Answer Set Programming and Combinatorial Voting

Rafał Graboś

University of Leipzig, Dept. of Computer Science,
Augustus Platz 10/11, 04109 Germany
grabos@informatik.uni-leipzig.de

Abstract. In this paper Logic Programming with Ordered Disjunction (LPOD), an extension of answer set programming for handling preferences, is used for representing and solving combinatorial vote problems. Various vote rules, used as procedures for determining optimal candidate for a group of voters, are defined by means of preference relations between answer sets of a program, representing a vote problem. Moreover, we present lexicographic extensions of some vote procedures, making these procedures more decisive.

1 Introduction

Vote theory is a well-known filed of research in social sciences, and voting problems have been extensively investigated in social choice community [7], [5], [11]. Roughly speaking, Social Choice Theory is concerned with the aggregation of individuals' preferences into a collective preference. Based on this social preference a collective decision can be taken. One of the most popular applications of Social Choice are voting scenarios. Here preferences (over candidates) are aggregated to elect one of the candidates. It is important to note that the set of candidates contains individuals and preferences of voters are expressed in terms of a preference relation (usually a weak order) or quantitatively, by means of a score function. However, the choice may depend on the method of aggregation. Methods used to find an optimal candidate are called vote rules (procedures) and much of work have been done to investigate properties of various types of such procedures. For example a (simple) majority voting rule may lead to a different candidate than application of a so called ``scoring rule", in which the voters may assign votes of different weight to different candidates.

Social choice methods are directly applicable, when candidates are individuals – then such candidates may be explicitly listed and ordered by voters. However, in some problems, listing and ordering of all possible candidates may be a complex and tedious task.

Consider as an example the following problem: voters have to agree on a common acceptable work-team containing experts of different domains (attributes), for instance in case of IT experts, domains may be: a set of programmers, net administrators, database administrators etc. In this case, each domain consists of individuals (experts of a domain) and a set of possible candidates (teams) may be equal to all possible combination of experts from all domains, i.e. it may be a Cartesian product of sets of the given domains. In case of our example, a possible candidate may be a

Dagstuhl Seminar Proceedings 05171
Nonmonotonic Reasoning, Answer Set Programming and Constraints
http://drops.dagstuhl.de/opus/volltexte/2005/262

work-team containing: John, who is programmer, Steven, who is database expert and Marry, who is net admin.

The similar problems are: committees' recruitment problems, product configuration problems, and multicriteria decision problems in general. It is well-known fact that in most of these problems, the set of candidates has a size exponential in the number of the attributes being considered (domains of experts in our case), and it is not reasonable asking the voters to rank all candidates directly. The problem is more complex, if the domains are preferentially dependent, e.g. when preferences over a set of values of one attribute are dependent on values of different attributes.

In order to avoid the above difficulties, various preference representation languages have been investigated in AI: logic-based approaches [9],[10], CP-nets [1] etc. The main advantage of such languages is that they enable a concise and succinct representation of the preference structures such that a preference ranking of the alternatives is deduced automatically. Moreover, preferences in the languages are expressed in a more human-like form, close to natural language expressions, hence providing a good readability and simplicity.

In this paper, we study logic programming approach to combinatorial voting. Logic programming with ordered disjunction has been invented by Brewka [2] as an extension of answer set programming [11] to represent priority among literals and rules in a program. From a practical perspective, ordered disjunction has been recognized as a useful tool for modeling and solving a wide range of knowledge representation problems [2], [3], for instance qualitative decision making under uncertainty [6], explaining unexpected observations that would otherwise lead to inconsistency [13], among others.

We show how LPOD may be used to represent and solve vote problems, in which a set of candidates has a combinatorial structure. Note that in case of combinatorial voting two complex problems must be solved: combinatorial search problem, since a set of possible candidates is not given explicitly and must be generated from a set of attributes' values and optimization problem, since preferences of voters must be aggregated into a group preference. Answer set programming with preferences seems to be a promising approach to such vote problems from at least two reasons: its applicability to solve combinatorial, search and constraint satisfaction problems, which has been exampled by means of various AI problems (diagnosis, planning, cryptography etc.). Secondly, because LPODs have been recognized as an expressive preference representation language, being able to express defeasible, conditional, partial preferences as well as meta-preference information.

2 Formal Background

2.1 Answer Set Programming

Answer Set Programming (ASP) is a declarative approach to knowledge representation and reasoning [11]. Consider a propositional language L, with atomic symbols called atoms. A literal is an atom or a negated atom (by classical negation $\neg$). Symbol *not* is called epistemic negation and the expression *not a* is true, if there is no reason

to believe, that a is the case. The symbol $\vee$ is called epistemic disjunction. Formally, rule r is an expression of the form:

$$c_1 \vee \ldots \vee c_k \leftarrow a_1, \ldots, a_m, \text{ not } b_1, \ldots, \text{ not } b_n \quad \textbf{(1)}$$

where $k \geq 0$, $n \geq m \geq 0$, c_i, a_l, b_k are literals, $Body^-(r) = \{b_{m+1}, \ldots, b_n\}$, $Body^+(r) = \{a_1, \ldots, a_m\}$ are conjunctions of literals and the disjunction $\{c_1 \vee \ldots \vee c_k\}$ is a $Head(r)$ of the rule r. A rule with an empty $Head$ ($\leftarrow Body$) is usually referred to as an integrity constraint. A logic program is a finite set of rules.

Intuitively, the above rule r means that if the $Body^+(r)$ of that rule is believed to be true and it is not the case that $Body^-(r)$ is believed to be true, then at least one literal of $Head(r)$ must be believed to be true.

