition 6, the $uncovered$ literal is applied to $p_i/2$ instead of $p/2$. That is to say, that those added preferences relate to the group (i.e. $p/2$) and not to an agent i in particular. With this alteration, the number of even loops is reduced, and as a consequence complexity as well. In addition, the preference to be added is usually related to the group and not to an agent in particular. Let us imagine a scenario where one agent prefers a to b whereas another prefers b to c. So that *transitivity* holds in that scenario, it is necessary to state that a is preferred to c. However, this information does not refer to any agent in particular - it belongs to the group as a whole. If considered otherwise, the addition of a preference could not be exploited as a metric of *unhappiness*; how could one assume that an agent has been disadvantaged with the addition of a preference that might be true (i.e. the referred agent may in fact prefer a to c)?

The stable models of $\Gamma(P)$, minimal with respect to the revisables, that do not lead to a contradiction are potential candidates for the *best diagnosis*. Let us focus on how cooperation among agents is polarised by choosing which preferences have to be *minimally* yielded by each agent, *in a fair way*. From the set of such minimal stable models (i.e. in the set of possible minimal diagnoses), we have to select the *best diagnosis*.

Since the entire process describes a mental process to be reiterated each time a task and/or the environment require, we need to take into account all previous losses an agent had in prior iterations (i.e. by recording every time some agent had to *yield* a preference). Therefore, the *best diagnosis* will be one determined by following a criterion of *fairness*. This is achieved by a cost function that weighs a given diagnosis in a *fair* way, i.e. by taking into account previous decisions, is minimised over the whole set of available diagnoses.

4.1 *YIELDING* A PREFERENCE

Let us analyse what *yielding a preference* formally means.

Definition 7. *Let P be a program containing all preferences of all agents involved in a given iteration, $D = \langle U, I \rangle$ a diagnosis for P, $P_A \subseteq P$ a program with all preferences for agent A and $p(x, y)$ a preference of x over y. For each $incorrect(p(x, y)) \in I$, an agent A has 'yielded' a preference if $p_A(x, y) \in P_A$. For each $uncovered(p(x, y)) \in U$, an agent A has 'yielded to add' a preference.*

Computing the number of times an agent has to *yield* or *yield to add* a preference in a given diagnosis gives us an idea of how much displeased that agent will be with the solution trade-off. The presence of *yielding to add* preferences in a given solution may result in erroneous assumptions about others' preferences; as such, solutions including

such assumptions are to be penalised. Still, *yielding* a preference and *yielding to add* a preference have not the same weight in the unhappiness of the agent. Therefore, we consider different weights for each type, w_y and w_{ya} respectively. In the experimental part of this paper $w_y = 2$ and $w_{ya} = 1$ are considered; that is to say that, *yielding* a preference is twice worse than *yielding to add* a preference, which intuitively makes sense. The exact proportion should be subject to further analysis, but out of the scope of this paper.

Notice the cost of *yielding* a preference only affects the agent in question, whereas the cost of *yielding to add* a preference affects all agents in the same amount. This means that *yielding* a preference increases the *dispersion* of *unhappiness* among agents, whereas *yielding to add* preferences increases the average number of *unhappiness*.

4.2 DIAGNOSIS ASSESSMENT

The goal is to determine which diagnosis (i.e. which minimal stable model) is less *expensive* in terms of the "yieldings" agents have to endorse. In order to do so, the cost of a diagnostic D is assessed via a cost function. Two main components are considered, namely: the *average of yieldings*, $a(D)$, and the *dispersion of yieldings*, $d(D)$. The *average of yieldings* is defined by $a(D) = \frac{1}{n} \cdot \sum_{a \in A} \omega_{yield}(a, D)$, where D is the diagnostic in question, A is the set of agents involved in the process, n is the number of agents in A, and $\omega_{yield}(a, D)$ returns the total cost of agent a in diagnosis D.

The total cost of an agent a in a diagnosis D is $\omega_{yield}(a, D) = \omega_y \cdot n_y(a, D) + \omega_{ya} \cdot n_{ya}(a, D)$, where ω_y and ω_{ya} are the weights of *yielding* and *yielding to add* a preference, respectively, $n_y(a, D)$ and $n_{ya}(a, D)$ are the number of *yield* and *yield to add* preferences respectively in diagnosis D (see definitions 7).

