Resource Allocation with Answer-Set Programming

João Leite
Universidade Nova de Lisboa
Portugal
jleite@di.fct.unl.pt

José Alferes
Universidade Nova de Lisboa
Portugal
jja@di.fct.unl.pt

Belopeta Mito
Universidade Nova de Lisboa
Portugal
titoeziv@hotmail.com

ABSTRACT

Multi-agent resource allocation is a growing area of research at the frontier between Economics and Computer Science. Despite the extensive theoretical work and raising number of practical applications, many fundamental problems in multi-agent resource allocation still require adequate attention to its computational aspects. This paper addresses computational aspects of multi-agent resource allocation through the use of a fully declarative and highly expressive logic programming paradigm – Answer Set Programming – to provide uniform, general and flexible solutions to many multi-agent resource allocation problems in a compact and declarative manner.

Categories and Subject Descriptors

I.2.4 [**Computing Methodologies**]: Artificial Intelligence – Knowledge Representation Formalisms and Methods

General Terms

Algorithms, Economics.

Keywords

Resource Allocation, Answer-Set Programming, Preference Representation.

1. INTRODUCTION

In this paper we use Answer-Set Programming (ASP) to provide modular, declarative, sound and complete solutions to several different multi-agent resource allocation problems.

Resource allocation is the problem of appropriately distributing resources among entities which have preferences over alternative allocations. Classically studied in the context of Socio-economic Sciences [1, 15], this problem has made its way into the area of Computer Science [12, 16]. Whereas in Socio-economic Sciences the focus is on the qualitative aspect of resource allocation (what is a suitable distribution of resources), in Computer Science the computational aspect (how to find a suitable distribution of resources) is the central point.

The recent and active area of research known as Multi-Agent Resource Allocation (MARA) [5] aims at using the Multi-Agent System (MAS) paradigm as the unifying framework in which to tackle both the qualitative and computational aspects of resource allocation, borrowing from Socio-economic Sciences and Computer Science in general, and Artificial Intelligence in particular.

Cite as: Resource Allocation with Answer-Set Programming, João Leite, José Alferes, Belopeta Mito, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 649–656
Copyright © 2009, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org), All rights reserved.

When investigating MARA problems, there are three issues to deal with: 1) find ways to represent agent's preferences; 2) describe what suitable allocations are, taking into account both the agent's individual preferences and the welfare of the society (multi-agent system); 3) define appropriate procedures to find such allocations.

There has been extensive research conducted on all these issues, with both theoretical and practical results (c.f. [5] and references therein). Different forms of preference representation have been studied, such as bundle form, k-additive, logic-based, prioritized goals, among others. Several ways to measure the social welfare of a particular allocation have been proposed such as utilitarian, egalitarian, elitist, Nash-product and k-rank dictator. Furthermore, we also witnessed investigations regarding the quality criteria necessary to fulfill the requirements of different MARA applications, leading to notions such as welfare optimality, pareto-optimality and envy-freeness. Finally, a number of protocols have been developed for both centralized and distributed allocation of resources.

At the heart of these investigations rests the omnipresent dialectic between expressivity and complexity.

Despite the extensive theoretical work done on MARA which contributed to the better understanding of resource allocation in general, and despite the increasing number of applications requiring MARA, its computational aspects still lack the appropriate attention. With the exception of combinatorial auctions, most of the problems still lack practical solutions. The existing implemented algorithms are context dependent, thus very inflexible and non adaptable to changes in preference representation, evaluation criteria, etc. The main reason underlying this implementation shortage is that for most preference representations and quality criteria of practical interest, the complexity of finding suitable allocations is at least in the NP-complete complexity class, often raising to Σ_2^P.

Thus, no matter which allocation protocol is used, some of the agents will need to be equipped with algorithms powerful enough for solving this kind of problems and, ideally, flexible enough to accommodate additional constraints, different quality criteria, and different preference representations.

In this paper we tackle this problem with logic programming under the answer-set semantics [11] as implemented by DLV [14].

Logic Programs under the answer-set semantics, or answer-set programs, are widely recognized as a valuable tool for knowledge representation and reasoning. On the one hand, they are fully declarative in the sense that the program specification resembles the problem specification, the semantics is very intuitive, and there is extensive theoretical work that facilitates proving different properties of answer-set programs. On the other hand, finding an answer set of a logic program without disjunction (or normal logic program) is an NP-Complete problem, allowing for compact representations of all NP and coNP problems, while disjunctive logic programs under answer sets semantics capture the complexity class Σ_2^P [8]. Finally, there are now efficient answer set solvers such as DLV.

The complexity class of normal and disjunctive logic programs makes them suitable to tackle MARA problems. Additionally, their knowledge representation features allow for clean and modular representation of MARA related concepts, leading to modular implementations where different preference representations, solution descriptions and constraints can easily be replaced, combined, extended, or simply modified. Furthermore, efficiency of DLV makes the implementation usable in practice, within the limits imposed by the complexity of the problem at hand.

In this paper we overview some of the main concepts found in MARA problems and, for each of them, present an ASP encoding which serves as a modular building block to solve several resource allocation problems. In particular, we will cover: a) preference representation in bundle form, k-additive, logic-based, weighted goals and prioritized goals; b) utilitarian, egalitarian, elitist and Nash-product social welfare; c) allocation orderings such as best-out, descrimin and leximin. Different combinations of these concepts will be used to provide sound and complete solutions to a total of 66 different instances of MARA problems, ranging from welfare optimisation and improvement to pareto optimality and envy-freeness.

The paper is structured as follows: in Sect. 2, we overview the syntax and semantics of Answer-Set Programming and its DLV implementation; in Sect. 3 we present the resource allocation problem and its components, following [5], together with the corresponding ASP encodings; in Sect. 4 we use the ASP encodings to present sound and complete solutions to 66 MARA problems; in Sect. 5 we discuss simplifications, enhancements, future work and conclude.

2. ANSWER-SET PROGRAMMING & DLV

In this section we provide a brief overview of the syntax and semantics of disjunctive logic programming as implemented in DLV, following [14] where the reader can find further details.

2.1 Syntax

Strings starting with uppercase letters denote variables, while those starting with lower case letters denote constants. DLV also supports positive integer constants. A *term* is either a variable or a constant. An *atom* is an expression $\mathtt{p(t_1,\ldots,t_n)}$, where p is a predicate of arity n and $\mathtt{t_1,\ldots,t_n}$ are terms. A *classical literal* is either an atom p, or a negated atom $\neg$p. A *negation as failure literal* is of the form l or not l, where l is a literal. Unless stated otherwise, by literal we mean a classical literal.