The semantics of ASP is defined by means of minimal set of literals satisfying all rules of the program. Let us assume now, that Lit_P is a set of all literals being present in the extended logic program P and I is an interpretation of P, $I \subseteq Lit_P$. We say that a set of literals I satisfies a rule of the form (1), if $\{a_1, \ldots, a_m\} \subseteq I$ and $\{b_{m+1}, \ldots, b_n\} \cap I = \emptyset$ imply that $\{c_1, \ldots, c_k\} \cap I \neq \emptyset$. The Gelfond-Lifschitz (GL) transformation of P with respect to I is a positive logic program P' which is obtained in two steps:

- deletion of all rules r of P, for which $Body^-(r) \cap I \neq \emptyset$
- deletion of the negative bodies ($Body^-(r)$) from the remaining rules of P

Then, I is an answer set of the logic program P, if I is a minimal model of the positive (without not) logic program P'; i.e. I is a minimal set of literals satisfying every rule in P' or if I contains a pair of complementary literals l and $\neg l$, then $I = Lit_P$.

Although answer set programs are basically propositional, it is possible to use rule schemata containing variables. These schemata are representations of their ground instances, and answer set solvers, like Smodels [17], use intelligent ground instantiation techniques before the actual answer set computation takes place.

2.2 Logic Programming with Ordered Disjunction

Consider now an extended logic program (with two negations), where the ordered disjunction $\times$ is allowed in *head* part of a rule. A logic program with ordered disjunction (*LPOD*), introduced in [2] consists of rules of the form:

$$c_1 \times \ldots \times c_k \leftarrow a_1, \ldots, a_m, \text{ not } b_1, \ldots, \text{ not } b_n \quad \textbf{(2)}$$

where $k \geq 0$, $n \geq 0$, $m \geq 0$, c_i, a_l, b_k are literals and the ordered disjunction $\{c_1 \times \ldots \times c_k\}$ is $Head(r)$ of the rule r. The rule is originally to be read: if possible c_1, if c_1 is not possible, then $c_2, \ldots$, if all of $c_1, \ldots, c_{k-1}$ are not possible, then c_k.

Answer set semantics is submitted to the *LPOD*. In order to use the standard ASP semantics, a split technique w.r.t. *LPOD* rules is applied, resulting in programs without the ordered disjunction[1]. In order to distinguish, which answer set is preferred one, the notion of the degree of satisfaction of an ordered disjunctive rule by answer set is introduced. Let S be an answer set of an *LPOD* P. An ordered disjunctive rule r:

[1] Details may be found in [3].

$$c_1 \times \ldots \times c_k \leftarrow a_1, \ldots, a_m, \text{ not } b_{m+1}, \ldots, \text{ not } b_n \quad (3)$$

is satisfied by S to degree:

- 1, *if* $a_j \notin S$ *for some* j *or* $b_i \in S$, *for some* i
- d $(1 \leq d \leq k)$, *if* $a_j \in S$ *for all* j, *and* $b_i \notin S$, *for all* i, *and* $d = min\ \{r \mid c_r \in S\}$.

Note that the degrees of satisfaction are treated as penalties, the smaller the degree the better the answer set is. Moreover, priority (meta-preferences) between preference rules can be expressed with the meaning: in case when it is not possible to satisfy all rules to the lowest degree, rules with the higher priorities must be satisfied first.

A problem to be solved is represented as a *LPOD* and answer sets of the program are ranked, according to the degrees of satisfaction of ordered disjunctive rules. In this way a global ranking of answer sets is obtained[2]. The following criteria have been proposed in [3] to build this ranking: cardinality optimal criterion- maximizing a number of rules satisfied to the lowest degree, inclusion optimal criterion, based on set inclusion of the rules satisfied to the certain degree and Pare*o* optimal criterion favoring the answer set satisfying all ordered disjunctive rules not worse, than any other answer set does, and one rule strictly better.

Consider as an example an *LPOD P*, representing a preferred dessert:

coffee × *tea.*

ice-cream × *pancake* × *tiramisu.*

← *coffee, pancake.* ¬ *ice-cream.*

Cardinality preferred answer set of P is $S_1 = \{coffee, tiramisu\}$, since only S_1 satisfies the rule to degree 1 (the first rule), while multiple Pareto and Inclusion optimal answer sets are obtained: $S_1 = \{coffee, tiramisu\}$ and $S_2 = \{tea, pancake\}$, since none of them satisfy all the rules best.

Computational complexity of *LPODs* under Pareto and inclusion preferences is proved to be in the same complexity class as disjunctive logic programs, namely in $\sum_P^2$-complete, while in Δ^2_P under the cardinality criterion.

Psmodels is a prototype implementation of logic programming with ordered disjunction under the above criteria[3]. Since in this paper several new criteria, suitable for voting applications are defined, they may be implemented in a similar way as presented in [3].

2.3 Extended LPOD

In the folllwong we propose a notion of an extended $LPOD^e$, which is a set of rules of the form:

$$C_1 \times \ldots \times C_k \leftarrow a_1, \ldots, a_m, \textit{ not } b_1, \ldots, \textit{ not } b_n.$$

where a_j and b_k are literals and C_i is more complex formula where conjunction:$\{a_1,\ldots,a_n\}$ or disjunction: $\{a_1 \vee \ldots \vee a_n\}$ of literals may appear. We use a meta-translation from an extended $LPOD^e$ to an *LPOD* such that the standard formal semantics is used to a resulting program. Instead of giving a formal description of the

[2] Different preference handling approaches in ASP may be found in [5].

