Università degli Studi di Bologna
DEIS

# The SOcieties of ComputeeS project (a position paper)

P. Torroni P. Mello A. Ciampolini
E. Lamma M. Milano R. Montanari
F. Riguzzi A. Roli

May 23, 2002

*DEIS Technical Report no. DEIS-LIA-02-004* *LIA Series no. 57*

# The SOcieties of ComputeeS project (a position paper)

P. Torroni[1] P. Mello[1] A. Ciampolini[1] E. Lamma[2]
M. Milano[1] R. Montanari[1] F. Riguzzi[2] A. Roli[1]

[1] *DEIS, Università di Bologna*
*V.le Risorgimento 2*
*40136 Bologna, Italy*
{aciampolini, pmello, ptorroni}@deis.unibo.it
[2] *Dipartimento di Ingegneria, Università di Ferrara*
*Via Saragat, 1*
*44100 Ferrara, Italy*
elamma@ing.unife.it

May 23, 2002

**Abstract.** In the first year of the project, Bologna (UNIBO) will spend 14 months on WP1, *a logic-based model for computees*, and 10 months on WP2, *modelling interactions between computees*. Ferrara (DIFERRARA) will spend 12 months on WP1 and 8 months on WP2.

This technical draft is the result of a series of preliminary, informal meetings held in Bologna, during September and October 2001. It wants to help ourselves introduce our background and viewpoints on the SOCS project, and to sketch a tentative workplan, aimed at starting a discussion with the other groups involved in the project, about the first steps to do within it. Much space is dedicated to possible scenarios that we can use in the course of this project, as possible testbeds to check if our objectives are met.

## 1 Introduction

The SOCS Project will start in early 2002, with two workpackages: WP1, *a logic-based model for computees*, and WP2, *modelling interactions between computees*. In the first year of the project, Bologna (UNIBO) will spend 14 months on WP1 whose leader is Cyprus (UCY), and 10 months on WP2. UNIBO will be the leader of WP2. Ferrara (DIFERRARA) will spend 12 months on WP1 and 8 months on WP2. In this preliminary phase of the project, before we start with WP1 and WP2, we would like to draw a tentative research line, although keeping the scope of our future possible investigation as broad as possible.

This technical draft is the result of a series of preliminary, informal meetings held in Bologna, during September and October 2001, among the UNIBO and DIFERRARA people. It wants to help ourselves introduce our background and viewpoints on the SOCS project, and to sketch a tentative workplan, aimed at starting a discussion with the other groups involved in the project, about the first steps to do within it.

The UNIBO and DIFERRARA group involved in SOCS is composed at the moment by eight people. Before we start, we roughly summarize its research background and interests.

- abduction, dialogue, negotiation (Paolo Torroni, Anna Ciampolini and Paola Mello);
- constraints, planning (Michela Milano);
- learning, agents (Fabrizio Riguzzi, Evelina Lamma);
- metaheuristics, complex systems, emergent behavior (Andrea Roli);
- policy negotiation, mobile agents (Rebecca Montanari).

As stated in the project proposal accepted by the EU commission, the aim of the project is 'to investigate computational and logical models of individual and collective behaviour of computational entities – which we refer to as *computees* – interacting in the context of global and open computing environments'. With a focus on the concept of *computee*, we intend to:

- study several possible scenarios that could motivate our work and put it in a context, identifying several classes of problems
- study the advantages and drawbacks of a logic-based approach to such classes of problems
- do a bibliographic study on the state of the art with respect to the existing (both logic-based and non logic-based) solutions to such classes of problems

In fact, we would like to give a solid motivation to our logic-based approach, showing where and why it is preferable to other non-logic-based solutions: this is why we intend to explore both logic-based and non logic-based solutions of literature. On the other hand, at this stage, we will abstract away from the actual implementation of the computee, whether it is based on Prolog rules and reflects a logic execution model, or it is a java class, or whatever. Therefore, at this level, when we talk about *(non) logic-based solutions* we refer to the (high level) description and specification of the architecture.

## 1.1 Computational capabilities

The question that naturally arises is: *what computational capabilities should computees be provided with?* In particular, which balance between reactivity and rationality [KS96]. How could we design an architecture for the computee, such that it is both reactive to stimuli from outside and rational, in that it exhibits a goal-oriented behaviour? For example, in a dialogue context, there could be the need to answer to a partner's question in a timely fashion, or the opportunity to make an aside question, generated from background reasoning activity. This stage may require a study about the application fields that we want to investigate (but it is difficult to determine a methodology... any hint or suggestion?)

When we talk about rational entities, it is reasonable to think of the computees as being goal-driven. With this respect, we feel that it could be interesting to have the ability to express different degrees of satisfaction. In fact, while it could be natural to think, in logic, that a goal is either fulfilled or not, in many applications we could consider different degrees of satisfaction, which depend on the individual computees. How can this be reached?

