Finding the Best Agents for Cooperation

F. Buccafurri, L. Palopoli, D. Rosaci, and G. M. L. Sarnè
DIMET, Università "Mediterranea" di Reggio Calabria,
Via Graziella Loc. Feo di Vito, 89100 Reggio Calabria (Italy)
E-mail: {bucca,palopoli,rosaci,sarne}@ing.unirc.it

Abstract.[1] One of the basic issues to face in Multi-Agent Systems is effectively implementing cooperation among agents. To this aim, a not trivial problem has to be solved: Among a (possible large) number of agents, how to detect which agents are promising candidates for cooperation? In this chapter a new approach for dealing with this problem is proposed. We define a formal model for representing agents and a number of semantic properties exploited for detecting fruitful cooperation. On the basis of this model we design a Multi-Agent System, called SPY, capable of managing and supporting cooperation in the agent community. The system learns semantic properties by monitoring the user behavior in such a way that it adapts its response to user expectation.

1 Introduction

Cooperation is often considered as one of the key concepts in Multi-Agent Systems (often denoted by MAS) [19, 12, 13, 18, 11, 35, 4, 15]. Indeed, each agent, in an agent community, does not have to learn only by its own discovery, but also by exploiting knowledge coming from other agents. Cooperation is usually implemented by integrating multiple (even heterogeneous) knowledge sources [2, 3, 26, 25, 22, 31, 5, 36]. But a basic problem concerns the coordination of agent behaviour in order to meet both the knowledge of the interest domain and the individual requirements. For this purpose, learning and adaptation are considered essential by many researchers in this field [32, 8, 28, 34, 29]. In order to realize such a cooperation, some techniques developed in the field of Machine Learning has been introduced in various Multi-Agent Systems [20, 9, 21, 10, 33].

This chapter gives a contribution in this context. Basically we study the following problem: Take a network of agents, each supporting a user, and let the individual knowledge of each agent be an ontology embedding also knowledge about the user behavior. When a user would contact other agents, in order to integrate their ontologies with that of her/his agents, or, in other words, when a user ask for cooperation of other agents, how she/he can select, among the universe of possible agents, the most appropriate ones?

For giving a solution to the above problem we propose a formal framework in which we represent by quantitative information a number of semantic properties that we consider important for detecting fruitful cooperation. Such properties take into account both structural similarities among agent ontologies, user perception, and attraction power of agents in the

[1] A short abridged version of this chapter appeared in the Proceedings of the Second International Conference on Intelligent Agent Technology 2001, pp. 44–53, World Scientific.

community. Potentially appropriate agents for cooperating with an agent, say a, of the net, are obtained by solving a linear system. The solution of such a system provides the user of a with a number of agent lists, each containing the *most appropriate* agents for cooperation, from which the user can choose agents she/he want to contact for supporting her/his activity. The multiplicity of such choice lists depends on the multiplicity of the properties that can be used as preference criteria. Users are free to use the suggested lists even partially, or to ignore them. In any case, user's behaviour induces a modification of some coefficients (describing reactive properties) in such a way that lists suggested in the future are (hopefully) closer to real user needs. Therefore, the system *learns* from user's behaviour about how to provide her/him with suggestions meeting as much as possible their expectancy.

On the basis of this model, we design a Multi-Agent System with a client-server architecture: Client agents, possibly cooperating with one another, support user activity, while the server side coordinates cooperation. Among all possible applications, we have considered the case of agent cooperation for helping the user in retrieving information. The user is provided with a set of recommendations that are generated both by her/his agent and by agents the system has detected as promising for cooperation.

The chapter is organized as follows. Section 2 relates our work with other proposals. Section 3 illustrates how individual knowledge of agents is represented in our model. Section 4 is the core of the chapter, since it describes which semantic properties we represent and how we model and solve the problem of cooperation. Section 5 describes how cooperation between two agents is implemented, that is the integration of the two respective knowledge bases. In Section 6, the system SPY, implementing the cooperation model, is presented. Section 7 gives a practical description of a system execution session and some discussion about the system behavior. Finally, Section 8 gives the conclusions.

2 Related Work

In the context of Machine Learning approaches, [34] illustrates the progress made in the available work on learning and adaptation in Multi-Agent Systems, and provides a general survey of Multi-Agent Systems using adaptation and learning. In [32], a demonstration of how reinforcement-learning agents can learn cooperative behavior in a simulated social environment is provided, specifying that if cooperation is done intelligently, each agent can benefit from other agents instantaneous information, episodic experience and learned knowledge. [8] concerns with how a group of intelligent agents can work together in order to solve a problem or achieve a common goal, by using Machine Learning techniques to refine their knowledge. An example of a practical situation that needs to be modeled in a group of agents is presented in [28], where a probabilistic reciprocity mechanism is introduced to generate stable and cooperative behavior among a group of self-interested agents. In [30], authors identify procedures and environments under which self-interested agents may find it beneficial to help others and point out that sharing of experiences about other agents among reciprocative agents will limit the exploitative gains of selfish agents.

A large numbers of Multi-Agent Systems using learning techniques have been proposed in literature. Among these, we cite some significant proposals:

- In [26], a learning system, called COLLAGE, that endows the agents with the capability to learn how to choose the most appropriate coordination strategy from a set of available coordination strategy, is presented.

- *Amalthaea* [21], that is an evolving Multi-Agent Ecosystem for personalized filtering, discovery and monitoring of information sites.
- [10] presents an information retrieval system where a multi-agent learning approach to information retrieval on the Web is proposed, in which each agent learns its environment from the user's relevance feedback by using a neural network mechanism.
- In [20] a system of collaborative agents is proposed, where the collaboration among agents assisting different users is introduced in order to improve the efficiency of the local learning.
- The system *Challenger* [9], consisting of agents which individually manage local resources and communicate with one another to share their resources in the attempt of utilizing them more efficiently, in order to obtain desirable global system objectives.
- [33] presents a multi-agent architecture applied to Cooperative System Engineering, useful in modeling activities and providing support to cooperative tasks.
- In [6, 7] the authors introduce a principle, called *Win or Learn Fast* (WOLF), for varying a learning rate in a new reinforcement learning technique. They examine this technique theoretically and also present empirical results on a variety of more general stochastic games.
- [27] aims to establish a mechanism that enables agents to cope with environments that contain both selfish and co-operative entities, where the mixture and the behavior of these entities is previously unknown to all agents. Authors achieve this by enabling agents to evaluate trust in others, based upon the observations they gather.

Such techniques open, on the one hand, the possibility of integrating individual agent knowledge for acquiring an enhanced knowledge of the environment. On the other hand, they consider the problem of determining which agents are promising candidates for suitable knowledge integration, but, differently from our approach, none of them proposes automatic techniques for solving such a problem.

We point out that in the above approaches the knowledge involved in the cooperative exchange is not stored in a complex data structure, but it generally consists of unstructured elementary information about both the environment and the various actions performed by the agents. But an important issue recently emerged in the MAS field deals with the necessity of organizing the available knowledge in *ontologies* [23, 16, 17, 14], that are sophisticated content oriented data structures.

3 The Knowledge Bases

Throughout the chapter we refer to a given set of agents Λ of cardinality n and we suppose that all agents in Λ can cooperate with each other. Thus we can see the set Λ as a undirected complete graph of agents whose arcs represent possible cooperation. W.l.o.g., we identify agents in Λ by the cardinal numbers $\{1, ..., n\}$.

3.1 An Ontology for describing the domain of interest

Since we consider only homogeneous agents, we suppose that a unique environment is associated with our agent net. We represent such an environment in our model by a set of objects. For the rest of the section we consider a set of objects O as given.

A domain D on O is a set of classes of objects. We suppose that a suitable semantics underlying the classification provided in a domain is given. The notion of domain is formally defined next.

Definition 3.1. *A Domain on O* , denoted by *D* , is a set $D \subseteq 2^O$ such that both: (1) $\forall o \in O$, $\{o\} \in D$, and (2) there exists an element r of D, called *the root* such that, $\forall o \in O$, $o \in r$. Elements of D are called *classes*.