Given a classical literal l, its *complementary literal* $\neg$l is defined as $\neg$p if l=p and p if l=$\neg$p. A set L of literals is said to be consistent if, for every literal l$\in L$, its complementary literal is not contained in L. DLV also provides built-in predicates such as the comparative predicates like equality, less-than, and greater-than (=, <, >) and arithmetic predicates like addition or multiplication. A *disjunctive rule* (rule, for short) r is a formula

$\mathtt{a_1;\ldots;a_n\text{:-}b_1,\ldots,b_k,not\ b_{k+1},\ldots,not\ b_m.}$

where $\mathtt{a_1,\ldots,a_n,b_1,\ldots,b_m}$ are classical literals, $n \geq 0$ and $m \geq k \geq 0$. The disjunction $\mathtt{a_1;\ldots;a_n}$ is the head of r, while the conjunction $\mathtt{b_1,\ldots,b_k,not\ b_{k+1},\ldots,not\ b_m}$ is the body of r. We will also define $H(r)$ =$\{\mathtt{a_1,\ldots,a_n}\}$ and $B(r) = B^+(r) \cup B^-(r)$ where $B^+(r)$ =$\{\mathtt{b_1,\ldots,b_k}\}$ and $B^-(r)$ =$\{\mathtt{b_{k+1},\ldots,b_m}\}$. A rule without head literals (i.e. $n = 0$) is usually referred to as an *integrity constraint*. A rule having precisely one head literal (i.e. $n = 1$) is called a *normal rule*. If the body is empty (i.e. $k = m = 0$), it is called a *fact*, and we usually omit the ":-" sign. We use the notation $\mathtt{p(t_1;\ldots;t_i)}$ or $\mathtt{p(t_1..t_i)}$ to denote the set of facts $\mathtt{p(t_1).,\ldots,p(t_i).}$

A *disjunctive logic program* Π is a finite set of rules. A program Π containing only normal rules is dubbed *normal logic program*.

The language of DLV also contains *weak constraints*, which are expressions of the form

$\mathtt{:\sim b_1,\ldots,b_k,not\ b_{k+1},\ldots,not\ b_m.\quad [w:l]}$

where for $m \geq k \geq 0$, $\mathtt{b_1,\ldots,b_m}$ are classical literals, while w (the weight) and l (the level, or layer) are positive integer constants or variables. The sets $B(r)$, $B^+(r)$ and $B^-(r)$ are also defined for weak constraints, as before.

A *(DLV) program* Π is a disjunctive logic program possibly containing weak constraints. Let $WC(\Pi)$ denote the set of weak constraints in Π, and $Rules(\Pi)$ denote the set of rules in Π. A rule is safe if each of its variables also appears in at least one positive literal, in its body, which is not a comparative built-in atom. We only consider programs with safe rules. A term (an atom, a rule, a program, etc.) is called ground if no variable appears in it. A finite ground program is also called a propositional program.

2.2 Semantics

The semantics of DLV programs is an extension of the answer-set semantics originally defined in [11], to deal with weak constraints. For any program Π, let U_Π (the *Herbrand Universe*) be the set of all constants appearing in Π and B_Π (*Herbrand Base*) the set of all ground (classical) literals constructible from the predicate symbols appearing in Π and the constants of U_Π. For any rule r (resp. weak constraint w) $Ground(r)$ denotes the set of rules (resp. weak constraints) obtained by applying all possible substitutions s from the variables in r (resp. w) to elements of U_Π. For any program Π, $Ground(\Pi) = \bigcup_{r \in Rules(\Pi) \cup WC(\Pi)} Ground(r)$. Note that for propositional programs, $\Pi = Ground(\Pi)$ holds. For every program Π, we define its *answer sets* using its ground instantiation $Ground(\Pi)$ in three steps: First we define the answer sets of positive disjunctive datalog programs, then we give a reduction of disjunctive datalog programs containing negation as failure to positive ones and use it to define answer sets of arbitrary disjunctive datalog programs, possibly containing negation as failure. Finally, we specify the way how weak constraints affect the semantics, defining the semantics of general DLV programs.

An interpretation I is a set of ground classical literals, i.e. $I \subseteq B_\Pi$ wrt. a program Π. A consistent interpretation I is called closed under Π (where Π is a positive disjunctive datalog program), if, for every $r \in Ground(\Pi)$, $H(r) \cap I \neq \emptyset$ whenever $B(r) \subseteq I$. An interpretation $I \subseteq B_\Pi$ is an answer set for a positive disjunctive datalog program Π, if it is minimal (under set inclusion) among all (consistent) interpretations that are closed under Π.

The reduct or Gelfond-Lifschitz transform of a ground program Π wrt. a set $I \subseteq B_\Pi$ is the positive ground program Π^I obtained from Π by: 1) deleting all rules $r \in \Pi$ for which $B^-(r) \cap I \neq \emptyset$ holds; 2) deleting the negative body from the remaining rules. An *answer set* of a program Π is a set $I \subseteq B_\Pi$ such that I is an answer set of $Ground(\Pi)^I$.

Given a ground program Π with weak constraints $WC(\Pi)$, we seek the answer sets of $Rules(\Pi)$ which minimize the sum of weights of the violated (unsatisfied) weak constraints in the highest priority level, and among them those which minimize the sum of weights of the violated weak constraints in the next lower level, etc. Formally, this is expressed by an objective function $H^\Pi(A)$ for Π and an answer set A as follows, using an auxiliary function f_Π which maps leveled weights to weights without levels:

$$\begin{aligned}
f_\Pi(1) &= 1\\
f_\Pi(n) &= f_\Pi(n-1) \cdot |WC(\Pi)| \cdot w^\Pi_{max} + 1, n > 1\\
H_\Pi(A) &= \textstyle\sum_{i=1}^{l^\Pi_{max}} \left(f_\Pi(i) \cdot \sum_{w \in N^\Pi_i(A)} weight(w) \right),
\end{aligned}$$

where w^{Π}_{max} and l^{Π}_{max} denote the maximum weight and maximum level over the weak constraints in Π, respectively; $N^{\Pi}_i(A)$ denotes the set of the weak constraints in level i that are violated by A, and $weight(w)$ denotes the weight of the weak constraint w. Intuitively, the function f_Π handles priority levels. It guarantees that the violation of a single constraint of priority level i is more "expensive" than the violation of all weak constraints of the lower levels (i.e., all levels $< i$). For a DLV program Π (possibly with weak constraints), a set A is an *(optimal) answer set* of Π if and only if: 1) A is an answer set of $Rules(\Pi)$ and 2) $H_P(A)$ is minimal over all the answer sets of $Rules(\Pi)$. Let $AS(\Pi)$ denote the set of *(optimal) answer sets* of Π.

DLV also allows for aggregate functions whose syntax is defined as follows. A (DLV) symbolic set is a pair $\{Vars : Conj\}$ where $Vars$ is a list of variables and $Conj$ is a conjunction of standard literals. Intuitively, a symbolic set `{X:a(X,Y),p(Y)}` stands for the set of `X`-values making `a(X,Y),p(Y)` true. An aggregate function is of the form $f(S)$ where S is a symbolic set, and f is a function name among `#count`, `#min`, `#max`, `#sum`, `#times`. An *aggregate atom* is $Lg \prec_1 f(S) \prec_2 Rg$ where $f(S)$ is an aggregate function, $\prec_1, \prec_2 \in \{=, <, \leq, >, \geq\}$and Lg and Rg are terms. One of Lg and Rg can be omitted. Rules can also include aggregate atoms in their bodies. As an example, consider a predicate `person(Name,Age,Salary)`. The aggregate atom `19<#min{A:person(N,A,S)}`$\leq$`30` then evaluates to true iff the age of the youngest person is in the range [20..30]. The aggregate function `#sum{S,N:person(N,A,S)}=T` returns the sum of the salaries of all the employees[1]. The formal semantics of aggregates can be found in [7].