An idea is that computees should be provided with a utility function to maximize, and/or a goal to achieve, or with both of them. The same concept applies to the societies of computees, when we may define a concept of 'global satisfaction'. A definition of "global frustration", that relates with this concept, is given in 3.3.

Other computational capabilities that must be taken into consideration include of course learning, adaptability, belief and theory revision, and planning, as we will see later.

## 1.2 Computees models and architecture

Another question is: *should such architecture be layered, or maybe shall computees be composed of several separated entities?* For instance, a computee could be composed of two different entities, one exhibiting a strongly reactive behaviour (that could be required for survival in a physical environment), one exhibiting a strongly rational behaviour (which could be used for instance for path planning). This option, anyway, could be relegated to lower level implementation choices. In fact, including both reactivity and rationality in a single entity could help focusing on a single kind of interactions, that among actual computees, i.e., on inter-agent interactions rather than intra-agent ones. A cycle à la Kowalski-Sadri could join both aspects in a single entity model.

Since the project is focused on interactions, possibly including dialogues of different kinds, it could be important for a computee to have a model of the other computees in the society. We must consider that the system we imagine is open to heterogeneous components that are free to join or leave it. Thus, an issue is: do other computees have goals? Which language do they understand? Do they have plans that can be modified? Do they argue? Can they be persuaded? Do they have learning capabilities? We imagine that, in some applications, computees can have to face other computees whose capabilities, goals, etc. are unknown, and adapt to the new situation. One more issue is, should we provide computees with the concept of group? (this can be useful for coalition-formation issues, or if we want to model the society in certain ways). We will come back to this issue later on.

Considerations of this kind may suggest that computees should be able, in some contexts, to learn the models of the other computees, in order to let them dynamic, modifiable, flexible.

## 1.3 Adaptation

Adaptation has been recognized to be a central point in our research. The problem is how to integrate it with the other aspects, and in particular with other reasoning mechanisms. When should learning intervene, should it be interleaved and interfere with them, what is the impact in terms of formal properties that hold in a computational environment that changes due to knowledge and theory updates and revision. Another problem is that it is not completely clear what it means for multiple computational entities to 'learn'. Whose knowledge or theory is updated? What does is mean that a *society* of computees actually learns? We will get back to this issue in Section 5.3.

## 1.4 Societies of computees

Emergent behaviour is both an interesting subject for research, and a possible threat for systems' stability in that it makes them unpredictable [Jen00]. When computees are put together in societies, we can assume that in certain situations they interact by means of protocols. Now, if we provide computees with suitable learning capabilities, and with a 'good' model (whatever it means) of the 'others', as such model is refined, for instance after a new acquaintance, we could imagine that the interaction protocols also change, hopefully becoming step by step more effective. Therefore, it would be interesting to use simulation techniques to discover emergent behaviours, and interaction protocols.

Of course, there could be an 'emergent behaviour' also in an environment where agents are not necessarily provided with learning features, but where if we look at the system as a whole, usually by means of a simulation, we can single out emergent properties and behaviour that had not been initially programmed, inside the computee. Which is not necessarily an advantage – this is why their study should be taken into account.

The study of emergent behaviour could also focus on 'positive' (as opposed to 'dangerous') properties: in fact, an interesting matter of research in this respect could be: how to intervene on the society of computees in order to provoke a certain behaviour. For instance: can we build computees that are particularly 'powerful' (whatever it means: skilled, or because of their knowledge, or because of some external norms), and then introduce them in the system when we want it to converge to a certain state?

The model of the other computees is only a part of the picture. In fact, the environment where computees act and move is not only composed of computational entities. Modeling the environment may require modeling sites, topologies, resources. What kind of resources do computees use? database records? URLs? physical items? plans? meta-knowledge? Which topology has the environment? If it is a network, how can a computee reach a remote site? How are sites described? Et cetera. It is important to take some decisions, possibly based on some concrete scenarios, otherwise the risk is to broaden up too much the spectrum until it becomes impossible to proceed.

## 1.5 Grouping Computees

So far, we have referred to societies of heterogeneous communicating computees. They are heterogeneous because they can have different goals, actions, architecture and domain knowledge. Under certain circumstances, they may share a protocol for understanding each other when communicating. However, the architecture of a society of computees (SoC) is another open issue. For instance, we could think of hierarchical organizations, flat 'anarchic' societies, or of SoCs as being constituted by groups of individuals. In particular, the computees that form a society may group to form *bunches* of agents that cooperate to achieve a common goal. Agents inside each group or bunch can be homogeneous if they all share the same actions and domain knowledge, or heterogeneous, when each agent is specialized in the execution of a particular task.

As we already said before, a computee should have the capability to model the other computees with whom it communicates: modeling computees in the same group is easier because the other computees' goals, actions and domain knowledge may be known from the beginning or can be asked to the subjects, while modeling computees outside the group is more difficult because the other computees' information must be hypothesized.