[3] http://www.tcs.hut.fi/Software/smodels/priority/

translation, we exampled it as follows: let's assume that $LPOD^e$ P contains an extended rule r of the form:

$(a_1 \vee \ldots \vee a_n) \times (b_1, \ldots, b_m) \vee (c_1, \ldots, c_j) \times head \leftarrow body.$

Where a_j, b_k and c_i are literals, "," is conjunction and $\vee$ denote disjunction. Then, in the first step the rule r is translated into a program:

$A \times (B \vee C) \times head \leftarrow body.$
$a_1 \vee \ldots \vee a_n \leftarrow A.$
$b_1 \leftarrow B.$
…
$b_m \leftarrow B.$
$c_1 \leftarrow C.$
…
$c_j \leftarrow C.$

which is finally translated into a standard $LPOD$ P' containing no extended syntax:

$A \times (BE) \times head \leftarrow body.$
$a_1 \vee \ldots \vee a_n \leftarrow A.$
$B \vee E \leftarrow BE.$
$b_1 \leftarrow B.$
…
$b_m \leftarrow B.$
$c_1 \leftarrow C.$
…
$c_j \leftarrow C.$

where C, B, E, $BE \notin Lit(P)$ are new atoms.

Since the syntax of a translated program is now equivalent to the syntax of *LPOD*s, the formal semantics of *LPOD* is applied. Intuitively, by use if an extended programs, we can express preferential equality among literals of a program (weak order), as well as preference between sets of literals. Note that although one may extend *LPOD* syntax by adding conjunction and disjunction connectives in head part of rules directly, instead of using a meta-translation, it requires changing the semantics of the ordered disjunction.

In section 2.2 formal semantics of the ordered disjunction has been presented. However, when deal with partial preferences, this semantics leads to unintuitive results. A common problem in the filed of MCDM is of partial preferences given by DM. A traditional solution to this problem consists in using a partial order relation for expressing preferences. Originally, the ordered disjunction is a complete relation. Consider a problem where a set of alternatives $A = \{a_1, a_2, a_3, a_4\}$, is encoded by the $LPOD$ P_3:

c_1: $a_1 \times a_2$.
c_2: $a_3 \times a_4$.
c_3: $a_1 \times a_4$.
$1\{a_1, a_2, a_3, a_4\}1$.

Unfortunately, there is no answer sets for this program, since no set of literals satisfies all rules of the program. In order to avoid this effect we propose a technique from the field of logic programming. Let $A = \{a_1,...,a_n\}$ denote a set of literals representing

alternatives under consideration. For each rule r' representing a partial order over alternatives, a new rule is obtained as given in the below program *LPOD P′*:

$a_1 \times a_2 \leftarrow not\ r\_1.$
$r\_1 \leftarrow \text{not } a_1, \text{not } a_2.$
$a_3 \times a_4 \leftarrow \text{not } r\_2.$
$r\_2 \leftarrow \text{not } a_3, \text{not } a_4.$
$a_1 \times a_4 \leftarrow \text{not } r\_3.$
$r\_3 \leftarrow \text{not } a_1, \text{not } a_4.$
$1\{a_1, a_2, a_3, a_4\}1.$

The method requires putting the literals present in each of incomplete *LPOD*'s rule to the body of the *r_i* rule, denoting *i*th partial rule, thus the manual search for the literals not present in this rule is avoided. However, this method has a major drawback when assuming the standard formal semantics of ordered disjunctive rule. In case of the program *P′*, the cardinality optimal decision is alternative a_2, although it is dominated on the first criterion by a_1 and it is not even present in terms of the remaining criteria. In order to avoid such undesirable effects, we introduce a modified semantics of *LPOD* rule:

Let *S* be an answer set of an *LPOD P*. An ordered disjunctive rule *r*:

$$c_1 \times \ldots \times c_k \leftarrow a_1, \ldots, a_m, \text{ not } b_{m+1}, \ldots, \text{not } b_n \qquad (4)$$

is satisfied by *S* to degree:

- *ir, if* $a_j \notin S$ *for some j or* $b_i \in S$*, for some i*
- *d* $(1 \le d \le k)$*, if* $a_j \in S$ *for all j, and* $b_i \notin S$*, for all i, and* $d = min\ \{r \mid c_r \in S\}$.

Then, $ir = const.$ and $\forall\ r \in P$, $ir > \max\ \{k \mid c_k \in head(r)\}$. In other words, the value of *ir* is greater than any possible degree of satisfaction of any rule of *P* by any answer set of *P*. By this means it is guaranteed that answer sets not containing any literal of the head part of a rule gets the equal degree of satisfaction, but smaller than any satisfaction's degree of this rule by a relevant answer set. It is not the case, if we would put an arbitrary unsatisfied literal at the end of every partial preference rule, just in case if no literals of this rule are satisfied.

3 LPOD for Combinatorial vote problems

Representation of a vote problem G is divided into two parts:

1. Logic program *P*, representing the descriptive part of the problem G: (possible candidates, voters etc).
2. A set of vote profiles of the form: $V_i = (LPOD\ P_i, P)$, where P_i represents preferences of *i*th voter over candidates and *P* is of the form given above

3.1 Vote problem

Given a vote problem G, in which $A = (X_1,\ldots,X_n)$ is a set of attributes (dimensions, criteria) and $Dom = (D(X_1),\ldots, D(X_n))$ is a set of domains of the attributes, then a descriptive part of G is a logic program *P* consists of rules of the form:

$$r_i \qquad l\ \{a\ ,\ldots\ a_n\}u \leftarrow$$

where rule r_i indicates ith attribute $\{a_1,\ldots,\ a_n\}$ are value atoms, representing values of the domain of ith attribute, i.e. $D(X_i) = \{a_1,\ldots,\ a_n\}$, l and u are the upper and the lower bounds expressing desirable constraints on number of values of ith attribute. Intuitively, a rule of the form above represents for each attribute the domain of its possible values, while the upper and the lower bounds determine constraints on number of values. Note that the number of rules is equal to the number of attributes, i.e. the cardinality of the set A.

Proposition 1. Set of candidates C is a set of interpretations of the logic program P satisfying all rules of P. Therefore, $C = Ans(P)$.