3. RESOURCE ALLOCATION

In this Section we take a closer look at resources, agents, allocations and preference representations, presenting, as we proceed, ASP encodings of these concepts that serve as the building blocks of the ASP programs used to solve the different MARA problems.

Before we proceed with a detailed account of the relevant concepts related to MARA we set forth a more formal definition both of the input and the solution space. A *Multi-Agent Resource Allocation Setting* (*MARA Setting*) is a triple $\langle A, R, U\rangle$ where $A = \{a_1, a_2, ..., a_n\}$ is a set of n agents, $R = \{r_1, r_2, ..., r_m\}$ is a set of m resources and $U = \{ur_{a_1}, ur_{a_2}, ..., ur_{a_n}\}$ is a set of preference representations of the agents. An allocation is a mapping $M : A \rightarrow 2^R$ from agents to a subset of resources R such that $M(a_i) \cap M(a_j) = \emptyset$ for any two agents $a_i \neq a_j$. If $\Theta = \langle A, R, U\rangle$ is a MARA Setting, then, $Ag(\Theta) = A$, $Res(\Theta) = R$, $Pref(\Theta) = U$ and M^{Θ} denotes the set of all possible allocations.

3.1 Resources, Agents and Allocations

Resources play a central role in any resource allocation problem. We can categorise resources according to their nature and the way they are treated during allocation. They are usually categorised as *continuous* or *discrete*, *indivisible* or *divisible*, *sharable* or *non sharable* and *static* or *non static* (c.f. [5] for explanation). In this paper we deal with the allocation of *discrete*, *indivisible*, *non sharable* and *static* resources, each represented by a fact in ASP. Likewise, agents will be represented by simple facts in ASP.

Let Θ be a MARA Setting. Then Π^{Θ}_{AR} contains:

`ag(a).` $\forall a \in Ag(\Theta)$.
`res(r).` $\forall r \in Res(\Theta)$.

[1]The sum is formed over the first variable. The second, N, ensures that one value per person is summed. Otherwise, two people with the same salary would only be summed once.

Before we proceed, we introduce the ASP encoding of an allocation. Let $M_i : A \rightarrow 2^R$ be an allocation. Then Π_{M_i} contains:

`has(a,r,i).` $\forall a, r : r \in M_i(A)$.
`al(i).`

3.2 Preference Representation

Agents' preferences represent the relative or the absolute agent's satisfaction with a particular allocation, and are necessary for defining the quality aspects of a solution to a resource allocation problem. A preference structure represents the preferences of an agent over a possible set of alternatives. We will consider two preference structures: *cardinal* and *ordinal*.

The *cardinal preference structure* consists of a utility function $u : 2^R \rightarrow Val$ where 2^R is the set of alternative bundles (sets of resources), and Val is a set of numeric values or a totally ordered scale of qualitative values (e.g. very bad, bad, average, good....). The former is called quantitative while the latter alternatives qualitative preference structure. In this paper we do not consider the qualitative case as it can be reduced to the quantitative one.

The *ordinal preference structure* consist of binary relations between alternatives $x \preceq y$ denoting that the bundle x is at least as preferred as the bundle y. Strict preference is denoted by $x \prec y$.

Other kinds not considered here include the binary preference structure which partitions the set of alternatives into good ones and bad ones, and fuzzy preference structure which is a fuzzy relation over the set of alternatives 2^R, i.e. a function $f : 2^R \times 2^R \rightarrow [0, 1]$ defining the degree to which one alternative is preferred over the other. Since such preferences have not been extensively studied and are rarely used in practice, they are not further considered here.

Preferences must be properly represented when used in practice. Since the number of alternatives is exponential wrt. the number of resources, an explicit representation (listing all alternatives with their utility values for cardinal preference, or the full relation for ordinal preferences) is exponentially large. For this reason, languages for succinct preference representation are important. In what follows, we present some common languages, first for the quantitative cardinal preference representation and then for the ordinal preference representation, together with their ASP encodings.

3.2.1 Bundle Form Preferences

The bundle form is the most basic approach for representing quantitative preferences. It simply enumerates all bundles for which the agent has a non-zero utility. Thus, a preference representation in the bundle form is a set of pairs $\langle B, u(B)\rangle$ for all bundles B such that $u(B) \neq 0$.

In ASP, we will encode quantitative preferences through the definition of a predicate `ut(`$\mathtt{A}_i$`,V,`$\mathtt{A}_j$`,X)` which represents the utility `V` agent $\mathtt{A}_i$ assigns to the set of resources (bundle) given to agent $\mathtt{A}_j$ in allocation `X`.

Let Θ be a MARA Setting with maximum utility u_{max} and preferences in the bundle form[2]. Then Π^{Θ}_{BFS} contains the rules:

```
utg(0..umax).
ut(a,u,A,X):-al(X),ag(A),
        has(A,r1,X),...,has(A,ri,X),
  not has(A,rj,X),...,not has(A,rn,X).
```

$\forall \langle B, u\rangle \in ur_a : B = \{r_1, ..., r_i\} \wedge Res(\Theta) - B = \{r_j, ..., r_n\}$ and $\forall ur_a \in Pref(\Theta)$.

The first two predicates in the body of the rule, `al(X),ag(A)`, are required to make the rules safe. Similar predicates will be used in other ASP encodings.

[2]u_{max} is the highest utility *expressed* by an agent, rather than the highest utility *expressible* by an agent, hence easy to determine.

For some problems it is useful to explicitly define a predicate `ut(A_i,0,A_j,X)` when the utility value is equal to zero: $\Pi^{\Theta}_{BF} = \Pi^{\Theta}_{BFS} \cup \Pi^{\Theta}_{Z}$ where Π^{Θ}_{Z} contains the rule:

```
has_ut(A1,A2,X):-al(X),ag(A1;A2),
                 ut(A1,V,A2,X),V!=0.
ut(A1,0,A2,X):-al(X),ag(A1;A2),
                not has_ut(A1,A2,X).
```

3.2.2 K-additive Preferences

Another approach is the k-additive form [6] which exploits regularities in a function in order to build efficiently computable succinct representations. Given $k \in \mathbb{N}$, a utility function u is k-additive if and only if for every set of resources T of size less or equal to k there exists a coefficient α_T such that $u(R) = \sum_{T \subseteq R} \alpha_T$, for every set of resources R. If a utility function is represented in terms of such coefficients then it is given in the k-additive form. Thus, a preference representation in the k-additive form is a set of pairs of the form $\langle T, \alpha_T \rangle$. Intuitively the coefficient α_T represents the additional gain of utility when having all the resources in T together i.e. it represents the synergetic value of T.

Let Θ be a MARA Setting with maximum utility u_{max} and preferences in k-additive form. Then Π^{Θ}_{KA} contains the rules:

```
utg(0..u_max).
c(a,α,i,A,X):-al(X),ag(A),
          has(A,r_1,X),...,has(A,r_n,X).
ut(A1,U,A2,X):-al(X),ag(A1;A2),utg(U),
          #sum{C,I:c(A1,C,I,A2,X)}=U.
```

$\forall t_i = \langle T, \alpha \rangle \in ur_a : T = \{r_1, ..., r_n\}$ and $\forall ur_a \in Pref(\Theta)$.