In words, a domain is a set object classes containing, at least, a root class collecting all the objects of O and, for each object o of O, the singleton class $\{o\}$. Throughout the rest of the section, we assume a domain D on O as given.

Set containment induces a partial ordering among elements of a domain D. A domain D plus this ordering in called *ontology*.

Definition 3.2. An *ontology* on D, denoted by O_D, is a partially ordered set $\langle D, \subseteq \rangle$. The *ontology graph of* O_D is a directed graph $G(O_D)$ with set of nodes coinciding with D an set of arcs A defined as the binary relation obtained as transitive reduction of the relation $\subseteq$ of O_D[2]. The node r of $G(O_D)$, where r is the root of D, is called *root of* $G(O_D)$.

Note that, as a consequence of item (2) of Definition 3.1, each non-root node is reachable from the root r of $G(O_D)$. Furthermore, by item (1) of Definition 3.1, nodes of $G(O_D)$ with out-degree 0 coincide with singleton classes of D. An ontology based on a generalization hierarchy is suitable for representing many real-world situations, like the topics in Web engines and in Web sites, the items in e-commerce, the staff hierarchy of an organization and so on. It is worth noting that this is not the only possible choice for representing the environment of agents. Indeed, in different contexts, as semi-structured data in Web documents, other kinds of ontologies may be better suited (for example *OEM-graphs* [1], *SDR-networks* [24], etc.).

3.2 The Local Knowledge Base

The ontology represents the common knowledge about the environment in which the agents work. However, each agent may have a partial view of the ontology representing the portion of the world the user monitored by the agent selects by her/his activity. Inside this portion of the ontology, different priorities for the classes can be inferred by exploiting user behaviour. This is encoded in the notion of *the Local Knowledge Base* (LKB for short), defined next.

Definition 3.3. Given an ontology O_D on D and an agent a, a *Local Knowledge Base* LKB^a (of a on O_D), is a pair $\langle G^a, \rho^a \rangle$, such that: (i) $G^a = \langle N^a, A^a \rangle$ is a sub-graph of $G(O_D)$ (i.e., $N^a \subseteq D$, $A^a \subseteq A$) containing the root r of $G(O_D)$ and such that each $n \in N^a$ is reachable in G^a from r, and (ii) ρ is a function, called *priority function*, defining a real weight ranging from 0 to 1 associated to each arc (i, j) of G^a such that:

$$\rho(i,j) = \frac{c_{ij}}{\sum_{k \in Adj(i)} c_{ik}}$$

[2] (A, B) is in the transitive-reduction of $\subseteq$ iff $A \subseteq B$ and $\nexists\, C$ such $A \subseteq C$ and $C \subseteq B$.

where $Adj(i)$ is the set of nodes adjacent to i, and for each $k \in Adj(i)$, c_{ik} counts how many times the user of a has selected an object (that is, a leaf node) through a path selection including the arc (i, k). Note that coefficients c_{ij} in a path $\langle r, i_1, ...i_s \rangle$ are updated only when the leaf node i_s, corresponding to a single object of the domain, is selected. The root r of $G(O_D)$ is also called the *root of* LKB^a.

A Local Knowledge Base, representing the local view of the agent, is then obtained by extracting a sub-graph from the ontology graph including all the classes accessed by the user (and thus at least the root node). Moreover, arcs of the so obtained graph are weighted for assigning highest priority to most accessed classes.

4 Extraction of the Semantic Properties

Besides his/her local agent, each user looks at the other agents of the net as a source of potentially interesting information in order to enrich the support to his/her activity. Interest in agents can be defined by considering some semantic properties. Such properties, useful for driving users' choices are of two types: (i) *local properties*, taking into account information stored in the LKBs, and (ii) *global properties*, merging local properties with external knowledge extracted from the general context. An important feature of the model is that the merge performed in the construction of global properties is based on an adaptive learning technique involving some parameters by which the user behaviour is taken into account. In other words, global properties exploit an important kind of properties (encoded in a number of parameters) directly reflecting reactions of users to system advice. We call such additional properties *reactive properties*. Next we describe the set of properties used in the model.

4.1 Local properties: Similarity

The only local property we consider is the property we call *similarity* between two agents i and j, representing a measure of the similarity of the two corresponding LKBs. Such a coefficient is a real value ranging from 0 to 1.

Definition 4.1. Let i and j be two agents. Let $G^i = \langle N^i, A^i \rangle$ and $G^j = \langle N^j, A^j \rangle$ be the two graphs of their LKBs. Let ρ^i and ρ^j be the corresponding priority functions. We define the *similarity* S_{ij} between i and j as:

$$S_{ij} = 1 - \frac{1}{\mid A^i \cup A^j \mid} \sum_{(h,k) \in A^i \cup A^j} \gamma_{hk}$$

where

$$\gamma_{hk} = \begin{cases} \mid \rho^i(h,k) - \rho^j(h,k) \mid & \text{if } (h,k) \in A^i \cap A^j \\ 1 & \text{otherwise.} \end{cases}$$

Observe that the term $\frac{1}{|A^i \cup A^j|} \sum_{(h,k) \in A^i \cup A^j} \gamma_{hk}$ in the expression defining S_{ij} (for two agents i and j) represents a *dissimilarity* between agents i and j. This is defined as a mean of a number of contributions γ_{hk}, each corresponding to an arc (h, k) belonging to the set $A^i \cup A^j$. For common arcs of the two LKBs, that is, arcs belonging to the intersection between A^i and A^j, γ_{hk} is the difference (in absolute value) between the respective priority functions (note

that such a difference is a real value ranging from 0 to 1). In words, common arcs can be view as "homologous" arcs, and their dissimilarity measures how much these arcs differ in terms of weight. For the remaining arcs $(h,k) \notin A^i \cap A^j$, we assign the value 1 to the coefficient γ_{hk}. Indeed, an arc belonging to A^i but not belonging to A^j has not a "homologous" arc in the LKB graph of the agent j (and vice versa), and thus this is the case of maximum dissimilarity, leading to a contribution (to the overall dissimilarity) saturated to the value 1.

4.2 Global Properties: Interest and Attractiveness

Recall that global properties merge local properties with knowledge extracted from the context. In this section we introduce the notion of *interest coefficient*, representing just a measure of the global properties of a given agent as perceived by another one. Hence, for a pair of agents i and j, the interest coefficient, besides the similarity between i and j, must take into account also knowledge extracted from the context. But *which kind of contextual knowledge has to be considered as meaningful?*. The choice we make in our model is the following: The knowledge extracted from the context, used by the agent i for defining the interest coefficient I_{ij} w.r.t. another agent j, is a measure of the global interest of all the other agents (different from i) w.r.t. the agent j, that is a measure of a sort of *attractiveness* of the agent j as perceived by the agent i. Recalling that the interest, besides the contextual knowledge, must take into account also the local knowledge (that is, the similarity), the above definition of contextual knowledge leads to require that, for each $i \in \Lambda \setminus \{j\}$:

$$I_{ij} = \phi_{ij}(S_{ij}, \mu_{ij}(\{I_{kj} \mid k \neq i,j\})) \tag{1}$$

where μ_{ij} and ϕ_{ij} are suitable functions yielding real values from 0 to 1. In particular, μ_{ij} returns a measure of the attractiveness of the agent j detected by the agent i from the value of the interest coefficients of all the agents (different from i) w.r.t j, while ϕ_{ij} combines such a measure with the similarity S_{ij}. Clearly, the function ϕ_{ij} plays also the role of weighing the importance for the agent i of the local knowledge w.r.t. the contextual one.