Let's assume that $Vot = \{1,\ldots,m\}$ is a set of voters under consideration, and $A = (X_1,\ldots,X_n)$ is a set of attributes with the domains $Dom = (D(X_1),\ldots,\ D(X_n))$, where $D(X_i) = \{a_1,\ldots,\ a_n\}$ represents values of the domain of an attribute i and $P = \{r_1,...,\ r_n\}$ is a logic program consists of rules representing domains of the attributes and constraints on the domains. Answer sets of the program constitutes a set of possible candidates. In order to find a common acceptable candidate, preferences of voters must be taken into account. As already mentioned, voters are asked to express preferences not over elements of a set of candidate, which is in our case a set of answer sets of the program P, but over elements of sets of domains of attributes. In other words, a voter orders values of some attributes, possibly expressing some preferential dependencies between different domains.

Formally, preferences of voter i are represented by rules of the form:

$$C_1 \times\ldots\times C_k \leftarrow a_1,\ \ldots,a_m\ ,\ \text{not}\ b_1\ ,\ \ldots\ ,\ \text{not}\ b_n$$

where $1 \le k$ and $m \ge 0$ and $n \ge 0$, C_i may be of the disjunctive form only and for all i, $C_i \subseteq D(X_k)$ represent disjunctive sets of values literals of a domain D and $D(X_k) \subset Dom$ but $a_k\ ,\ b_j \in \cup\ Dom$ are values literals of arbitrary domains. Intuitively, the head part of the rule represents preference order of value of an attribute, while the body of this rule is satisfied. Note that no values have to be totally ordered; hence partial preferences can be represented.

Each voter is asked to specify his preferences and then they are represented by rules of the form given above. The next task is to aggregate such individual preference into a social preferences in order to obtain optimal candidate. Most of vote rules, used for aggregation purposes, assume counting the number of voters in order to determine optimal candidate. In the context of *LPOD* approach, preference criteria, used to identify a ranking of answer sets of a program, rely on the notion of the degree of satisfaction of rule by answer set and they exploit this concept to count the number of rules satisfied to a certain degree.

The problem with such method is the following: if possibly partial and conditional preferences of a voter are expressed by *LPOD*'s rules, trivially the number of rules does not have to be equal to the number of voters, i.e. number of rules should not affect a result of voting.. Therefore, instead of one aggregation problem, two such problems arise:

- aggregation of each voter's preferences over values of attributes into a ranking of candidates
- aggregation of individual preference ordering of candidates into a common acceptable preference ordering of candidates

Therefore we assume that each preference profile is represented as a separated *LPOD* and the number of such programs is equal to the numbers of voters under consideration. Formally, $V_i = (LPOD\ P_i, P)$ denotes a vote profile of ith voter, containing the descriptive part, constraining all configurations of values of attributes to admissible ones, and the preference part, consisting in rules expressing preferences of voters over values of attributes. Note that the following relation holds:

$$Ans(V_i) \subset Ans(P) \text{ for all } i$$

Then each *LPOD*, representing a voter's preference profile, leads to a ranking of its answer sets, thus a weak order over a set of candidates, according to a given vote method is obtained. By use of this procedure we obtain partial rankings of candidates, where the number of these rankings is equal to the number of voters. Then, a crucial task is to aggregate such rankings into one common ordering of candidates. To do this, we define below some vote rules in the context of LPOD approach.

3.2 Voting procedure

The procedure is as follows, given a logic P representing a problem's description and a preference profile V_i, for all i do:

1. Compute a ranking of answer sets of V_i, representing a ranking of candidates of a vote problem: if $Ans(V_i)$ denotes answer sets of V_i, then

$$S_1 \geq \ldots \geq S_n$$

where $S_j \in Ans(V_i)$ for all j, $n = | Ans(V_i) |$ represents a weak order of answer sets of V_i obtained by use of a particular preference criteria (e.g. lexicographic ordering). The problem of appropriate preference criterion used for aggregating preference rules to ordering of answer sets depends on the commensurability assumption, i.e. if degrees of satisfaction of rules by an answer set are commensurate across the rules. Let's assume that we use lexicographic way of ordering and exploit the cardinality criterion. However, any other criterion may be applied instead.

2. The order of answer sets of V_i: $S_1 \geq \ldots \geq S_n$, s.t. $S_i \in Ans(V_i)$ for all i, is the order of sets of literals of the form:

$$S_1 = \{a_1,\ldots,a_i\}_1 \geq \ldots \geq S_n = \{a_1,\ldots,a_i\}_n$$

Since each answer set contains literals interpreted conjunctively, we can represent such conjunctions by an extended ordered disjunctive rule, containing these conjunctions as its elements:

$$\alpha_i \qquad (a_1,\ldots,a_i)_1 \times \ldots \times (a_1,\ldots,a_i)_n \leftarrow$$

Then $LPOD^e\ P_v = \{\alpha_1,\ldots,\alpha_n\}$ represents a vote problem, where n denotes the number of voters under consideration. According to the meta-translation, given in the 3 section, the extended $LPOD^e\ P_v$ is translated into the normal $LPOD\ P_v$ containing rules with literals, instead of conjunctions, only.

It is clear that P_v contains rules representing partial orders of candidates w.r.t. all voters, therefore the aggregation of preferences over values of attributes into preference ordering of candidates has been done. Then, the original vote problem remains: the aggregation voters' preferences of candidates into a social preference of candidates.

Several vote rules have been define in the social choice literature. The idea is to define some of them in the context of LPOD formalism, such that we can take the ad-

vantage of properties of these rules in our framework. In this paper four vote procedures are being studied: Condorcet ruel, Pliurality procedure, Veto and Borda rules.