The *dispersion* of a diagnosis D, $d(D) = \sqrt{\frac{1}{n} \cdot \sum_{a \in A} (w_{yield}(a, D) - a(D))^2}$, refers to the concept of standard deviation from statistics. It is worth emphasising the quadratic term in the standard deviation definition, which increases the cost of a diagnosis quadratically, in relation to its displacement from the average. Accordingly, as the *unfairness* of solutions increases, their related cost increases at a faster pace.

The standard deviation concept is used to measure how much *unfair* a diagnosis is for a given agent; in other words, the more disparate is the sacrifice of a given agent when compared to the average sacrifice, the more unfair the diagnosis is. The purpose of the cost function is to reduce the sacrifice of all agents as well as to avoid that some agents be much more sacrificed than the average. The next formula describes the minimising of the cost function in order to obtain the *best diagnosis* of the current iteration, $b_d[n]$:

$$b_d[n] = \min_D \left(\frac{w_{win}}{w_{all}} (\beta_d \cdot d(D) + \beta_a \cdot a(D)) \right) \quad (1)$$

where w_{win} is the quantity of accumulated victories of the agent less sacrificed (i.e. with smaller w_{yield}) in D, w_{all} is the quantity of accumulated victories the agent with greater amount of accumulated victories has, and β_a and β_d are weights. An agent accumulates a victory each time it is the one with smaller ω_{yield} in the *best diagnosis*

of a given iteration. It is assumed that the preferences of agents may change between iterations.

The $\frac{w_{win}}{w_{all}}$ component endows the system with memory, which allows to take into account previous iterations. This way, the selection of the *best diagnosis* takes into consideration that some agents may have been more sacrificed in the past than others. Intuitively, the possibilities of accepting the diagnosis in question as the *best diagnosis*, increase if the winning agent has less victories than the one with more victories, proportionally. As a result, all agents gradually converge into an homogeneous number of victories.

5 EXPERIMENTAL RESULTS

The proposed method has been implemented in the XSB-Prolog[3] using the XSB-XASP Package[4], which implements the Answer Set semantics (cf. [9] for other application examples) and allows computing the stable models of a given program.
In this section we will briefly go over some implementation details. To do so, some (simplified for presentation purposes) XSB-PROLOG code will be exhibited. With the purpose of creating several scenarios that can be covered by different stable models, it is necessary to create even loops over default negation (where $tnot$ is XSB-Prolog's tabled not), such as:

```
covered(X):-tnot(uncovered(X)).  incorrect(X):-tnot(correct(X)).
uncovered(X):-tnot(covered(X)).  correct(X):-tnot(incorrect(X)).
```

where each loop allows for its two opposite stable models solutions. The integrity constraints have been defined as follows (a two agent scenario has been considered):

```
false(X,_,_) :- p(X,X).       false(X,Y,_) :- p(X,Y), p(Y,X).
false(X,Y,Z) :- p(X,Y), p(Y,Z), X \= Z, tnot( p(X,Z) ).
```

Agents' preferences can be defined in this manner (note that $incorrect/1$ literals have already been added):

```
p(son,X,soaps) :- tnot( incorrect(p(son,X,soaps)) ).
p(son,cinema,X) :- tnot( incorrect(p(son,cinema,X)) ).
```

which states that the son prefers to see anything in the TV to soaps, and prefers cinema to anything else. This code allows creating scenarios where new preferences are added (i.e. where $uncovered/1$ literals are inserted):

```
p(X,Y):-p(agent_1,X,Y).                p(X,Y) :- p(agent_2,X,Y).
p(X,Y):-not p(agent_1,X,Y),not p(agent_2,X,Y),uncovered(p(X,Y)).
```

The following code generates several hypotheses (i.e. stable models), which are then considered for the selection of the *best diagnosis*), by performing a call to the XSB-XASP package so it computes in Res the stable models with *ok* as the top goal; *ok* succeeds only if a given scenario does not fail to cope with each integrity constraint:

[3] http://xsb.sourceforge.net
[4] http://xsb.sourceforge.net/packages/xasp.pdf

```
ok :- tnot( notok ).
notok :- option(X), option(Y),option(Z), notok(X,Y,Z).
notok(X,Y,Z) :- false(X,Y,Z).
pstable_model(ok,Res,1) % Top goal call
```