3.2.3 Logic-based Preferences

An alternative approach for compact representation of preferences is the explicit use of logic languages [13]. In a *logic-based* preference representation each resource r_i corresponds to a propositional formula p_{r_i} that is true if the agent owns that resource and false otherwise. Every bundle corresponds to a model. Agents express their preferences in terms of propositional formulae (goals) they want to be satisfied. The simplest such representation is to have one propositional formula representing the goal G. The agent's utility is 1 if the received bundle R satisfies the goal G ($R \models G$), and 0 otherwise ($R \not\models G$). We adopt a refinement where we consider a preference representation as a set of goals $\{G_1, \ldots, G_n\}$ and then count the number of satisfied goals in R. We assume propositional formulae to be in disjunctive normal form.

Let Θ be a MARA Setting with maximum utility u_{max} and preferences in logic-based form. Then Π^{Θ}_{LB} contains the rules:

```
utg(0..u_max).
g(a,i,A,X):-al(X),ag(A),
          has(A,r_1,X),...,has(A,r_h,X),
  not has(A,r_i,X),...,not has(A,r_j,X).
ut(A1,U,A2,X):-al(X),ag(A1;A2),utg(U),
          #count{G:g(A1,G,A2,X)}=U.
```

$\forall d_k = p_{r_1} \wedge ... \wedge p_{r_h} \wedge -p_{r_i} \wedge ... \wedge -p_{r_j} : 1 \leq k \leq m$ where $d_1 \vee ... \vee d_m = G_i, \forall G_i \in ur_a$ and $\forall ur_a \in Pref(\Theta)$.

Another refinement is to associate a weight w_i to each goal (i.e. the preference representation is $\{\langle G_1, w_1 \rangle, \ldots, \langle G_n, w_n \rangle\}$). The weight represents the importance of a goal and is used to associate penalties to the bundles that do not satisfy it. Here, for simplicity, we consider the utility of a bundle to be the sum of the weights of all the goals it satisfies. This representation is called *weighted goals*. As the k-additive representation, this representation can express the synergetic value of having two resources together.

Let Θ be a MARA Setting with maximum utility u_{max} and preferences as weighted goals. Then Π^{Θ}_{WG} contains the rules:

```
utg(0..u_max).
g(a,i,w,A,X):-al(X),ag(A),
          has(A,r_1,X),...,has(A,r_h,X),
  not has(A,r_i,X),...,not has(A,r_j,X).
ut(A1,U,A2,X):-al(X),ag(A1;A2),utg(U),
          #sum{W,G:g(A1,G,W,A2,X)}=U.
```

$\forall d_k = p_{r_1} \wedge ... \wedge p_{r_h} \wedge -p_{r_i} \wedge ... \wedge -p_{r_j} : 1 \leq k \leq m$ where $d_1 \vee ... \vee d_m = G_i, \forall \langle G_i, w \rangle \in ur_a$ and $\forall ur_a \in Pref(\Theta)$.

3.2.4 Prioritized Goals

We now turn our attention to *ordinal preference* representations and consider *prioritized goals*.

When representing ordinal preferences, logic based languages play a central role. In essence, a very similar representation to the one of weighted goals is used for representing ordinal preferences. The only difference is that instead of weights this representation uses priority relation on goals and thus it is called prioritized goals. A goal base is a set of goals with an associated function $\langle \{G_1, \ldots, G_n\}, r \rangle$ where if $r(G_i) = j$ then j is the rank of the goal G_i. By convention a smaller rank means a higher priority. In this case, we define the predicate `sat(A1,G,A2,X)` indicating that goal `G` of agent `A1` is satisfied by the bundle assigned to agent `A2` in allocation `X`. These predicates will subsequently be used to define preference relations between bundles and allocations. Let Θ be a MARA Setting and preferences in logic-based form as prioritized goals with maximum rank r_{max}. Then Π^{Θ}_{PG} contains the rules:

```
r(1..r_max+1).
g(a,g,r).
sat(a,g,A,X):-al(X),ag(A),
        has(A,r_1,X),...,has(A,r_h,X),
not has(A,r_i,X),...,not has(A,r_j,X).
```

$\forall d_k = p_{r_1} \wedge ... \wedge p_{r_h} \wedge -p_{r_i} \wedge ... \wedge -p_{r_j} : 1 \leq k \leq m$ where $d_1 \vee ... \vee d_m = G_i, \forall G_i \in GB, \forall \langle GB, r \rangle \in ur_a, r = r(G_i), \forall ur_a \in Pref(\Theta)$ and $g = \Phi(G_i)$ where Φ is a function that returns a unique identifier for each goal.

For each agent, a preference relation can be defined over the allocations (resp. the bundles allocated to different agents in extended preferences) with respect to the ordinal preferences. There are three common orderings: *best out*, *descrimin* and *leximin* [4]. For each, the ASP encoding defines a predicate `pref_al(A,X1,X2)` meaning that an agent `A` prefers the bundle received in allocation `X1` at least as much as the one received in allocation `X2` (resp. `pref_b(A,A1,A2,X)` meaning that in allocation `X`, agent `A` prefers the bundle allocated to `A1` at least as much as the one allocated to `A2`). Some auxiliary predicates, common to the definition of both `pref_al` and `pref_b` will also be defined.

Best-out ordering: An allocation M_2 is best out preferred over an allocation M_1 if the minimal rank of goal(s) not satisfied in M_1 is less or equal to the minimal rank of goal(s) not satisfied in M_2. Formally, $M_1 \preceq^{bo}_{GB} M_2$ iff $min\{r(G_i) : M_1 \not\models G_i\} \leq min\{r(G_i) : M_2 \not\models G_i\}$. The encoding in ASP is accomplished by the programs Π_{BOc} containing the common predicates, Π_{BOal} and Π_{BOb} defining the `pref_al` and `pref_b` predicates respectively.

- Π_{BOc} contains the rules:

```
nsat(A1,A2,R,X):-al(X),ag(A1;A2),g(A1,G,R),
                          not sat(A1,G,A2,X).
hasmin(A,A1,X):-nsat(A,A1,R,X).
```

- $\Pi_{BOal} = \Pi_{BOc}$ together with the rules:

```
pref_al(A,X1,X2):-al(X1;X2),X1!=X2,ag(A),
      M2<=M1,#min{R1:nsat(A,A,R1,X1)}=M1,
            #min{R2:nsat(A,A,R2,X2)}=M2.
```

```
pref_al(A,X1,X2):-al(X1;X2),X1!=X2,ag(A),
                         not hasmin(A,A,X1).
```

• $\Pi_{BOb} = \Pi_{BOc}$ together with the rules:

```
pref_b(A,A1,A2,X):-al(X),ag(A,A1;A2),A1!=A2,
         M2<=M1,#min{R1:nsat(A,A1,R1,X)}=M1,
                 #min{R2:nsat(A,A2,R2,X)}=M2.
pref_b(A,A1,A2,X):-al(X),ag(A,A1;A2),A1!=A2,
                         not hasmin(A,A1,X).
```

Descrimin ordering: An allocation M_2 is descrimin preferred over an allocation M_1 if the minimal rank of goal(s) not satisfied in M_1 but satisfied in M_2 is lower than the minimal rank of goal(s) not satisfied in M_2 but satisfied in M_1, or if the sets of goals satisfied in each allocation are equal. Formally, let $d(M_1, M_2) = min\{r(G_i) : M_1 \not\models G_i \wedge M_2 \models G_i\}$. Then $M_1 \preceq^{do}_{GB} M_2$ iff $d(M_1, M_2) < d(M_2, M_1)$ or $\{G_i : M_1 \models G_i\} = \{G_i : M_2 \models G_i\}$. The encodings in ASP are:

• Π_{DOal} contains the rules:

```
alD(A,X1,X2,D):-al(X1;X2),X1!=X2,ag(A),
g(A,G,D),not sat(A,G,A,X1),sat(A,G,A,X2).
hasD(A,X1,X2):-al(X1;X2),X1!=X2,ag(A),
                             alD(A,X1,X2,D).
pref_al(A,X1,X2):-al(X1;X2),X1!=X2,ag(A),
        G2<G1,#min{D1:alD(A,X1,X2,D1)}=G1,
              #min{D2:alD(A,X2,X1,D2)}=G2.
pref_al(A,X1,X2):-al(X1;X2),X1!=X2,ag(A),
                           not hasD(A,X1,X2).
```

• Π_{DOb} contains the rules:

```
bD(A,A1,A2,X,D):-al(X),ag(A;A1;A2),g(A,G,D),
           not sat(A,G,A1,X),sat(A,G,A2,X).
hasbD(A,A1,A2,X):-al(X),ag(A;A1;A2),
                              bD(A,A1,A2,X,D).
pref_b(A,A1,A2,X):-al(X),ag(A,A1;A2),G2<G1,
              #min{D1:bD(A,A1,A2,X,D1)}=G1,
              #min{D2:bD(A,A2,A1,X,D2)}=G2.
pref_b(A,A1,A2,X):-al(X),ag(A,A1;A2),
                          not hasbD(A,A1,A2,X).
```

Leximin ordering: An allocation M_2 is leximin preferred over an allocation M_1 if it satisfies more goals of certain rank i, and an equal number of goals for all ranks which are smaller than i. Formally, let $d_k(M) = |\{G_i : M \models G_i \wedge r(G_i) = k\}|$. Then, $M_1 \prec^{lo}_{GB} M_2$ iff $\exists k : d_k(M_1) < d_k(M_2)$ and $\forall j < k : d_j(M_1) = d_j(M_2)$. $M_1 \preceq^{lo}_{GB} M_2$ iff $M_1 \prec^{lo}_{GB} M_2$ or $\forall j : d_j(M_1) = d_j(M_2)$. The encodings in ASP are:

• Π_{LOc} contains the rules:

```
dk(A,A2,R,DK,X):-al(X),ag(A;A2),r(R),
#count{G:sat(A,G,A2,X),goal(A,G,R)}=DK.
```

• $\Pi_{LOal} = \Pi_{LOc}$ together with the rules:

```
difalP(A,X1,X2,R):-al(X1;X2),X1!=X2,ag(A),
          r(R;R2),R>R2,dk(A,A,R2,DK1,X1),
                dk(A,A,R2,DK2,X2),DK1!=DK2.
pref_al(A,X1,X2):-al(X1;X2),X1!=X2,ag(A),
    r(R),dk(A,A,R,DK1,X1),dk(A,A,R,DK2,X2),
          DK1>DK2,not difalP(A,X1,X2,R).
pref_al(A,X1,X2):-al(X1;X2),X1!=X2,ag(A),
              not difalP(A,X1,X2,rmax+1).
```

• $\Pi_{LOb} = \Pi_{LOc}$ together with the rules:

```
difbR(A,A1,A2,X,R):-ag(A;A1;A2),A1!=A2,al(X),
          r(R;R2),R>R2,dk(A,A2,R2,DK2,X),
              dk(A,A1,R2,DK1,X),DK1!=DK2.
pref_b(A,A1,A2,X):-al(X),ag(A;A1;A2),A1!=A2,
     r(R),dk(A,A1,R,DK1,X),dk(A,A2,R,DK2,X),
          DK1>DK2,not difbR(A,A1,A2,X,R).
pref_b(A,A1,A2,X):-al(X),ag(A;A1;A2),A1!=A2,
            not difbR(A,A1,A2,X,rmax+1).
```

3.3 Qualitative Issues

In general, the quality of a solution to a resource allocation problem is defined by a metric that depends, in one way or another, on the preferences of the agent. The aggregation of the individual preferences in such metric is often modeled using the notion of social welfare as studied in Welfare Economics and Social Choice Theory. Depending on the system's goals, different social welfare functions can be used for measuring the quality of an allocation [1, 15].

The most fundamental quality criterion for a solution is *Pareto Optimality*. Given a MARA Setting Θ, an allocation M is pareto-dominated by another allocation Q if and only if: 1) $\forall a_i \in Ag(\Theta) : M(a_i) \preceq_{a_i} N(a_i)$ and 2) $\exists a_i \in Ag(\Theta) : M(a_i) \prec_{a_i} N(a_i)$. In the case of utility functions, we simply replace $M(a_i) \preceq_{a_i} N(a_i)$ and $M(a_i) \prec_{a_i} N(a_i)$ with $u_i(M(a_i)) \leq u_i(N(a_i))$ and $u_i(M(a_i)) < u_i(N(a_i))$. An allocation is *pareto-optimal* if and only if it is not pareto-dominated by any other allocation. The ASP encodings of pareto dominance define a predicate `pd(X1,X2)` indicating that allocation `X2` is dominated by allocation `X1`. They are, for the cardinal and ordinal preferences respectively:

• Π_{POc} contains the rules:

```
pd(X1,X2):-al(X1;X2),dm(X1,X2),not dm(X2,X1).
dm(X1,X2):-al(X1;X2),ag(A),ut(A,U1,A,X1),
                         ut(A,U2,A,X2),U1>U2.
```

• Π_{POo} contains the rules:

```
pd(X1,X2):-al(X1;X2),dm(X1,X2),not dm(X2,X1).
dm(X1,X2):-al(X1;X2),ag(A),pref_al(A,X1,X2),
                         not pref_al(A,X2,X1).
```

Envy freeness is also a purely ordinal quality criterion. An allocation is envy-free when every agent is at least as satisfied with the bundle allocated to it as with the bundles allocated to other agents. Given a MARA Setting Θ, an allocation M is envy-free if and only if $\forall a_i, a_j \in Ag(\Theta) : M(a_j) \preceq_{a_i} M(a_i)$ for the case of ordinal preferences, and $\forall a_i, a_j \in Ag(\Theta) : u_i(M(a_j)) \leq u_i(M(a_i))$. The ASP encodings of envy-freeness define a predicate `¬ef(X)` indicating that allocation `X` is *not* envy-free. They are, for the cardinal and ordinal preferences respectively:

• Π_{EFc} contains the rules:

```
¬ef(X):-al(X),ag(A1;A2),A1!=A2,
ut(A1,U1,A1,X),ut(A1,U2,A2,X),U2>U1.
```

• Π_{EFo} contains the rules:

```
¬ef(X):-al(X),ag(A1;A2),A1!=A2,
pref_b(A1,A2,A1,X),not pref_b(A1,A1,A2,X).
```

In some cases, when no envy-free allocation exists, we may be interested in minimizing some measure of the degree of envyness. Given an allocation, the degree of envyness of some agent wrt. another agent is given by the difference between its utility and the other agent's utility, in case its own is smaller. This is encoded in ASP as follows:

• Π_{EFod} contains the rules:

```
envy(A1,A2,D,X):-al(X),ag(A1;A2),A1!=A2,
ut(A1,U1,A1,X),ut(A1,U2,A2,X),U2>U1,U2-U1=D.
```

Given a MARA Setting Θ and an allocation M, $envy(M)$ denotes the sum of the degree of envyness of all pairs of agents.