3.2.1 Plurality rule

Intuitively, plurality winner is a candidate which dominates other candidates for maximal number of voters. Plurality optimal selects the candidate maximizing the number of voters who ranked it at the first position. Then, candidate C_1 is plurality preferred to C_2 if C_1 is not dominated for n number of voters, C_2 is not dominated for m number of voters, and $n > m$.

Since very often multiple Plurality optimal candidates exist (ties), we propose a lexicographic extension of the plurality rule. Intuitively, candidate C_1 and C_2 are not dominated candidates for maximal number of voters but none of them is plurality preferred, we count the number of voters for whom C_1 and C_2 are ranked at the second positions (dominated by only one candidate), and so on. Although even this method does not guarantee a unique optimal candidate, it is much more decisive than pure plurality rule. Moreover, we propose the notion of weak optimal plurality candidate, such that a candidate is weak plurality winner if and only if there is no candidate, who is plurality preferred to it. Note that candidates being not dominated for the same number of voters may exist, however. In this case, multiple weak plurality optimal candidates may be obtained.

In order to definenthe notion of lex-plurality preferred answer set, we recall the notion of cardianlty preferred answer set, presented in [2]. Let's denote $S^i(P) = \{ r \in P | \deg(r) = i\}$.

Given *LPOD P* and $S_1, S_2 \in Ans(P)$, S_1 is g-plurality (global plurality) preferred to S_2 ($S_1 >_{g\text{-}p} S_2$) iff there is i such that $|S^i_1(P)| > |S^i_2(P)|$ and for all $j < i$, $|S^i_1(P))| = |S^i_2(P)|$. Intuitively, answer set is cardinality preferred if it satisfied maximal number of rules to the minimal degree.

Definition 2. Given *LPOD* P_D representing a vote problem D, and $S \in C$ of D, S is lex-plurality optimal candidate of D iff $S \in Ans(P_v)$ and $\forall S' \in Ans(P_v)$, $S >_c S'$.

Definition 3. Given *LPOD* P_D representing a vote problem D, and $S \in C$ of D, S is weak lex-plurality optimal candidate of D iff $S \in Ans(P_v)$ and $\forall S' \in Ans(P_v)$, $S >_{w\text{-}c} S'$.

Note that in order to obtain the following result, the modified semantics of ordered disjunction must be applied to P_v, since answer sets being irrelevant to some rules are not interpreted as obtaining 1 degree of satisfaction. By this means partial preferences are handled properly.

3.2.2 Condorcet rule

A candidate S is Condorcet winner if S can defeat all other candidates in separate pairwise contests, where defeating means that S is strictly preferred to any other candidate w.r.t. the maximal number of voters. Note that the Condorcet optimal candidate does not have to be not dominated solution for any of voter, since the rule counts the number of victories in pairwise comparisons only. Since Condorcet winner often does not exist (but if does, it is unique), we define the notion of weak Condorcet winner (WCW). A candidate S_1 is the WCW if it can defeat or draw all other candidates in pairwise comparison.

Let's assume that $\deg S_1(r)$ denotes degree of satisfaction of a rule r by an answer set S_1 and $C_P(S_1, S_2) = \{r \in P \mid deg S_1(r) < deg S_2(r)\}$. Then:

- Given *LPOD P* and $S_1, S_2 \in Ans(P)$, S_1 is *Condorcet* preferred to S_2 ($S_1 >_{c\text{-}p} S_2$) iff $|C_P(S_1, S_2)| > |C_P(S_2, S_1)|$.

Originally, the Condorcet winner is a candidate which defeats all other candidates in pairwise comparison, which means in our case, that:

- Given *LPOD* P_D representing a vote problem D, and $S \in C$ of D, S is Condorcet optimal candidate of D iff $S \in Ans(P_D)$ and $\forall S' \in Ans(P_D)$, $S >_{c\text{-}p} S'$.

However, very often no answer set satisfies this requirement, i.e. no answer set wins against all other answer set. Therefore, we introduce a weaker version of Condorcet winner by defining the Condorcet score of answer set S of *LPOD P* as follows: let $S \in Ans(P)$, then $C_P(S)$ denote the Condorcet score of S obtained from the below formula:

$$C_P(S) = |C_P^+(S)| - |C_P^-(S)|$$

where:

$C_P^+(S) = \{k \mid S >_{c\text{-}p} S_k$, where $S_k \in Ans(P)\}$

$C_P^-(S) = \{k \mid S <_{c\text{-}p} S_k$, where $S_k \in Ans(P)\}$

Intuitively, $C_P^-(S)$ denotes a set of answer sets of logic program P which are Condorcet preferred to answer set S , while $C_P^+(S)$ denotes a set of answer sets of P being Condorcet dominated by answer set S. Then, the Condorcet score of S is the ratio of cardinalities of the two sets. We define now the notion of weak Condorcet preferred answer set:

- Given *LPOD P* and $S_1, S_2 \in Ans(P)$, S_1 is weak *Condorcet* preferred to S_2 ($S_1 >_{wc\text{-}p} S_2$) iff $C_P(S_1) \geq C_P(S_2)$.

Gven the concept og weak Condorcet preferred answer set we can formally define the notion of weak Condorcet optimal candidate:

- Given *LPOD* P_D representing a vote problem D, and $S \in C$ of D, S is weak Condorcet optimal candidate of D iff $S \in Ans(P_D)$ and $\forall S' \in Ans(P_D)$, $S >_{wc\text{-}p} S'$.

3.2.3 Veto rule

Intuitively, for each candidate a veto score is computed such that candidate C obtains a score 1 w.r.t. voter V iff C dominates at least one other candidate. Hence, a candidate who maximizes the number of voters for whom it is not dominated is the Vote winner. Since counting the voters for whom a candidate dominates at least one candidate is inversely proportional to counting the voters for whom this candidate is dominated w.r.t. all other candidate (in this case veto to this candidate is given), we use the former procedure to find the Veto optimal candidate.