An everyday multi-agent conflict resolution scenario is the one we can find in our houses when all family members have to cooperate (at least it is so expected) to decide on which TV programme they will watch. A simple version of such a scenario has been implemented with these characteristics:

Involved Agents Father, mother, and son;
Available TV Programmes Soaps, news, cinema, and documentaries;
Weights $\beta_1 = 1$ and $\beta_2 = 3$. β_2 should be greater than β_1 in order to guarantee that the solutions converge to a small amount of yielded preferences. As long as this proportion between the two parameters is kept, the final results are not very sensitive to parameter variation.
Son's preferences: $cinema \succ x$; $documentaries \succ news$
Mother's preferences: $soaps \succ x$; $cinema \succ news$; $documentaries \succ news$
Father's preferences: $x \succ soaps$; $news \succ x$; $documentaries \succ news$

Starting with the accumulated victories as $(mother, father, son) = (1, 1, 1)$, the selected TV program (such that no other is preferred to in the *best diagnosis*) is cinema, the accumulated victories change to $(mother, father, son) = (1, 1, 2)$, and the explanation is ("inc" represents "incorrect"):

```
[inc(p(son,cinema,cinema)),inc(p(father,news,cinema)),
 inc(p(father,news,news)),inc(p(mother,soaps,cinema)),
 inc(p(mother,soaps,news)),inc(p(father,soaps,soaps)),
 inc(p(mother,soaps,soaps))]
```

The son is the one yielding less preferences and as a result he wins. Cinema is the only TV programme such that no other is preferred to in the *best diagnosis* and so it is selected. All preferences leading to contradiction are said to be incorrect. For instance, the son prefers cinema to anything, but he may not prefer cinema to cinema itself (irreflexivity). The father prefers news to anything else, but since the son prefers cinema to anything else (including news), the father has to yield the preference of news over cinema in particular (asymmetry). Other solutions exist, such as adjusting son's preferences so the father could see what he most prefers. However, those solutions are not so balanced (i.e. *fair*) than the selected one, in the current context.

In a subsequent iteration the selected TV programme is soaps, the accumulated victories become $(mother, father, son) = (2, 1, 2)$, and the explanation is:

```
[inc(p(son,cinema,cinema)),inc(p(father,cinema,soaps)),
 inc(p(son,cinema,soaps)),inc(p(father,news,cinema)),
 inc(p(father,news,news)),inc(p(father,news,soaps)),
 inc(p(father,soaps,soaps)),inc(p(mother,soaps,soaps))]
```

Although seeing soaps increases the total number of preferences to be yielded (i.e. now the son has to yield two preferences), soaps is preferred because the mother was in

disadvantage in terms of victories. If the past had not been considered, at this iteration the result would be the same as in the previous iteration. In a third iteration, the selected TV programme is soaps, the accumulated victories become $(mother, father, son) = (2, 2, 2)$ and the explanation is:

```
[inc(p(son,cinema,cinema)),inc(p(son,cinema,news)),
 inc(p(mother,cinema,news)),inc(p(son,cinema,soaps)),
 inc(p(father,news,news)),inc(p(mother,soaps,cinema)),
 inc(p(mother,soaps,news)),inc(p(father,soaps,soaps)),
 inc(p(mother,soaps,soaps))]
```

A new *best diagnosis* is produced, which is caused by the disadvantage of the father. From this sequence of three examples, one can observe the evolution of the solution so as to maintain everybody happy.

In a second experiment a new son's preference, $x \succ soaps$, which is going to augment the displeasure in watching soaps (the father also prefers anything to soaps), has been added. Resetting the accumulated victories to $(mother, father, son) = (1, 1, 1)$ we obtain cinema as the selected TV programme, $(mother, father, son) = (1, 1, 2)$ as the new accumulated victories, and the following explanation:

```
[inc(p(son,cinema,cinema)),inc(p(father,news,cinema)),
 inc(p(father,news,news)),inc(p(mother,soaps,cinema)),
 inc(p(mother,soaps,soaps)),inc(p(father,soaps,soaps)),
 inc(p(son,soaps,soaps)),inc(p(mother,soaps,soaps))]
```