3.3.1 Social Welfare

When the agents have their preferences expressed with a utility function, then every allocation M gives rise to the utility vector $\langle u_1(M), ..., u_n(M)\rangle$. A collective utility function (CUF) is a mapping from such a vector to a numeric value. Since every allocation determines a utility vector, CUF can also be considered as a function from allocations to numeric values which represent the social welfare of the corresponding allocation. Thus, given a CUF sw, an allocation M is socially preferred to allocation N if and only if $sw(N) < sw(M)$.

The most common CUF is the *utilitarian social welfare* defined as the sum of the individual utilities of the agents. Formally, given a MARA Setting Θ, $sw_{ut}(M) = \sum_{a \in Ag(\Theta)} u_a(M)$. This CUF is useful e.g. when the concern is the overall profit in e-commerce applications. This CUF is encoded in ASP as follows: let Θ be a MARA Setting with a agents and maximum utility u_{max}. Let $i = a * u_{max}$ and $np = u_{max}{}^{a}$ Then Π^{Θ}_{SWut} contains the rules:

```
w(0..i).
sw(S,X):-al(X),w(S),#sum{U,A:ut(A,U,A,X)}=S.
```

If ensuring fairness is a priority, then the appropriate CUF is *egalitarian social welfare* defined by the utility of the agent that is currently worst off. Formally, given a MARA Setting Θ, $sw_{eg}(M) = min(u_a(M) : a \in Ag(\Theta))$. This CUF is useful when there is the need to satisfy the minimum needs of a large number of customers. Considering the same setting, this CUF is encoded in ASP with Π^{Θ}_{SWeg} which contains the rules:

```
w(0..u_max).
sw(S,X):-al(X),w(S),#min{U:ut(A,U,A,X)}=S.
```

The opposite of the above CUF is the *elitist social welfare* defined by the utility of the agent that is currently best off. Formally, given a MARA Setting Θ, $sw_{el}(M) = max(u_a(M) : a \in Ag(\Theta))$. This CUF is useful when only one agent is required to achieve its goals. Considering the same setting, this CUF is encoded in ASP with Π^{Θ}_{SWel} containing the rules:

```
w(0..u_max).
sw(S,X):-al(X),w(S),#max{U:ut(A,U,A,X)}=S.
```

A compromise between the utilitarian and egalitarian social welfare in the sense that it favors overall utility and reduces inequalities between agents is provided by the *Nash product*, defined as the product of the individual utilities. Formally, given a MARA Setting Θ, $sw_{ut}(M) = \prod_{a \in Ag(\Theta)} u_a(M)$. Considering the same setting, this CUF is encoded in ASP with Π^{Θ}_{SWnp} containing the rules:

```
w(0..np).
sw(S,X):-al(X),w(S),
#times{U,A:ut(A,U,A,X)}=S.
```

4. ASP BASED SOLUTIONS OF MARA

Now that we have ASP representations of the main MARA related concepts, we can turn our attention to the use of ASP to solve the MARA problems. Solving problems using DLV implementation of ASP follows the classical *Guess/Check/optimise* programming methodology. This is done by guessing a possible allocation (or enumerating all possible valid allocations), checking its compliance with a certain criteria (or eliminating the ones which do not satisfy such criteria) and optimise among the ones that passed the previous check (further eliminating the non-optimal ones). As will be seen, this last step is not always required.

We start with a program module to enumerate the possible allocations. The preference representations do not matter at this stage since they are only used for the checking part. One possible approach to the enumeration of all allocations is to use the DLV rule:

```
has(A,R,o);¬has(A,R,o):-ag(A),res(R).
```

whose meaning is that for each agent `A` and each resource `R`, in allocation `o` either the agent has the resource, represented by the predicate `has(A,R,o)` or the agent does not have the resource, represented by the predicate `¬has(A,R,o)`. This encoding would have to be accompanied by a set of rules to eliminate allocations in which some resource is allocated to more than one agent, and those where not all resources are allocated. Instead, we opt for a different encoding in ASP which allocates each resource to exactly one agent using a technique known as *general enumeration of exactly one* [2]. Let Θ be a MARA Setting. Then:

• $\Pi^{\Theta}_{G} = \Pi^{\Theta}_{AR} \cup \Pi^{\Theta}_{Gen}$ where Π^{Θ}_{Gen} contains the rules:

```
al(o).
has(A,R,o):-ag(A),res(R),not ¬has(A,R,o).
¬has(A,R,o):-ag(A;B),res(R),has(B,R,o),A!=B.
```

Note that we are already including the rules of Π^{Θ}_{AR}, previously defined, in Π^{Θ}_{G} since they will always be present.

4.1 Welfare Optimisation Problem

The *welfare optimisation problem* is the problem of checking whether there is an allocation which exceeds a certain value in respect to a particular social welfare function. The input is a MARA Setting Θ and a value k. Then Π^{k}_{WOd} contains the rule:

```
:-sw(SW,o),SW<=k.
```

THEOREM 1 (DECISION). *Let Θ be a MARA Setting with a agents, maximum utility u_{max} and preferences in one of bundle, k-additive, logic based or weighted goals form. Let $k \leq u_{max}{}^{a}$. Let sw be one of utilitarian, egalitarian, elitist or Nash-product social welfare functions. Let Π^{Θ}_{Pref} and Π^{Θ}_{SW} be defined according to the table below. Then*[3]*:*

• for each allocation $M \in M^{\Theta} : sw(M) > k$ there is an answer set $S \in AS(\Pi^{\Theta}_{G} \cup \Pi^{\Theta}_{Pref} \cup \Pi^{\Theta}_{SW} \cup \Pi^{k}_{WOd})$ such that `has(a,r,o)`$\in S$ *iff $r \in M(a)$.*

• for each answer set $S \in AS(\Pi^{\Theta}_{G} \cup \Pi^{\Theta}_{Pref} \cup \Pi^{\Theta}_{SW} \cup \Pi^{k}_{WOd})$ there is an allocation $M \in M^{\Theta}$ such that `has(a,r,o)`$\in S$ *iff $r \in M(a)$ and $sw(M) > k$.*

Preference Form	Π^{Θ}_{Pref}
bundle	Π^{Θ}_{BF}
k-additive	Π^{Θ}_{KA}
logic based	Π^{Θ}_{LB}
weighted goals	Π^{Θ}_{WG}

Social Welfare	Π^{Θ}_{SW}
Utilitarian	Π^{Θ}_{SWut}
Egalitarian	Π^{Θ}_{SWeg}
Elitist	Π^{Θ}_{SWel}
Nash-product	Π^{Θ}_{SWnp}

The above theorem indicates that deciding if such allocation exists is equivalent to deciding if an answer set exists, and finding one (resp. all) such allocation(s) is equivalent to finding one (resp. all) answer-set(s). We now turn our attention to finding the optimal solutions. As this is a problem that falls outside the NP and coNP complexity classes, it can only be solved with weak constraints.