Let assume that S is an answer set of *LPOD P*, i.e. $S \in Ans(P)$ .Then the veto set of S w.r.t. P, denoted by $V_P(S)$, is obtained as follows:

$$V_P(S) = \{r \in P \mid \text{there is no } k \text{ s. t. } deg S(r) < deg S_k(r)\} \text{ and there is some } i \text{ s.t. } deg S_i(r) < deg S(r)\}.$$

Intuitively, the veto set of answer set S of *LPOD P* is the set of rules of P satisfied by S to the maximal degree. The second condition of the above definition guarantees, that given rule $r \in P$, which is satisfied by S to the maximal degree but there is no other $S' \in Ans(P)$ satisfying r strictly better, then r does not belong to the veto set of S, although the first condition of the definition is fulfilled.

Intuitively, such situation takes place if only one literal of the *head*(*r*) is applicable, i.e. only one literal of the *head*(*r*) belongs to any answer set of *P*. It is worth to note that the veto set of irrelevant answer set *S* is equal to the number of rules, w.r.t. which *S* is irrelevant.

We define now the notion of weak and strict veto preferred answer set:

- Given *LPOD P* and $S_1, S_2 \in Ans(P)$, S_1 is veto preferred to S_2 ($S_1 >_v S_2$) iff $| V_P(S_1) | < | V_P(S_2) |$.
- Given *LPOD P* and $S_1, S_2 \in Ans(P)$, S_1 is weak veto preferred to S_2 ($S_1 >_{w\text{-}v} S_2$) iff $| V_P(S_1) | \leq | V_P(S_2) |$.

Let's consider our approach to combinatorial vote problems. We want to find veto optimal candidates of a given veto problem. The following holds:

- Given *LPOD* P_D representing a vote problem *D*, and $S \in C$ of *D*, *S* is veto optimal candidate of *D* iff $S \in Ans(P_D)$ and $\forall S' \in Ans(P_D)$, $S >_v S'$.

In case of weak veto optimality, we change the relation from $>_v$ to $>_{w\text{-}v}$, as expected. Since many veto preferred answer sets is usually obtained, due to weak restrictions, we introduce now the lexicographic extension of the veto procedure. Intuitively, this vote method may be seen as inversion of lex-plurality procedure, since instead of counting rules satisfied to the minimal degree we start from the opposite direction and count the number of rules satisfied to the maximal degree. Threfore, we built a ranking of the most unpreferred answer sets, and then by reversing the scale we obtain lex-veto optimal candidates. Formally:

- Given *LPOD P* and $S_1, S_2 \in Ans(P)$, S_1 *lex-veto unpreferred* to S_2 ($S_1 >_{v\text{-}lex} S_2$) iff there is *i* such that $| S_1^i (P) | > | S_2^i (P) |$ and for all $j > i$, $| S_1^j (P) | = | S_2^j (P) |$.

Then, Given *LPOD P* and $S \in Ans(P)$, *S* is called pessimal answer set of *P* iff $\forall S' \in Ans(P)$, $S >_{v\text{-lex}} S'$.

Then, in order to find lex-veto optimal candidate, we must reverse the order of pessimality, such that the least pessimal answer set becomes the lex-veto optimal answer set.

- Given *LPOD* P_D representing a vote problem *D*, and $S \in C$ of *D*, *S* is lex-veto optimal candidate of *D* iff $S \in Ans(P_D)$ and $\forall S' \in Ans(P_D)$, $S' >_{v\text{-}lex} S$.

3.2.4 Borda rule

Borda voting procedure may be seen as a semi-qualitative vote method, since it consists in assigning to ranked candidates numbers, reflecting their positions, such that the lowest-ranked candidate receives 0 point, the next lowest 1 point etc. Then, the points are summed across all voters and the candidate with the most points wins.

It has been showed in the Social Choice literature that Borda system is very syntax-sensitive and therefore easy to manipulate due to:

- preference truncation (incomplete rankings) in which not ranked candidates by some voters may change positions of these candidates in an overall ranking
- dependence of irrelevant alternatives: by adding a new candidate, which does not change the voters rankings on remaining candidates, the result of voting may be changed
- syntax- manipulation: changing positions of some not highest-ranked candidates may change the effect of a vote procedure

Borda preference relation on answer sets may be define as foolows:

- Given *LPOD P* and $S_1, S_2 \in Ans(P)$, S_1 is *Borda* preferred to S_2 ($S_1 >_b S_2$) iff $\sum^{n}_{i=1} deg_{S_1}(r_i) < \sum^{n}_{i=1} deg_{S_2}(r_i)$, where $n = |P|$

In order to obtain a weaker notion of min-sum preferences, we change < by ≤.
Originally, Borda rule interpret not listed candidates by not assigning them any number reflecting their attractiveness for a voter. In our approach, degrees of satisfaction of rules by answer sets are treated as penalties, and the smaller the degree the better answer sets are. Since we assign to irrelevant answer sets the degree of satisfaction greater than any degree of satisfaction, but equal in all cases, we punish irrelevant answer sets instead of not rewarding them. Then:

- Given *LPOD* P_D representing a vote problem *D*, and $S \in C$ of *D*, *S* is Borda optimal candidate of *D* iff $S \in Ans(P_D)$ and $\forall S' \in Ans(P_D)$, $S >_b S'$.

3.3 Example

Let's consider the following vote problem as an example: 3 agents want to recruit IT experts from the following domains: net administrators, database admins, programmers. Since many applicants are applying for the position (let's assume that 3 applications for each position have been submitted), they must specify the number of experts of every domain to be recruited. For simplicity, let's assume that the agents want to have one representative of every domain in the workgroup, i.e. one programmer, one net administrator and one database expert. Therefore, a candidate is not a particular expert, but a group of individuals.