For μ_{ij} and ϕ_{ij} (where i and j are two agents) we adopt in our model the following choices: (i) μ_{ij} is a function returning the average of the interest coefficients of all the other agents different from j, (ii) ϕ_{ij} is a function computing a linear combination of the similarity coefficient between i and j and the attractiveness of j w.r.t i. Applying the above definitions for μ_{ij} and ϕ_{ij}, (1) becomes the following linear system:

$$(\forall i \in \Lambda \setminus \{j\}) \quad \Big(I_{ij} = \psi_{ij} \cdot (P_i \cdot S_{ij} + (1 - P_i) \cdot S_{ij} \cdot \tfrac{1}{n-2} \textstyle\sum_{k \in \Lambda \setminus \{i,j\}} I_{kj})\Big) \tag{2}$$

where ψ_{ij} and P_i, for each $i \in \Lambda \setminus \{j\}$, are adaptive parameters ranging from 0 to 1 representing a measure of *reactive* properties that we suppose to be learned from the user behaviour. ψ_{ij} plays the role of a reducing factor, filtering the advice of the system on the basis of the user behaviour, while P_i measures the importance that the user gives to the local knowledge (similarity) w.r.t. the contextual one. Note that both ψ_{ij} and P_i can be estimated once the reactive properties are defined. We deal with this issue in the next section. Thus, given an agent j, any value assignment to the interest coefficients of all the other agents w.r.t. j must satisfy (2). The next theorem shows that, for every value of the parameters occurring in (2), there exists a unique solution of the linear system (2), that is a value assignment to the interest coefficients satisfying (2). Obviously, such a solution can be polynomially computed.

Lemma 4.1. *Given an agent $j \in \Lambda$ and a set of 3-tuples of $[0,1]$ real coefficients $\{\langle P_i, \psi_{ij}, S_{ij}\rangle \mid i \in \Lambda \setminus \{j\} \wedge (\forall h \in \Lambda \setminus \{j\},\ P_h \neq 0 \vee \psi_{ij} \neq 1 \vee S_{ij} \neq 1)\}$, then there exists a unique solution of the system (2).*

Proof. The lemma is trivially true in case $0 \leq n \leq 2$. Thus, we have to prove the claim for $n > 2$. To this aim, it is sufficient to show that the rank r of the coefficient matrix H of (2) is full, i.e., $r = n - 1$[3]. W.l.o.g., just for notation convenience, we suppose $1 < j < n$. The coefficient matrix H is:

$$H = \begin{pmatrix} -1 & \psi_{1j} \cdot S_{1j} \cdot \frac{1-P_1}{n-2} & \ldots & \psi_{1j} \cdot S_{1j} \cdot \frac{1-P_1}{n-2} \\ \ldots & \ldots & \ldots & \ldots \\ \psi_{nj} \cdot S_{nj} \cdot \frac{1-P_n}{n-2} & \psi_{nj} \cdot S_{nj} \cdot \frac{1-P_n}{n-2} & \ldots & -1 \end{pmatrix}$$

We proceed by contradiction supposing that $r < n - 1$. In such a case, there exists a row i of H that can be expressed as linear combination of the other rows by means of $n - 2$ coefficients, say a_h, $h = 1..n$, $h \neq i, j$. In particular, for the diagonal element $H(i,i) = -1$ the following holds:

$$H(i,i) = \sum_{h=1..n, h\neq i,j} \psi_{hj} \cdot S_{hj} \cdot \frac{1-P_h}{n-2} \cdot a_h = -1. \tag{3}$$

For the other elements $H(i,t)$ of the row i, where $t = 1..n$, $t \neq i, j$, we obtain:

$$H(i,t) = \psi_{tj} \cdot S_{tj} \cdot \frac{1-P_t}{n-2} = -a_t + \sum_{h=1..n, h\neq i,j,t} \psi_{hj} \cdot S_{hj} \cdot \frac{1-P_h}{n-2} \cdot a_h \tag{4}$$

that is:

$$\begin{aligned} \psi_{tj} \cdot S_{tj} \cdot \tfrac{1-P_t}{n-2} = \quad & -a_t \left(1 + \psi_{tj} \cdot S_{tj} \cdot \tfrac{1-P_t}{n-2}\right) + \\ & + \textstyle\sum_{h=1..n, h\neq i,j} \psi_{hj} \cdot S_{hj} \cdot \tfrac{1-P_h}{n-2} \cdot a_h. \end{aligned} \tag{5}$$

Exploiting (3), (5) becomes:

$$\psi_{tj} \cdot S_{tj} \cdot \frac{1-P_t}{n-2} = -a_t \cdot \left(1 + \psi_{tj} \cdot S_{tj} \cdot \frac{1-P_t}{n-2}\right) - 1 \tag{6}$$

from which we derive that $a_t = -1$. For symmetry, $a_h = -1$ for each $h = 1..n$, $h \neq i, j$. As a consequence, (3) becomes:

$$\sum_{h=1..n, h\neq i,j} \psi_{hj} \cdot S_{hj} \cdot \frac{1-P_h}{n-2} = 1. \tag{7}$$

Hence, we have reached a contradiction, since by hypothesis for each h, either $P_h > 0$ or $\psi_{hj} < 1$ or $S_{hj} < 1$. Thus, (7) is false since $\sum_{h=1..n, h\neq i,j} \psi_{hj} \cdot S_{hj} \cdot \frac{1-P_h}{n-2} < 1$.

Theorem 4.1. *Given an agent $j \in \Lambda$ and a set of 3-tuples of $[0,1]$ real coefficients $\{\langle P_i, \psi_{ij}, S_{ij}\rangle \mid i \in \Lambda \setminus \{j\} \wedge (\forall h \in \Lambda \setminus \{j\},\ P_h \neq 0 \vee \psi_{ij} \neq 1 \vee S_{ij} \neq 1)\}$, there exists a unique $(n-1)$-tuple of $[0,1]$ real values $S = \langle I_{1j}, \ldots, I_{(j-1)j}, I_{(j+1)j}, \ldots I_{nj}\rangle$ satisfying (2).*

[3] Recall that the size of H is $n - 1$

Proof. Existence and uniqueness is ensured by Lemma 4.1. Thus, we have only to prove that I_{hj}, for each $h \in \Lambda \setminus \{j\}$, belongs to the interval $[0, 1]$.The theorem is trivially true in case $0 \leq n \leq 2$. Thus, we have to prove the claim for $n > 2$.

$I_{hj} \geq 0$, for each $h \in \Lambda \setminus \{j\}$. We start by proving that $I_{hj} \geq 0$, for each $h \in \Lambda \setminus \{j\}$. In particular, we show that the set $V_j = \{h \in \Lambda \setminus \{j\} \mid I_{hj} < 0\}$ is empty. We proceed by contradiction, supposing that $V_j \neq \emptyset$. Let $W_j = \{h \in \Lambda \setminus \{j\} \mid I_{hj} \geq 0\}$.

(2) can be rewritten as follows:

$$I_{ij} = \psi_{ij} \cdot S_{ij} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \left(\sum_{r \in V_j} I_{rj} + \sum_{r \in W_j} I_{rj} - I_{ij} \right) \right).$$

thus:

$$I_{ij} \cdot \left(1 + \frac{\psi_{ij} \cdot S_{ij} \cdot (1 - P_i)}{n-2} \right) = \psi_{ij} \cdot S_{ij} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \left(\sum_{r \in V_j} I_{rj} + \sum_{r \in W_j} I_{rj} \right) \right).$$

Now, posing $a_{ij} = 1 + \frac{\psi_{ij} \cdot S_{ij} \cdot (1-P_i)}{n-2}$ and applying the summation for each $i \in V_j$, we obtain:

$$\sum_{i \in V_j} I_{ij} = \sum_{i \in V_j} \frac{\psi_{ij} \cdot S_{ij}}{a_{ij}} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \left(\sum_{r \in V_j} I_{rj} + \sum_{r \in W_j} I_{rj} \right) \right)$$

from which we derive:

$$\sum_{r \in V_j} I_{rj} \cdot \left(1 - \sum_{i \in V_j} \frac{\psi_{ij} \cdot S_{ij} \cdot (1 - P_i)}{a_{ij} \cdot (n-2)} \right) = \sum_{i \in V_j} \frac{\psi_{ij} \cdot S_{ij}}{a_{ij}} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \sum_{r \in W_j} I_{rj} \right).$$

Posing $b_j = \sum_{i \in V_j} \frac{\psi_{ij} \cdot S_{ij} \cdot (1-P_i)}{a_{ij} \cdot (n-2)} = \sum_{i \in V_j} \frac{1}{1 + \frac{n-2}{C_{ij} \cdot S_{ij} \cdot (1-P_i)}}$, we obtain:

$$\sum_{r \in V_j} I_{rj} = \frac{1}{1 - b_j} \cdot \sum_{i \in V_j} \frac{\psi_{ij} \cdot S_{ij}}{a_{ij}} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \sum_{r \in W_j} I_{rj} \right). \tag{8}$$

Since V_j is not empty by hypothesis, $\sum_{r \in V_j} I_{rj} < 0$. As a consequence, (8) is false, since its right-hand term is greater than or equal to 0. Indeed, as it can be easily verified, $b_j < 1$. We have thus reached a contradiction and hence the set V_j must be empty. It remains to prove that $I_{hj} \leq 1$, for each $h \in \Lambda \setminus \{j\}$.