In line with the victory of the son, cinema is selected. Running a new the selected TV program is cinema, $(mother, father, son) = (2, 1, 2)$ is the new set of accumulated victories, and the explanation is as follows:

```
[inc(p(son,cinema,cinema)),inc(p(father,news,cinema)),
 inc(p(father,news,news)),inc(p(father,news,soaps)),
 inc(p(son,news,soaps)),inc(p(mother,soaps,cinema)),
 inc(p(father,soaps,soaps)),inc(p(son,soaps,soaps)),
 inc(p(mother,soaps,soaps))]
```

This time the mother wins since she was at a disadvantage in terms of victories; nevertheless, cinema is seen rather than soaps. In a last iteration we get news as the selected programme, $(mother, father, son) = (2, 2, 2)$ as the new accumulated victories, and the following explanation:

```
[inc(p(son,cinema,cinema)),inc(p(son,cinema,news)),
inc(p(mother,cinema,news)),inc(p(father,news,news)),
inc(p(mother,soaps,cinema)),inc(p(mother,soaps,news)),
inc(p(father,soaps,soaps)),inc(p(son,soaps,soaps)),
inc(p(mother,soaps,soaps))]
```

The father wins because he was at a disadvantage in terms of victories and, as a consequence, news is the selected programme. This experience demonstrates that it is not enough to be disadvantaged in terms of victories to obtain what one desires. More concretely, the mother was at a disadvantage and, as a result, she won the second iteration; still, soaps was not the chosen programme category. This occurred because the son also made clear that he would prefer to see anything but soaps.

6 CONCLUDING REMARKS

A method for preference revision in a multi-agent scenario has been presented. Instead of considering explicit priorities among agents and/or preferences, the method proposes a dynamic approach. A cost function that considers generic features of the solution (e.g. the quantity of preferences yielded by agents) has been employed to obtain a general approach, avoiding parameters too application dependent. Introducing memory in the revision process enables the emergence of cooperation as iterations unfold. Cooperation shows up in the form of an homogeneous amount of victories amongst all agents.

Assessing which preferences one should add or remove in the way prescribed in this paper, allows us to enact a flexible method for preference revision. If instead priorities among preferences are to be considered as the sole method for preference revision, intensive and tedious parameter tuning is going to be required so as to guarantee that preferences are conveniently revised. In such a memory-less solution, the system is not able to evolve towards cooperation as iterations unfold, resulting in fully deterministic and static solutions. On the contrary, we consider our approach of special interest for dynamic environments.

We have employed the two-valued Stable Models semantics to provide meaning to our logic programs, but we could just as well have employed the three-valued Well-Founded Semantics [10] for a more skeptical preferential reasoning. Also, we need not necessarily insist on a strict partial order for preferences, but have indicated that different constraints may be provided.

References

1. Chomicki, J.: Preference formulas in relational queries. ACM Transactions on Database Systems **28** (2003) 427–466
2. Andreka, H., Ryan, M., Schobbens, P.Y.: Operators and laws for combining preference relations. Journal of Logic and Computation **12** (2002) 13–53
3. Yager, R.R.: Fusion of multi-agent preference ordering. Fuzzy Sets and Systems **117** (2001) 1–12
4. Rossi, F., Venable, K.B., Walsh, T.: mCP nets: representing and reasoning with preferences on multiple agents. In: Procs. of the 19th Conf. on Artifi cial Intelligence, LNCS 749. (2004) 729–734
5. Doyle, J.: Prospects for preferences. Computational Intelligence **20** (2004) 111–136
6. Dell'Acqua, P., Pereira, L.M.: Preference revision via declarative debugging. In Bento, C. et al., ed.: Progress in Artifi cial Intelligence, Procs. 12th Portuguese Int. Conf. on Artifi cial Intelligence (EPIA'05), Covilhã, Portugal, Springer, LNAI 3808 (2005)
7. Pereira, L.M., Damásio, C., Alferes, J.J.: Debugging by diagnosing assumptions. In Fritzson, P., ed.: Procs. of the 1st Int. Workshop on Automatic Algorithmic Debugging (AADEBUG'93), Springer–Verlag, LNCS 749 (1993) 58–74
8. Gelfond, M., Lifschitz., V.: The stable model semantics for logic programming. In: Procs. of the 5th Int. Logic Programming Conf. (1998)
9. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge U.P. (2003)
10. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM **38** (1991) 620–650