Then, they have some preferences concerning particular experts and possibly preference dependencies between some domains may hold. Since for each position 3 individuals are applying, the number of possible workgroup is 3^3, which amounts to 27 possible configurations of experts. In case of some preferential dependencies, the problem becomes even more complex.

Let's assume that following sets are given: $A = (N, B, Pr)$ is a set of attributes containing sets of *net administrators*, *database administrators* and *programmers* under considerations, respectively with the following domains: $D(N) = \{a, b, c\}$, $D(B) = \{e, f, g\}$ and $D(Pr) = \{h, j, k\}$, hence the set of domains is $Dom = (D(N), D(B), D(Pr))$. Moreover, a set of voters $Vot = (1, 2, 3)$ is given. According to the two layers architecture of our framework, we represent the descriptive part of the problem as a logic program *P* containing 3 rules, since $|A| = 3$, of the form:

r_1 $\quad 1\{a, b, c\}1 \leftarrow$
r_2 $\quad 1\{e, f, g\}1 \leftarrow$
r_3 $\quad 1\{h, j, k\}1 \leftarrow$

Then, the set of possible candidates of the vote problem is the set of answer sets of the above program, i.e. $C = Ans(P)$. In fact, we do not have to list all such candidates, due to the compact preference language enabling concise representation of preference structures.

Next step is to represent preferences of voter in vote profile programs such that a vote profile of *i*th voter is the pair: $V_i = (LPOD\ P_i, P)$ containing descriptive part of the problem and preferences of *i*th voter. The preference layer is as follows:
Let's assume that the following preference profiles are given:

V_1:

$a \times b \leftarrow$
$e \times f \leftarrow a.$
$j \times h.$
$r_1 \quad 1\{a, b, c\}1 \leftarrow$
$r_2 \quad 1\{e, f, g\}1 \leftarrow$
$r_3 \quad 1\{h, j, k\}1 \leftarrow$

V_2:
$b \times a \times c \leftarrow$
$h \times j \leftarrow a.$
$f \times e \leftarrow$
$r_1 \quad 1\{a, b, c\}1 \leftarrow$
$r_2 \quad 1\{e, f, g\}1 \leftarrow$
$r_3 \quad 1\{h, j, k\}1 \leftarrow$

V_3:
$b \times a \leftarrow h.$
$f \times e \leftarrow j.$
$h \times j \leftarrow$
$r_1 \quad 1\{a, b, c\}1 \leftarrow$
$r_2 \quad 1\{e, f, g\}1 \leftarrow$
$r_3 \quad 1\{h, j, k\}1 \leftarrow$

- $Ans(V_1)$ = { $\{a, e, j\}$, $\{b, f, j\}$,$\{b, g, j\}$,$\{b, e, j\}$, $\{a, e, \mathrm{h}\}$,$\{a, f, j\}$, $\{b, g, h\}$, $\{b, e, h\}$,$\{b, f, h\}$,$\{a, f, h\}$ }
- $Ans(V_2)$ = { $\{a, f, h\}$,$\{b, e, h\}$, $\{b, e, j\}$, $\{b, e, k\}$, $\{c, f, j\}$, $\{c, f, h\}$, $\{c, f, k\}$, $\{a, e, h\}$, $\{a, f, j\}$, $\{c, e, k\}$, $\{c, e, h\}$, $\{c, e, j\}$, $\{a, e, j\}$, $\{b, f, k\}$, $\{b, f, j\}$, $\{b, f, h\}$ }
- $Ans(V_3)$ = { $\{b, g, h\}$, $\{b, e, h\}$, $\{b, f, h\}$, $\{a, f, j\}$, $\{b, f, j\}$, $\{c, f, j\}$, $\{a, g, h\}$, $\{a, e, h\}$, $\{a, f, h\}$, $\{c, e, j\}$, $\{a, e, j\}$, $\{b, e, j\}$ }

Let's assume now that the preference order of candidates of each voter is obtained by use of the cardinality criterion, presented in the previous section. It is worth to note that in this case, to answer sets not satisfying the body of a rule 1 degree of satisfaction is given. Below we present rankings of answer sets, ipso facto the candidates of every voter:

- V_1: $\{a, e, j\} >_c \{b, f, j\} \sim_c \{b, g, j\} \sim_c \{b, e, j\} \sim_c \{a, e, \mathrm{h}\} \sim_c \{a, f, j\} >_c \{b, g, h\} \sim_c \{b, e, h\} \sim_c \{b, f, h\} \sim_c \{a, f, h\}$
- V_2: $\{a, f, h\} \sim_c \{b, e, h\} \sim_c \{b, e, j\} \sim_c \{b, e, k\} >_c \{c, f, j\} \sim_c \{c, f, h\} \sim_c \{c, f, k\} >_c \{a, e, h\} \sim_c \{a, f, j\} >_c \{c, e, k\} \sim_c \{c, e, h\} \sim_c \{c, e, j\} >_c \{a, e, j\} \sim_c \{b, f, k\} \sim_c \{b, f, j\} \sim_c \{b, f, h\}$
- V_3: $\{b, g, h\} \sim_c \{b, e, h\} \sim_c \{b, f, h\} >_c \{a, f, j\} \sim_c \{b, f, j\} \sim_c \{c, f, j\} \sim_c \{a, g, h\} \sim_c \{a, e, h\} \sim_c \{a, f, h\} >_c \{c, e, j\} \sim_c \{a, e, j\} \sim_c \{b, e, j\}$

Note that since preferences of voters are partial, the ordering of candidates of P is not total, i.e. only candidates considers by particular voters as "interesting" are or-

dered. The interesting candidates are candidates listed in some preference rule of a voter.