$I_{hj} \leq 1$, for each $h \in \Lambda \setminus \{j\}$. We shall demonstrate that the set $M_j = \{h \in \Lambda \setminus \{j\} \mid I_{hj} > 1\}$ is empty. We proceed by contradiction, supposing that $M_j \neq \emptyset$. Let $N_j = \{h \in \Lambda \setminus \{j\} \mid I_{hj} \leq 1\}$.

First observe that M_j cannot be a singleton. Indeed, if in the a tuple $S = \langle I_{1j}, \ldots, I_{(j-1)j}, I_{(j+1)j}, \ldots I_{nj} \rangle$ just one element, say it I_{hk}, is greater than 1, then (2) is not satisfied, as $\psi_{hk} \cdot S_{hk} \cdot (P_h + (1 - P_h) \cdot \frac{1}{n-2} \sum_{r \in \Lambda \setminus \{h,k\}} I_{rk}) \leq 1$.

Thus, we have to consider only the case $|M_j| \geq 2$. Denote by s the cardinality of M_j.

First, rewrite (2) as follows:

$$I_{ij} = \psi_{ij} \cdot S_{ij} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \left(\sum_{r \in M_j} I_{rj} + \sum_{r \in N_j} I_{rj} - I_{ij} \right) \right)$$

thus:

$$I_{ij} \cdot \left(1 + \psi_{ij} \cdot S_{ij} \cdot \frac{(1 - P_i)}{n-2} \right) = \psi_{ij} \cdot S_{ij} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \left(\sum_{r \in M_j} I_{rj} + \sum_{r \in N_j} I_{rj} \right) \right).$$

Now, posing $a'_{ij} = 1 + \psi_{ij} \cdot S_{ij} \cdot \frac{(1-P_i)}{n-2}$ and applying the summation for all $i \in M_j$, we obtain:

$$\sum_{i \in M_j} I_{ij} = \sum_{i \in M_j} \frac{\psi_{ij} \cdot S_{ij}}{a'_{ij}} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \left(\sum_{r \in M_j} I_{rj} + \sum_{r \in N_j} I_{rj} \right) \right)$$

from which we derive:

$$\sum_{i \in M_j} I_{ij} \cdot \left(1 - \sum_{i \in M_j} \frac{\psi_{ij} \cdot S_{ij} \cdot (1 - P_i)}{a'_{ij} \cdot (n-2)} \right) = \sum_{i \in M_j} \frac{\psi_{ij} \cdot S_{ij}}{a'_{ij}} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \sum_{r \in N_j} I_{rj} \right).$$

Posing $b'_j = \sum_{i \in M_j} \frac{\psi_{ij} \cdot S_{ij} \cdot (1-P_i)}{a'_{ij} \cdot (n-2)} = \sum_{i \in M_j} \frac{1}{1 + \frac{n-2}{\psi_{ij} \cdot S_i \cdot (1-P_i)}}$, we have:

$$\sum_{r \in M_j} I_{rj} = \frac{1}{1 - b'_j} \cdot \sum_{i \in M_j} \frac{\psi_{ij} \cdot S_{ij}}{a'_{ij}} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \sum_{r \in N_j} I_{rj} \right). \tag{9}$$

The right-hand term of (9) is upper-bounded by

$$\sum_{i \in M_j} \frac{1}{1 + \frac{1-P_i}{n-2}} \cdot \left(P_i + \frac{1 - P_i}{n-2} \cdot \sum_{r \in N_j} I_{rj} \right).$$

In turn, the latter is upper-bounded by:

$$\sum_{i \in M_j} \frac{1}{1 + \frac{1-P_i}{n-2}}$$

since $\sum_{r \in N_j} I_{rj} \leq n - s \leq n - 2$.

But, $\frac{1}{1 + \frac{1-P_i}{n-2}} < 1$, for each $i \in M_j$ and, thus, the right-hand term of (9) is less than s. As a consequence, (9) is false, since $\sum_{r \in M_j} I_{rj} > s$ and thus we have reached a contradiction. This concludes the proof.

The above result allows us to define the *interest coefficients list* of an agent j as the unique solution of (2).

Definition 4.2. Given an agent $j \in \Lambda$, the *interest coefficient list of* j is the unique $(n-1)$-tuple of real values $\langle I_{1j}, \ldots, I_{(j-1)j}, I_{(j+1)j}, \ldots I_{nj} \rangle$ satisfying (2). Given an agent $i \neq j$, the *interest coefficient* of i w.r.t j is the value I_{ij} occurring in the interest coefficient list of j.

Besides the interest property, from the knowledge of the interest coefficients lists, agents can exploit a second type of property. Indeed, an agent can compare different agents on the basis of their *attractiveness coefficient*, representing the component of the interest capturing only the contextual knowledge.

Definition 4.3. Given a pair of agents $i, j \in \Lambda$, the *attractiveness of* j *perceived by* i, is the real coefficient A_{ij} (ranging from 0 to 1) defined as: $A_{ij} = \frac{1}{n-2} \sum_{k \in \Lambda \setminus \{i,j\}} I_{kj}$, where $\langle I_{1j}, \ldots, I_{(j-1)j}, I_{(j+1)j}, \ldots I_{nj} \rangle$ is the interest coefficients list of the agent j.

4.3 Choice Lists

Suppose the user of an agent i has the intention of contacting other agents in order to establish a cooperation. Suppose the similarities between i and every other agent is known as well as both the interest coefficient of i w.r.t. every other agent and the attractiveness of all the agents perceived by i. As previously discussed, such values can be effectively computed once a number of parameter are set (actually, they can be suitably initialized and updated by learning from the behaviour of the user, as we shall explain in the sequel). Thus, three agent lists can be presented to the user *i* associated to the agent i, each associated with one of the properties similarity, interest and attractiveness. We denote these lists $L_S(i)$, L_I(i), and $L_A(i)$. $L_S(i)$ ($L_I(i)$, $L_A(i)$, resp.) is the list of the $n-1$ agents j (different from i) ordered by decreasing similarity (interest, attractiveness, resp.) coefficient S_{ij} (I_{ij}, A_{ij}, resp.). When the user i chooses an agent j from the list $L_S(i)$ ($L_I(i)$, $L_A(i)$, resp.), it means that she/he perceived only the property of similarity (interest, attractiveness, resp.) about the agent j. From the choices of the users, useful knowledge can be thus drawn, which is potentially usable as feedback for correcting advice given to them. This issue is discussed in the next section.