• Π_{WOo} contains the rule:

```
:~w(SW),not sw(SW,o).[SW:1].
```

THEOREM 2 (OPTIMISATION). *Let Θ be a MARA Setting with preferences in one of bundle, k-additive, logic based or weighted goals form. Let sw be one of utilitarian, egalitarian, elitist or Nash-product social welfare functions. Let Π^{Θ}_{Pref} and Π^{Θ}_{SW} be defined according to the table above, and Π^{Θ}_{G} and Π_{WOo} as before. Then:*

• for each allocation $M \in M^{\Theta} : \nexists M' \in M^{\Theta}, sw(M') > sw(M)$ there is an answer set $S \in AS(\Pi^{\Theta}_{G} \cup \Pi^{\Theta}_{Pref} \cup \Pi^{\Theta}_{SW} \cup \Pi_{WOo})$ such that `has(a,r,o)`$\in S$ *iff $r \in M(a)$.*

• for each answer set $S \in AS(\Pi^{\Theta}_{G} \cup \Pi^{\Theta}_{Pref} \cup \Pi^{\Theta}_{SW} \cup \Pi_{WOo})$ there is an allocation $M \in M^{\Theta}$ such that `has(a,r,o)`$\in S$ *iff $r \in M(a)$ and $\nexists M' \in M^{\Theta}, sw(M') > sw(M)$.*

4.2 Welfare Improvement Problem

The *welfare improvement problem* is the problem of checking whether a given allocation is optimal with respect to a particular social welfare function or, in other words, if there is another allocation which improves the value of such welfare function. The input is a MARA Setting Θ and an initial allocation M.

• Π_{WI} contains the rule:

```
:-sw(SW_i,i),sw(SW_o,o),SW_i>=SW_o.
```

[3]Lack of space prevents the presentation of the proofs of theorems.

THEOREM 3 (DECISION). *Let Θ be a MARA Setting with preferences in one of bundle, k-additive, logic based or weighted goals form. Let $M_i \in M^\Theta$ be an allocation. Let sw be one of utilitarian, egalitarian, elitist or Nash-product social welfare functions. Let Π^Θ_{Pref} and Π^Θ_{SW} be defined according to the table above, and Π^Θ_G, Π_{M_i} and Π_{WI} as before. Then:*

• for each allocation $M_o \in M^\Theta : sw(M_o) > sw(M_i)$ there is an answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{Pref} \cup \Pi^\Theta_{SW} \cup \Pi_{WI} \cup \Pi_{M_i})$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$.

• for each answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{Pref} \cup \Pi^\Theta_{SW} \cup \Pi_{WI} \cup \Pi_{M_i})$ there is an allocation $M_o \in M^\Theta$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$ *and* $sw(M_o) > sw(M_i)$.

4.3 Pareto Optimality Problem

The *pareto optimality problem* is the problem of checking whether a particular allocation is pareto optimal, i.e. it is not pareto dominated (there is no other allocation which increases the utilities of at least one agent without decreasing the utility of the others). The input is a MARA Setting Θ and an initial allocation M.

• Π_{PO} contains the rule:
```
:-not pd(o,i).
```

THEOREM 4 (DECISION - CARDINAL PREFERENCES). *Let Θ be a MARA Setting with preferences in one of bundle, k-additive, logic based or weighted goals form. Let $M_i \in M^\Theta$ be an allocation. Let Π^Θ_{Pref} be defined according to the table above, and Π^Θ_G, Π_{POc}, Π_{M_i} and Π_{PO} as before. Then:*

• for each allocation $M_o \in M^\Theta$ such that M_o pareto dominates M_i there is an answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{Pref} \cup \Pi_{POc} \cup \Pi_{PO} \cup \Pi_{M_i})$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$.

• for each answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{Pref} \cup \Pi_{POc} \cup \Pi_{PO} \cup \Pi_{M_i})$ there is an allocation $M_o \in M^\Theta$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$ *and M_o pareto dominates M_i.*

THEOREM 5 (DECISION - ORDINAL PREFERENCES). *Let Θ be a MARA Setting with preferences in logic-based form as prioritized goals. Let the preference relation defined over bundles be based on one of best out, descrimin and leximin orderings. Let $M_i \in M^\Theta$ be an allocation. Let Π_{Ordal} be defined according to the table below, and Π^Θ_G, Π_{POo}, Π^Θ_{PG}, Π_{M_i} and Π_{PO} as before. Then:*

• for each allocation $M_o \in M^\Theta$ such that M_o pareto dominates M_i there is an answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{PG} \cup \Pi_{POo} \cup \Pi_{Ordal} \cup \Pi_{PO} \cup \Pi_{M_i})$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$.

• for each answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{PG} \cup \Pi_{POo} \cup \Pi_{Ordal} \cup \Pi_{PO} \cup \Pi_{M_i})$ there is an allocation $M_o \in M^\Theta$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$ *and M_o pareto dominates M_i.*

Preference Ordering	Π_{Ordb}	Π_{Ordal}
best out	Π_{BOb}	Π_{BOal}
descrimin	Π_{DOb}	Π_{DOal}
leximin	Π_{LOb}	Π_{LOal}

The optimisation version of the Pareto Optimality problem i.e. finding all allocations which are not pareto dominated, is in the Σ^P_2 complexity class which can be captured by a strictly disjunctive logic program under the answer set semantics. However, the guess and check formalization of such problems in ASP is not as intuitive as in the case of NP or coNP problems since the use of default negation is restricted in the check part of the program. Essentially, representing Σ^P_2 problems in ASP is accomplished through a method known as saturation [9, 8] in which the minimal property of ASP is explicitly exploited. Lack of space prevents us from presenting such program which, besides the initial data representation, would not reuse any previously presented code. Furthermore, unlike other problems, the POO problem will require a separate formalization for every type of preference representation.

4.4 Envy Free Problem

The *envy free problem* is the problem of checking whether there exists an allocation which is envy free, i.e. an allocation in which all agents prefer the bundle allocated to them when compared to the bundles allocated to other agents. • Π_{EF} contains the rule:
```
:-¬ef(o).
```

THEOREM 6 (DECISION - CARDINAL PREFERENCES). *Let Θ be a MARA Setting with preferences in one of bundle, k-additive, logic based or weighted goals form. Let Π^Θ_{Pref} be defined according to the table above, and Π^Θ_G, Π_{EFq} and Π_{EF} as before. Then:*

• for each allocation $M_o \in M^\Theta$ such that M_o is envy free, there is an answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{Pref} \cup \Pi_{EFc} \cup \Pi_{EF})$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$.

• for each answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{Pref} \cup \Pi_{EFc} \cup \Pi_{EF})$ there is an allocation $M_o \in M^\Theta$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$ *and M_o is envy free.*

THEOREM 7 (DECISION - ORDINAL PREFERENCES). *Let Θ be a MARA Setting with preferences in logic-based form as prioritized goals. Let the preference relation defined over bundles be based on one of best out, descrimin and leximin orderings. Let Π_{Ordb} be defined according to the table above, and Π^Θ_G, Π^Θ_{PG}, Π_{EFo} and Π_{EF} as before. Then:*

• for each allocation $M_o \in M^\Theta$ such that M_o is envy free, there is an answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{PG} \cup \Pi_{Ordb} \cup \Pi_{EFo} \cup \Pi_{EF})$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$.