Next step is to represent the order of answer sets of each voter by an extended ordered disjunctive rule, containing sets of literals as its elements. Note that any answer set of a program representing s vote problem must have the same cardinality, due to the cardinality constraints rules of descriptive program, and this cardinality is equal to the cardinality of a set of attributes A.

Consider our example, the vote problem is represented by a new $LPOD^e$ $P_v = \{\alpha_1, \alpha_2, \alpha_3\}$ containing the following rules:

α_1 $(a, e, j) \times (b, f, j) \vee (b, g, j) \vee (b, e, j) \times (a, e, h) \vee (a, f, j) \times (b, g, h) \vee (b, e, h) \vee (b, f, h) \vee (a, f, h) \leftarrow$ not *rest*_1.

α_2 $(a, f, h) \vee (b, e, h) \vee (b, e, j) \vee (b, e, k) \times (c, f, j) \vee (c, f, h) \vee (c, f, k) \times (a, e, h) \vee (a, f, j) \times (c, e, k) \vee (c, e, h) \vee (c, e, j) \times (a, e, j) \vee (b, f, k) \vee (b, f, j) \vee (b, f, h) \leftarrow$ not *rest*_2.

α_3 $(b, g, h) \vee (b, e, h) \vee (b, f, h) \times (a, f, j) \vee (b, f, j) \vee (c, f, j) \vee (a, g, h) \vee (a, e, h) \vee (a, f, h) \times (c, e, j) \vee (a, e, j) \vee (b, e, j) \leftarrow$ not *rest*_3.

r_1 $1\{a, b, c\}1 \leftarrow$
r_2 $1\{e, f, g\}1 \leftarrow$
r_3 $1\{h, j, k\}1 \leftarrow$

$r\_1 \leftarrow$ not $head(\alpha_1)$.
$r\_2 \leftarrow$ not $head(\alpha_2)$.
$r\_3 \leftarrow$ not $head(\alpha_3)$.

The two rules at the bottom express constraints on the number of candidates to be elected and a domain of values of attributes, respectively. The above program with extended syntax is then translated to a standard *LPOD*. Our semantics of the ordered disjunction assumes that answer sets not containing any atoms of $head(r)$, satisfies r to the degree ir. Since we want to consider only those candidates, which are listed in some preference rankings, a candidate not present in such ranking is less preferred than any listed candidate, independently on its position in this ranking. Therefore, we use the modified semantics, presented on the previous secitions.

Let's consider our example again. We present three first positions of each vote rule:

- Lex-plurality voting: $\{b, e, h\} >_p \{b, e, j\} >_p \{a, f, h\}$
- Condorcet voting: $\{b, e, h\} >_{c\text{-}p} \{b, e, j\} >_{c\text{-}p} \{a, f, h\} \sim_{c\text{-}p} \{a, e, h\} \sim_{c\text{-}p} \{a, f, j\} \sim_{c\text{-}p} \{b, f, j\}$
- Lex-veto voting: $\{b, e, j\} >_p \{a, e, h\} >_p \{a, f, j\} >_p \{b, e, h\}$
- Borda voting ($ir = 10$): $\{b, e, h\} \sim_{c\text{-}p} \{b, e, j\} >_{c\text{-}p} \{a, f, h\} >_{c\text{-}p} \{a, e, h\}$

The above result shows that the candidate $\{b, e, h\}$ is elected by most of the vote rules, while being the same preferred on the Borda procedure. The candidates ranked in next positions differ, however. As long as the candidate $\{b, e, j\}$ is ranked at the second position by the lex-plurality and Condorcet rules, in case of the lex-veto method, this candidate takes a further position. Interesting is that the Borda rule,

known for its strong decisiveness, results in multiple optimal candidates: {*b, e, h*} and {*b, e, j*}.

Conclusions

In this paper we have proposed logic programming approach to combinatorial voting. We have shown how logic programming with ordered disjunction framework may be used to represent and solve vote problems. Various vote rules in the context of *LPOD* approach have been defined defined, ass well as some extensions enabling more decisiveness have been introduced. Finally, we have showed that answer set programming with preferences may be promising approach to voting problems from the following reasons: it is suitable for solving combinatorial, search and constraint satisfaction problems, as well as it is an expressive preference representation language, being able to express defeasible, conditional, partial preferences as well as meta-preference information.

References

1. Boutilier, C. et al.: CP-nets: A Tool for Representing and Reasoning with Conditional *Ceteris Paribus* Preference Statements. Journal of Artificial Intelligence Research 21 (2004) 135–191
2. Brewka, G.: Logic Programs with Ordered Disjunction. Proc. AAAI, Canada (2002) 100-105
3. Brewka, G.: Implementing Ordered Disjunction Using Answer Set Solvers for Normal Programs. Proc. JELIA'02, Italy (2002) 444-455
4. Brewka, G.: Answer Sets and Qualitative Decision Making, Synthesis, to appear 2004
5. Dummett, M.: Voting Procedures. Oxford, UK (l984). Oxford University Press
6. Grabos, R.: Qualitative Model of Decision Making. Proc. AIMSA'04, Varna (2004) 480-489
7. Kelly, J. S.: Social choice theory. Berlin.(1987) Springer
8. La Mura, P. and Y. Shoham: Expected utility networks. Proc. UAI'99, Sweden (1999) 366-373
9. Lang, J.: From preference representation to combinatorial vote, Proc. KR'02, France 277-288
10. Lang, J.: Logical Preference Representation and Combinatorial Vote. Ann. Math. Artif. Intell. 42(1-3): 37-71 (2004)
11. Lifschitz, V.: Answer set programming and plan generation. AI 138 (2002) 39-54
12. Moulin, H.: 1988, Axioms of Cooperative Decision Making. Cambridge University Press.
13. Osorio, M. et.al.: Generalized Ordered Disjunction and its Applications, not published