4.4 Reactive Properties

For reactive properties we mean properties describing reactions of users to the suggestions received from the system at a given time, that must be taken into account for adapting future responses of the system. We implement such adaptation of the system to the user behaviour by including into the interest coefficient definition (see Section 4.2) some specific coefficients that are automatically updated during system running. In this section we describe both the role of such coefficients and the rules defining their adaptation to user behaviour. Recall that, given a pair of agents i and j, for defining the interest coefficient I_{ij}, two parameters P_i and ψ_{ij} must be set. They are real parameters ranging from 0 to 1. P_i encodes the *preference property* and is called *preference coefficient* of the agent i, while ψ_{ij} is the product $B_{ij} \cdot C_{ij}$ between the *benevolence coefficient* B_{ij} and *consent coefficient* C_{ij}, resp., of i w.r.t. j. Recall that, given an agent i, by $L_S(i)$, $L_I(i)$, and $L_A(i)$, we denote the three choice lists presented to the user of agent i by the system.

The Preference Property. It is described by a real coefficient ranging from 0 to 1, denoted by P_i, and called *preference coefficient*. The property measures how much for an agent i the

similarity is more important than the attractiveness property for defining global properties. It is easily recognizable that in the definition of interest given in Section 4.2 the coefficient P_i plays just this role. Now we define how the coefficient P_i is updated. Suppose that at a given time the user of the agent i makes a selection of agents. Denote by SI_i (SS_i, SA_i, resp.) the set of the agents that the user has selected from the list $L_I(i)$ ($L_S(i)$, $L_A(i)$, resp.). Such choices are interpreted in order to define how to update the coefficient P_i. The reasoning we adopt is the following: the ratio between the number of agents selected according to the similarity property and the total number of selected agents, provides us a perception of the importance the user gives to the similarity versus the attractiveness. Thus, such a ratio could be used for evaluating the *new* value of P_i. How to *infer* the number of agents selected for their similarity? Certainly all the agents of SS_i are chosen for their similarity. On the contrary, it is reasonably assume that agents of SA_i do not give any contribution to the above number, since they has been chosen *only* on the basis of the attractiveness property. What about agents in SI_i? Here, the choice was done on the basis of the interest property, which mixes similarity and attractiveness. But we can use the old value of P_i for inferring which portion of SI_i has been chosen for the similarity. And this is coherent with the semantic we have given to the preference coefficient. Thus, the total number of agents chosen on the basis of similarity can be assumed to be $P_i \cdot |SI_i| + |SS_i|$. Taking into account the above observations, updating P_i after a selection step is defined as:

$$P_i = \frac{1}{2} \cdot \left(\frac{\mid SI_i \mid \cdot P_i + \mid SS_i \mid}{\mid SI_i \mid + \mid SS_i \mid + \mid SA_i \mid} + P_i \right)$$

where $\mid SI_i \mid + \mid SS_i \mid + \mid SA_i \mid$ is the total number of selected agents. This updating is obtained by computing the average between the new contribution with the old value of P_i in order to keep memory of the past and avoiding drastic changing of the coefficient.

The Benevolence Property. This property measures a sort of availability of the agent j to which a user i requires to share knowledge. Such a property is used in order to weight the interest of i w.r.t. j. For instance, an agent j that recently, and for several times, has denied collaboration in favor of i should become of little interest for i. The parameter encoding such a knowledge is called *benevolence coefficient*, denoted by B_{ij}, and takes real values ranging from 0 to 1. $B_{ij} = 0$ (resp., $B_{ij} = 1$) means the agent j is completely unavailable (resp., available) to fulfill the requests of i. The response of j to requests of i updates the value of B_{ij} according to the following rules:

$$B_{ij} = \begin{cases} min(1, B_{ij} + \delta) & \text{if } j \text{ grants the request of } i \\ max(1, B_{ij} - \delta) & \text{if } j \text{ denies the request of } i \end{cases}$$

where δ is a (reasonably small) positive real value.

The Consent Property. This property describes how much the user of an agent i trusts suggestions of the system regarding another agent j done on the basis of the interest property. The coefficient associated with this property is denoted by C_{ij} and is called *consent coefficient*. The updating rules defining how to adapt the coefficients C_{ij} after a user selection step take into account only the portion of the selection performed on the list $L_I(i)$. Indeed, from this portion of the user selection, we can draw information about the opinion of the user about the suggestions provided by the system. For instance, if the user of i completely trusts the system capability of providing the best suited agent for cooperation by providing the list $L_I(i)$ she/he will choose exactly only the first k agents appearing in $L_I(i)$, where k is the size

of the portion of her/his selection extracted from $L_I(i)$. This is not in general the case, that is, some of the k agents chosen from $L_I(i)$ do not occur in the set of the first k agents of $L_I(i)$. We defined updating rules by taking into account the above observations according to the following idea: every agent h chosen by the user from $L_I(i)$ produces a gain of the consent coefficient C_{ih} if h is a candidate from the system to be selected, produces an attenuation of C_{ih} otherwise. More formally, given an agent i and a selection S_i (set of agents) extracted by the user of i from $L_I(i)$, for each $h \in S_i$:

$$C_{ih} = \begin{cases} min(1, C_{ih} + \delta) & \text{if } h \text{ appears among the first } |S_i| \text{ elements of } L_I(i) \\ max(0, C_{ih} - \delta) & \text{otherwise} \end{cases}$$

where δ is a (reasonably small) positive real value.

5 Integration of Knowledge Bases

Cooperation between two agents is implemented in our model by the integration of their LKBs. Thus, the user of an agent i which has selected an agent j from one of the three choice lists can exploit the cooperation of j by consulting the *Integrated Knowledge Base* LKB^{ij}, obtained by integrating LKB^i with LKB^j. We show next how LKB^{ij} is defined. Once the LKB^{ij} has been computed, the integration of the knowledge of the agent j with that of the client agent i is simply implemented by replacing its LKB with the new LKB^{ij}.

Definition 5.1. Let i be an agent. Let $L \in \{L_S(i), L_I(i), L_A(i)\}$ be one of the choice lists of i and j be an agent selected by i from the list L. Let $G^i = \langle N^i, A^i \rangle$ and $G^j = \langle N^j, A^j \rangle$ be the two graphs of their LKBs .

The *Integrated Knowledge Base* of j *in* i, denoted by IKB^{ij}, is the pair $\langle G^{ij}, \rho^{ij} \rangle$, where $G^{ij} = \langle N^i \cup N^j, A^i \cup A^j \rangle$ and $\rho^{ij}(h,k)$ (i.e., the priority function) is computed on the basis of the coefficients c_{hk} of the arcs outgoing h in G^{ij} as defined by Definition 3.3. Such coefficients, according to the semantics we have given to the priority function, are defined as follows:

$$c_{hk} = \begin{cases} c^i_{hk} & \text{if } (h,k) \in A^i \setminus A^j \\ c^j_{hk} & \text{if } (h,k) \in A^j \setminus A^i \\ c^i_{hk} + c^j_{hk} & \text{if } (h,k) \in A^i \cap A^j \end{cases}$$

where we denote by a superscript the source ontology the coefficients refer to.

In words, the coefficient of an arc in the integrated LKB is obtained by copying the corresponding coefficient from the source LKB, say it i, in case the arc belongs only to i, by summing up the corresponding coefficients, in case the arc appears in both LKBs. The integration process is further illustrated by the following example.