• for each answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{PG} \cup \Pi_{Ordb} \cup \Pi_{EFo} \cup \Pi_{EF})$ there is an allocation $M_o \in M^\Theta$ such that `has(a,r,o)`$\in S$ *iff* $r \in M_o(a)$ *and M_o is envy free.*

Finally, we turn to the problem of finding allocations that minimize the overall degree of envyness, given by the sum of individual agent's envyness. • Π_{EFom} contains the rule:
```
:~envy(A1,A2,D,o).[D:1]
```

THEOREM 8 (OPTIMISATION). *Let Θ be a MARA Setting with preferences in one of bundle, k-additive, logic based or weighted goals form. Let Π^Θ_{Pref} be defined according to the table above, and Π^Θ_G, Π_{EFod}, and Π_{EFom} as before. Then:*

• for each allocation $M \in M^\Theta : \nexists M' \in M^\Theta, envy(M') < envy(M)$ there is an answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{Pref} \cup \Pi_{EFod} \cup \Pi_{EFom})$ such that `has(a,r,o)`$\in S$ *iff* $r \in M(a)$.

• for each answer set $S \in AS(\Pi^\Theta_G \cup \Pi^\Theta_{Pref} \cup \Pi_{EFod} \cup \Pi_{EFom})$ there is an allocation $M \in M^\Theta$ such that `has(a,r,o)`$\in S$ *iff* $r \in M(a)$ *and $\nexists M' \in M^\Theta, envy(M') < envy(M)$.*

5. DISCUSSION AND CONCLUSIONS

In this paper we used Answer-Set Programming to provide modular, declarative, sound and complete solutions to 66 different multi-agent resource allocation problems. We have privileged the main MARA problems, for illustrative purposes, leaving out other problems which we have equally solved in ASP, such as determining the next proposal under the Zeuthen strategy in the Monotonic Concession protocol and the winner determination problem in combinatorial auctions (for bids in XOR and OR languages).

Since our main concern was the modularity and declarative nature of the ASP encodings, the solutions presented can, sometimes, be optimised at the cost of modularity. As an example, for the case of egalitarian social welfare we could replace Π_{WOo} with Π_{WOout} and not include $\Pi^{\ominus}_{SWeg}$ where Π_{WOout} contains the rule:

`:~utg(U),ag(A),not ut(A,U,A,o).[U:1]`

Furthermore, the declarative general and modular character of the proposed solutions makes them amenable to several extensions, refinements and addition of constraints. For example, if we wish to impose that two specific resources r_1 and r_2 should always be allocated together, we simply add the rule:

:-ag(A$_1$;A$_2$),has(A$_1$,r$_1$,o),has(A$_2$,r$_2$,o),A$_1$!=A$_2$.

To impose that resource r can only be allocated to the agents a_1 and a_2, we simply add the rule:

:-ag(A),has(A,r,o),A!=a$_1$,A!=a$_2$.

To impose that the utility of each agent should not be less than 10, we simply add the rule:

`:-ag(A),ut(A,U,A,o),U<10.`

To impose that every agent should be allocated at least 5 resources we add the rule:

`:-ag(A),#count{R,A:has(A,R,o)}<5.`

The possibility to easily extend and refine ASP encodings makes the work here presented an excellent tool to investigate and experiment with other MARA notions, of possible interest in some applications, before tailored and more efficient algorithms are developed. For example, we could combine the notions of welfare improvement and envy freeness to check if there is some allocation which improves the value of some welfare function and is envy free by combining both programs i.e. use the program $\Pi^{\ominus}_{G} \cup \Pi^{\ominus}_{Pref} \cup \Pi^{\ominus}_{SW} \cup \Pi_{WI} \cup \Pi_{M_i} \cup \Pi_{EFq} \cup \Pi_{EF}$. The k-rank dictator welfare notion could be implemented using the rule:

sw(S,X):-al(X),w(S),#min{U,A:ut(A,U,A,X),
    #count{A$_1$:ut(A$_1$,U$_1$,A$_1$,X),U>=U$_1$}>=k}=S.

But we could also as easily define new notions and combine with the already existing ones.

Concerning efficiency, lack of space prevents us from showing and discussing our preliminary results. These are, however, quite good given the fact that we are solving problems which are NP at least, but more thorough testing is mandatory, as there are many variables involved. In any case, the uniformity of these encodings makes this implementation an excellent candidate to be used as the basis for benchmarking other more efficient solutions. In this respect, MARA is not the only one that can benefit from answer-set programming. In fact, the programs discussed here can be used as benchmarks for testing different answer set solvers, as well as provide very interesting examples illustrating the use of ASP for solving problems in different complexity classes.

Other nice features include the possibility to use this implementation as an off-the-shelf MARA solver for certain domains where correctness, instead of efficiency, is the crucial requirement.

In what concerns related work, it is worth mentioning the implementation of the winner determination problem in ASP of [3]. Here we chose a more extensive and comprehensive set of MARA problems, although we also have an implementation of the winner determination problem for bids in the XOR and OR languages. Most other implementations of MARA focus on specialized settings, using domain knowledge to improve their efficiency. Thus, they are not comparable with our work in a fair way. On the one hand they are much faster than ours but, on the other hand, they completely lack the generality, modularity, declarative nature, and amenability to change and incorporation of new notions.

Future work includes thorough benchmarking, implementing other kinds of preference representations, extending to other MARA problems of high complexity, studying other constraints and relaxations, and exploring the use of ASP in distributed allocation mechanisms which need to consider the complexity of communication [10].

To conclude, we believe that our modular, declarative, sound and complete solutions can be used as an off-the-shelf implementation in certain scenarios where MARA is needed, and as a framework in which to experiment both with the different notions implemented and with new ones that can easily be incorporated.

6. ACKNOWLEDGMENTS

We would like to thank Martin Slota and the anonymous referees for very helpful comments and suggestions.

7. REFERENCES

[1] K. Arrow, A. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*, volume 1. North-Holland, 2002.

[2] C. Baral. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge Univ. Press, 2003.

[3] C. Baral and C. Uyan. Declarative specification and solution of combinatorial auctions using logic programming. In *Procs. of LPNMR'01*, volume 2173 of *LNAI*. Springer, 2001.

[4] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Procs. of IJCAI'93*, 1993.

[5] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J. A. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30(1):3–31, 2006.

[6] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Multiagent resource allocation in k-additive domains: preference representation and complexity. *Annals of Operations Research*, 163(1):49–62, 2008.

[7] T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate functions in disjunctive logic programming: Semantics, complexity, and implementation in DLV. In *Procs of IJCAI'03*, 2003.

[8] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–418, Sept. 1997.

[9] T. Eiter and A. Polleres. Towards automated integration of guess and check programs in answer set programming: a meta-interpreter and applications. *Theory and Practice of Logic Programming*, 6(1-2):23–60, 2006.

[10] U. Endriss and N. Maudet. On the communication complexity of multilateral trading. *Autonomous Agents and Multi-Agent Systems*, 11(1):91–107, 2005.

[11] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3–4):365–385, 1991.

[12] T. Ibaraki and N. Katoh. *Resource Allocation Problems: Algorithmic Approaches*. MIT Press, 1988.

[13] J. Lang. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

[14] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

[15] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[16] T. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 5, pages 201–258. The MIT Press, 1999.