Example 5.1. Consider two agents a and b working in a musical environment and let L_A and L_B of Figure 1 be their respective LKBs (for brevity, we do not show in the figure the common ontology on which the two LKBs are defined). In Figure 1, the LKB L_{AB} obtained by integrating the LKBs L_A and L_B is also shown. Observe that, according to Definition 5.1, the coefficients of the arcs $(Music, Rock)$ and $(Rock, Springsteen)$ in L_{AB} have been obtained as sum of the coefficients of the corresponding arcs in L_A and L_B. Indeed, such arcs

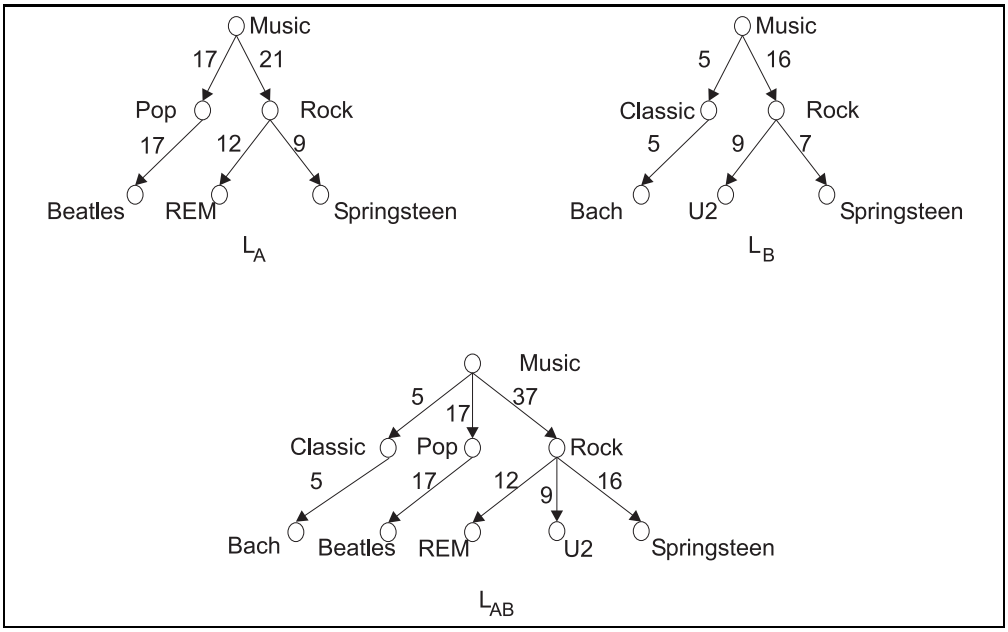

Figure 1: Integration of two LKBs

appear both in L_A and in L_B. Moreover, all the other arcs of L_{AB} keep the coefficient of the LKB from which they come. For instance, the coefficient of the arc $(Classic, Bach)$ is 5, corresponding to the value weighting the arc $(Classic, Bach)$ in the LKB L_B.

Once the integration process is completed and the LKB^{ij} has been computed, we update the LKB of i to be $LKB^i = LKB^{ij}$.

6 The SPY System Architecture

The SPY system implements the model described in the previous sections. It is based on a client-server architecture as shown in Figure 2. Cooperation of client agents is coordinated and managed from the server side (i.e., the *Agency*). Client agents, possibly cooperating each other, support user activity. Among all possible applications, we have considered the case of agent cooperation for helping the user in retrieving information. The user is provided with a set of recommendations that are generated both by her/his agent and by agents the system has detected as promising for cooperation. The server maintains a copy of the LKB of each agent in the network, updating it each time a change arises. Figure 2 zooms in on the (generic) client agent i, showing its architecture.

It consists of 4 main modules, that are: (1) the *Interface*, (2) the *Knowledge Base Set*, (3) the *Knowledge Base Management System* and (4) the *Agent Manager*. We illustrate in details each of these modules and their function.

(1) *Interface.* It is the front-end of the client agent and provides the user with tools for visualizing choice lists (see Section 4.3), selecting agents to which ask for the cooperation and, consequently, performing the LKB update (by calling the integration procedure). Of course, in general, the interface must include also tools which depend on the considered application setting. Recall that, in our case, the chosen application setting is the information retrieval. As usual in front-end modules, it is possible to identifies two sub-modules,

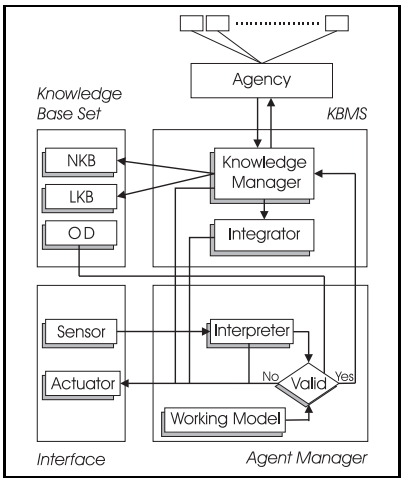

Figure 2: The Agent Architecture

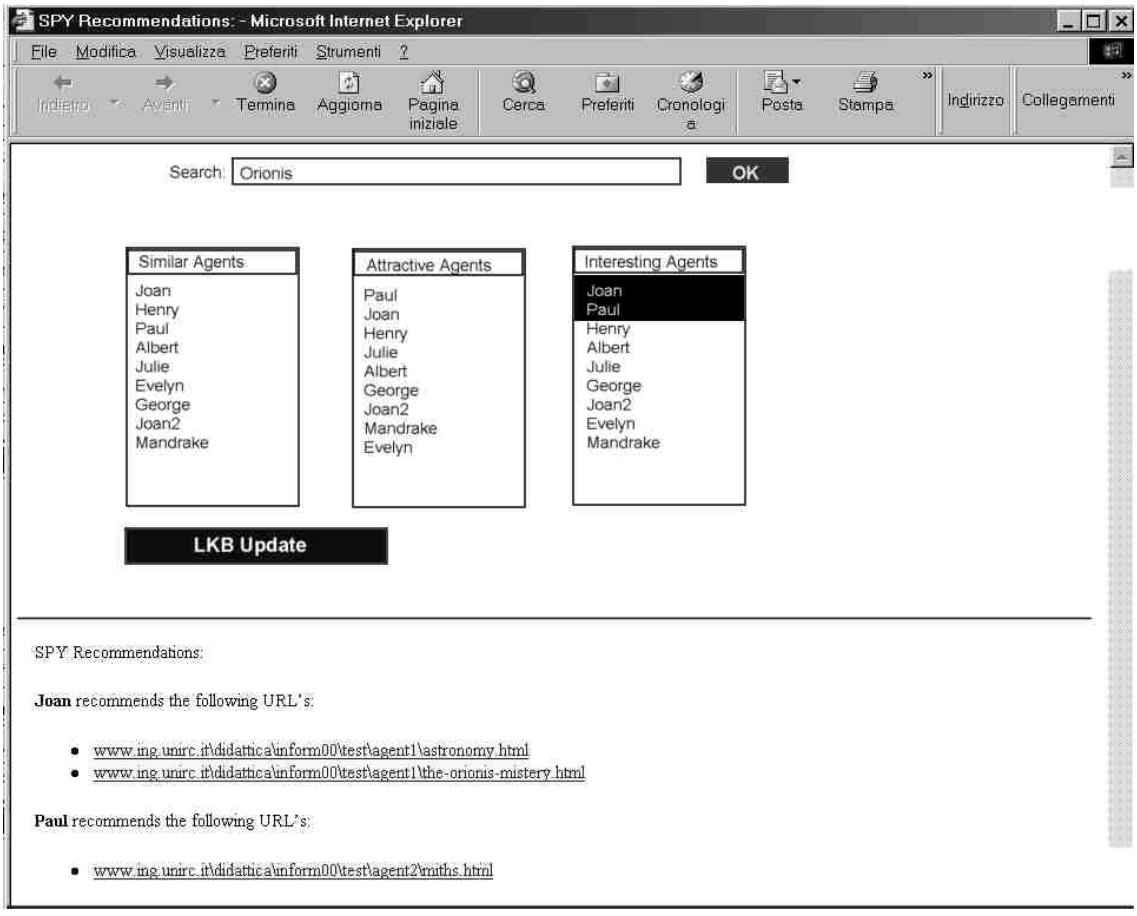

Figure 3: The Agent Interface Module

called *Sensor* and *Actuator*, the former including all the functions allowing the user to send request to the system, the latter collecting the functions managing the answer of the system to the user's request (see Figure 2). Figure 3 reports a snapshot of the interface at run time. The button OK, the search box, the tool for selecting agents in the three list boxes and the button *LKB update* belong to the sub-module *Sensor*: When the user clicks on the OK button, the system is advertised that she/he would like to contact the selected agents for submitting them the keywords specified in the search box. Furthermore, by clicking on the button LKB Update, the user may require the integration between her/his local knowledge base and the set of local knowledge bases of the contacted agents, in order to enrich its local knowledge with an external knowledge she/he considers useful. The box visualizing the three choice list and the box containing the set of recommendations provided by the system belong to the sub-module *Actuator*.

(2) *Knowledge Base Set.* It contains all the information needed for the agent for performing its task. Management and update of such an information are carried out by the module (3) (i.e., the *KBMS*) which is described below.

As shown in Figure 2, the Knowledge Base Set is composed of the following three sub-modules:

- the sub-module OD, that is the local copy of the Ontology Domain (recall that it represents the *a-priori* knowledge shared by all the agents of the net);
- the sub-module LKB, that is the Local Knowledge Base of the agent i (recall that it embeds the knowledge about how the user explores the Ontology Domain);
- the sub-module *Network Knowledge Base* (NKB, for short), which stores the list of agents of the community and the lists of similarity, attractiveness, and interest coefficients (S_{ij}, A_{ij}, I_{ij}, for $j \neq i$) currently determined between the agent i and all the other agents j of the net. Such coefficients, are computed by the Agency according to rules defined in Section 4 and transmitted to the client agent each time a change arises. Recall that the computation of such coefficient needs the knowledge of preference, benevolence and consent coefficients. These are kept by the Agency and automatically updated on the basis of the choice list exploitation and the cooperation grants/denials performed by the users (according to rules defined in Section 4). Observe that the information stored in the NKB is needed for providing the user with the three choice lists.

(3) *Knowledge Base Management System* (*KBMS*, for short). It is composed of two sub-modules, called *Knowledge Manager* (KM, for short) and *Integrator* (see Figure 2). The Knowledge Manager manages and updates data stored into the Knowledge Base Set (described above) and communicates with the Agency both for retrieving information about the other agents of the community (including the LKBs of agents chosen for cooperation) and for guaranteeing the consistency between local and remote (i.e., server side) data (coefficients, LKB, etc.). Moreover, the KM communicates with the module (4), i.e. the *Agent Manager* for receiving user's request and sending her/him answers (the functions of this component are next explained).

The other sub-module, that is the Integrator, is the component which perform the integration between LKBs of the cooperating agents, according to the procedure defined in Section 5.

(4) *Agent Manager.* This module accepts inputs received from the interface *Sensor* and from the Knowledge Manager (see above). This latter information is transmitted (after a suitable processing) to the interface *Actuator* in order to make it available to the user. The Agent Manager is composed of the following sub-modules:

- the *Interpreter*, which analyzes the Sensor data and, after having checked their correctness (by submitting a request to the sub-module *Validation Component* which we will next describe), sends a request to the $KBMS$; the request can be of two types: either a *searching request*, or an *update request*. The former concerns with the supported application, the latter corresponds to the user's request of integrating the LKB of her/his agent with the LKBs of the agents chosen for cooperation;
- the *Working Model Component*, which stores the list of the legal actions the user can carry out through the Interface. For instance, the Working Model Component could contain a rule of the type: *the user cannot choose more than 10 agents from the interesting agent list*.
- the *Validation Component*, whose task is deciding whether the request is consistent with both the Domain Ontology and the Working model of the agent. An example of failure of such a test is the case that the user submit just a keyword not corresponding to any concept appearing in the Ontology Domain. This task is implemented by accessing to both the Ontology Domain and the Working Model Component. If the request is valid, then it is sent to the $KBMS$; otherwise, a message of *illegal action* is sent to the interface *Actuator*.

7 Experiments

In this section we provide a practical demonstration of the system by tracing its execution during an experimental session. Moreover, we summarize some results obtained from the analysis of a number of experimental execution sessions.

Consider the case of the Figure 3. A user, named Frank, has selected two agents from the Interesting Agent list, namely Joan and Paul, and has specified the keyword *Orionis* in the Search box. After Frank clicks on the button OK, the *Sensor* transmits the user specifications to the Agent Manager. This module generates an updating request to the $KBMS$ (requiring the update of the preference and consent coefficients in the NKB), and then verifies that the specified keyword both corresponds to a concept of the Ontology Domain and satisfies the Working Model. Then, the Agent Manager generates a searching request for the $KBMS$. As a consequence, the Knowledge Manager of the $KBMS$ requires to the Agency the transmission of the LKBs of Joan and Paul. After this, the Knowledge Manager looks for the concept *Orionis* in the LKBs of Joan and Paul for the concept *Orionis*, and finds it in both of them. Starting from the concept *Orionis* in the LKB of Joan, the Knowledge Manager finds the leaf nodes accessed with a priority greater than a suitable threshold. In our case, such nodes correspond to the URLs `www.ing.unirc.it/didattica/inform00/test/agent1/astronomy.html` and `www.ing.unirc.it/didattica/inform00/test/agent/theorionismistery.html`.

Similarly, from the analysis of the Paul's LKB, the Knowledge Manager finds the URL `www.ing.unirc.i/didattica/inform00/test/agent2/miths.html`. The three URLs are then transmitted by the Knowledge Manager to the Actuator, for visualizing them as recommendations.

Now, Frank, after having received the recommendations, decides to click on `www.ing.unirc.it/didattica/inform00/test/agent1/astronomy.html`. Then, the Sensor notifies the Agent Manager, and this generates an request to the $KBMS$ of updating the priority coefficients of the LKB's nodes involved in the action performed by Frank (note that the action of accessing the node *Orionis* implies the updating of all the ancestor nodes of *Orionis* in the LKB of Frank).

At this point, Frank, being satisfied of the recommendations provided by Joan, decides to integrate his LKB with the Joan's one. In order to obtain such an integration, he clicks on the button *LKB update*. Then, the Sensor transmits a signal to the Agent Manager which sends an update request to the $KBMS$; as a consequence, the Knowledge Manager of the $KBMS$ exploits the Integrator component for carrying out the integration between the user's LKB and the Joan's LKB, and updates the Franks's LKB with the integrated LKB. Finally, the Knowledge Manager sends the updated LKB to the Agency.

From the analysis of behavior the system showed during several experiments, it results that the semantic properties, encoded by suitable coefficients, drive users on selecting from the agent net the most promising candidate agents for fruitful cooperation. User choices are exploited as feedback for adapting coefficients in such a way that a trade-off among similarity and attractiveness, on the one hand, agent congestion and user dissatisfaction, on the other hand, is obtained. To summarize, we report here three meaningful cases tested during experiments: (i) An agent a with high similarity and low attractiveness perceived by another agent b. The user of b decides to contact a less similar, but more attractive, agent c, and this means that the current similarity does not fully satisfy b. Since b has chosen c, as expected, it makes choices more similar to those of c than to those of a, and the similarity between a and b decreases, coherently with dissatisfaction of the user. (ii) An agent a with high interest and low similarity (or low attractiveness) perceived by another agent b. The user of b decides to contact a less interesting, but more similar (or more attractive) agent c. As a consequence, the interest for a perceived by b decreases, due to the decreasing of the consent coefficient C_{ba}. (iii) An agent a with high interest and high attractiveness perceived by another agent b. The user of b knows that high attractiveness means probably long waiting time for obtaining answers from a and decides to contact a less interesting agent c. As a consequence, the interest of b for a decreases.

8 Conclusion and Future Work

In this chapter a framework for representing and managing cooperation among agents in a Multi-Agent community is provided. The core of the proposal is the definition of a formal model based on several semantic properties and on a linear system involving some coefficients associated to such properties. The solution of such a system allows the user to find the best agents for cooperation in the net, that is those agents from which the most fruitful cooperation can be reasonably expected. Cooperation between two agents is implemented by the integration of the respective knowledge bases. The main direction for extending this work in our future research concerns the case of heterogeneous domains (in this chapter we assume that agents work in a common environment). In such a case, problems arising from the possible occurrence of semantic heterogeneities have to be faced. Another future work is of implementation type: beside enriching the current prototype, we plan to update the system for incorporating the extensions of the formal model.

References

[1] S. Abiteboul. Querying semi-structured data. In F.N. Afrati and P. Kolaitis, editors, *Proceedings of the 6th International Conference on Database Theory, ICDT 97, Delphi, Greece*, volume 1186 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 1997.

[2] S. Adali and R. Emery. A uniform framework for integrating knowledge in heterogeneous knowledge systems. In P. S. Yu and A. L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering, ICDE 95, Taipei, Taiwan*, pages 513–520. IEEE Computer Society, 1995.

[3] S. Adali and V.S. Subrahmanian. Amalgamating knowledge bases, iii: algorithms, data structures, and query processing. *Journal of Logic Programming*, 28(1):45–88, 1996.

[4] K. Arisha, S. Kraus, R. Ross F. Ozcan, and V. Subrahmanian. Impact: The interactive maryland platform for agents collaborating together. *IEEE Intelligent Systems Magazine*, 14(2):64–72, 1998.

[5] G. Boella and L. Lesmo. Norms and cooperation: Two sides of social rationality. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA*, pages 4–10. Morgan Kaufmann Publishers, 2001.

[6] M. Bowling and M. Veloso. Rational and convergent learning in stochastic games. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA*, pages 1021–1026. Morgan Kaufmann Publishers, 2001.

[7] M. Bowling and M. Veloso. A multiagent learning using a variable learning rate. *Artificial Intelligence*, to appear, 2002.

[8] C. Byrne and P. Edwards. Collaborating to refine knowledge. In D. Gordon, editor, *Proceedings of The Machine Learning '95 Workshop on "Agents that Learn from Other Agents", Tahoe City, California, USA*. electronically on WWW, 1995.

[9] A. Chavez, A. Moukas, and P. Maes. Challenger: A multi-agent system for distributed resource allocation. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents, Agents 97, Marina del Rey, California, USA*, pages 323–331. ACM Press, 1997.

[10] Y. S. Choi and S. I. Yoo:. Multi-agent Web information retrieval: Neural network based approach. In D. J. Hand, J. N. Kok, and M R. Berthold, editors, *Proceedings of the Third International Symposium Intelligent Data Analysis, IDA 99, Amsterdam, The Netherlands*, volume 1642 of *Lecture Notes in Computer Science*, pages 499–512. Springer, 1999.

[11] T. Dagaeff, F. Chantemargue, and B. Hirsbrunner. Emergence-based cooperation in a multi-agent system. In D.E. Hollnagel, editor, *Proceedings of the Second European Conference on Cognitive Science, ECCS 97, Manchester, U.K.*, pages 91–96. Univ. of Manchester Press, 1997.

[12] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review*, 12(3):309–314, 1997.

[13] M. Fisher, J. Muller, M. Schroeder, G. Staniford, and G. Wagner. Methodological foundations for agent-based systems. *The Knowledge Engineering Review*, 12(3):323–329, 1997.

[14] F. Gandon. Engineering an ontology for a multi-agents corporate memory system. In S. Tsuchiya and J.P. Barths, editors, *Proceedings of the Eighth International Symposium on the Management of Industrial and Corporate Knowledge, ISMICK 2001, Compigne, France*, pages 209–228, 2001.

[15] P. J. Gmytrasiewicz and E. H. Durfee. Rational coordination in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 3(4):319–350, 2000.

[16] N. Guarino, C. A. Welty, and C. Partridge. Towards ontology-based harmonization of Web content standards. In A. H. F. Laender, S. W. Liddle, and V. C. Storey, editors, *International Conference on Conceptual Modeling, ER Workshops, ER 2000, Salt Lake City, Utah, USA*, volume 1920 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2000.

[17] T. Helmy, S. Amamiya, and M. Amamiya. User's ontology-based autonomous interface agents. In N. Zhong, J. Liu, and J. Bradshaw, editors, *Proceedings of The Second International Conference on Intelligent Agent Technology, IAT 2001, Maebashi City, Japan*, pages 264–273. World Scientific, 2001.

[18] C. Iglesias, M. Garijo, J. Centeno-Gonzalez, and J. R. Velasco. Analysis and design of multiagent systems using MAS-common KADS. In M. P. Singh, A. S. Rao, and M. Wooldridge, editors, *Proceedings of Agent Theories, Architectures, and Languages, ATAL 97, Providence, Rhode Island, USA*, volume 1365 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 1997.

[19] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1-2):79–97, 1997.

[20] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In B. Hayes-Roth and R.E. Korf, editors, *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI 94, Seattle, Washington, USA*, volume 1, pages 444–450. AAAI Press, 1994.

[21] A. Moukas and P. Maes. Amalthaea: An evolving multi-agent information filtering and discovery system for the WWW. *Autonomous Agents and Multi-Agent Systems*, 1(1):59–88, 1998.

[22] M. Mundhe and S. Sen. Evolving agent societies that avoid social dilemmas. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.G. Beyer, editors, *Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2000, Las Vegas, Nevada, USA*, pages 809–816. Academic Press, 2000.

[23] M. H. Nodine, J. Fowler, and B. Perry. Active information gathering in infosleuth. *International Journal of Cooperative Information Systems*, 9(1-2):3–27, 2000.

[24] L. Palopoli, G. Terracina, and D. Ursino. A graph-based approach for extracting terminological properties of elements of xml documents. In *Proceedings of the 17th International Conference on Data Engineeringm, ICDE 2001, Heidelberg, Germany*, pages 330–337. IEEE Computer Society, 2001.

[25] L. Peshkin, K.E. Kim, N. Meuleau, and L.P. Kaelbling. Learning to cooperate via policy search. In K.B. Laskey and H. Prade, editors, *Proceedings of The Sixteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, California, USA*, pages 307–314. Morgan Kaufmann Publishers, 2000.

[26] M. V. Nagendra Prasad and Victor R. Lesser. Learning situation-specific coordination in cooperative multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 2(2):173–207, 1999.

[27] M. Schillo and P. Funk. Learning from and about other agents in terms of social metaphors. In J. M. Vidal and S. Sen, editors, *Proceedings of the "Agents Learning about, from and with other Agents" Workshop of the 16th Joint Conference on Artificial Intelligence, Stockholm, Sweden*. Morgan Kaufmann Publishers, 1999.

[28] S. Sen. Reciprocity: A foundational principle for promoting cooperative behavior among self-interested agents. In V. Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS 95, Menlo Park, California, USA*, pages 322–329. AAAI Press/MIT Press, 1995.

[29] S. Sen. Active information gathering in infosleuth. *Trends in Cognitive Science*, 1(9):334–339, 1997.

[30] S. Sen, A. Biswas, and S. Debnath. Believing others: Pros and cons. In *Proceedings of the Fourth International Conference on Multi-Agent Systems, ICMAS 2000, Boston, Massachusetts, USA*, pages 279–286. IEEE Computer Society, 2000.

[31] R. Sun. Individual action and collective function: from sociology to multi-agent learning. *Cognitive Systems Research*, 2(1):1–3, 2001.

[32] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In P.E. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning, ICML 93, Amherst, Massachusetts, USA*, pages 330–337. Morgan Kaufmann, 1993.

[33] A. Wang, C. Liu, and R. Conradi. A multi-agent architecture for cooperative software engineering. In Knowledge Systems Institute staff, editor, *Proceedings of The Eleventh International Conference on Software Engineering and Knowledge Engineering, SEKE 99, Kaiserslautern, Germany*, pages 1–22. Knowledge Systems Institute, 1999.

[34] G. Weiß. Adaptation and learning in multi-agent systems: Some remarks and a bibliography. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multi–Agent Systems*, volume 1042 of *Lecture Notes in Computer Science*, pages 1–21. Springer Verlag, 1996.

[35] M. Wooldridge and N. R. Jennings. The cooperative problem-solving process. *Journal of Logic and Computation*, 9(4):563–592, 1999.

[36] P. Xuan, V. R. Lesser, and S. Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments learning to cooperate via policy search. In J.P. Mller, editor, *Proceedings of The 5th International Conference on Autonomous Agents, Agents 01, Montreal, Canada*, pages 616–623. ACM Press, 2001.