Computees in the same group cooperate to achieve a goal: each computee puts its resources at disposal of the group in exchange for nothing. Instead, a computee can put its resources at disposal of computees belonging to other groups only in exchange for something. In other words, a computee can commit to perform a task on behalf of a computee from another group only in exchange for a commitment by the other computee or for an amount of money, i.e. "currency" for buying services. Commitments are reached between computees belonging to different groups by means of negotiations that follow the law of demand and offer, i.e., computees belonging to different groups compete with each other. Instead computees belonging to the same group commit to perform tasks for one another in exchange for nothing, because each task is performed to achieve the same goal.

An example scenario where this architecture is natural is information gathering in some agent-based architectures: each human user owns a group of agents that he sends on the net in order to retrieve information about a particular subject. These agents cooperate with each other in order to find pieces of information and return it back to the user. The agents may interact with other agents sent by another user: they can perform tasks for the other user's agents only in exchange for other tasks or money (see 3.6).

The remainder of the paper is structured as follows: in Section 2 we will try and identify some elements that we consider fundamental in the SOCS framework, with respect to the abilities and knowledge representation of the individuals, given the considerations above. In Section 3 we will describe several possible scenarios for SOCS. In Section 4, we will give an overview of the state of the art in agent communication, from which we could take inspiration for the design of social aspects of SOCS. In Section 5 we will explain in more detail the ideas that the various groups would like to pursue in the project, at this first stage, and provide an essential bibliography to the specific topics.

# 2 Ingredients

This first overview suggests some directions we may adopt in choosing the 'ingredients' that can be included in the framework, and in particular in the computee abilities and knowledge representation.

- **rationality and reactivity**. We should consider computees that can exhibit both very rational and reasonably timely behaviour, depending on the external stimuli and internal goals and priorities. Computees must have the ability to represent and reason about goals, and therefore they must be provided with planning capabilities;

- **dialogue**, possibly of different kinds. We can start by considering the Walton & Krabbe's classification of dialogues [PSJ98, WK95];

- **learning**, for belief revision and theory revision, in order to make computees and SoCs adaptive. We can also think of a computee as a system with multiple components (e.g., one for the reactive part, one or more for the rational part), and capable to modify its own composition on the fly (e.g., giving more or less resources to a certain component?). This could be regarded as another type of 'meta-level' revision;

- **mobility**. In certain contexts it can be important to provide computees with the notion of site.

Other ingredients may be: **abduction**, to model incomplete knowledge, and to generate dialogues [STT02]; **constraints**, in a policy negotiation scenario (as we will see later), to define a subset of common policies; a model of **noise**, if we want to give a more applicable representation of the computees' communicative acts; the concept of **evolution**, if we want to describe how computees and societies evolve in time, and how properties and protocols emerge; **indirect communication**, as computees can overhear [ABDS01], or spy, other dialogues, **norms**, and **argumentation**, which could be used to reason with norms, and include them in the negotiation process. In the future, it would be also interesting to study the emergence of norms, in addition to protocols.

# 3 Scenarios

SOCS is a basic research project, therefore in principle we would not need a 'killer application' to motivate our work. Nonetheless, it is still important to keep it ground to concrete problems, to use them as a testbed, and to put our work into a practical context.

By 'scenario' we do not mean a specific application, but rather an *abstraction of a possible set of applications*, where computees and societies of computees could be used. Before we get to defining such abstraction, following a bottom-up approach, we introduce several somewhat detailed independent scenarios, largely derived from existing problems of literature.

## 3.1 Policy Negotiation and Mobility Decisions

A first scenario is derived from distributed systems and policy negotiation. In that case, computees will model mobile software agents that enter a system. Several interesting problems could arise from this setting. We outline two of them in the following: the first relates to SOCS in that it focuses on societies, the latter in that it deals with mobility.

### 3.1.1 Policy Negotiation

Mobility of software and hardware components is an enabling technology for the design of wide area pervasive applications, but it introduces new challenges in the critical aspect of dynamic control of component behaviour. The dynamic determination of potential new execution contexts and the possibility to move and perform computation in new environments involves to cross multiple administrative authorities.

In this context a primary requirement is to dynamically establish and control component behaviour in a way that does not conflict with the administrative and security policies of the entering domain. Let us assume the following scenario. The behaviour of each mobile component is controlled by means of policies defining component duties (obligation policies) and component rights needed to perform application tasks (authorization policies). Each execution domain is governed by a set of policies that rule the interactions between entering components and domain resources. In addition, each execution domain can concurrently host the execution of multiple components, and any of them subjected to their own defined policies.

In dynamic and wide scale environments mobile components cannot have an *a priori* knowledge of the policies of the entering domain and of the co-located components. This scenario requires to address two issues:

- dynamic discovering of policies;
- dynamic negotiation of policies to enable to match domain and component duties and rights.

The result of the negotiation protocol is to find a set of policies that do not conflict and that can be used to govern the behaviour of entering components. This means that mobile components dynamically adapt their initial set of rights and duties and acquire the new rights and duties that are produced by the negotiation process.

As an application example, we can think of the dynamic service composition in web environments, which is a typical business-to-business scenario, and the problem is: how to compose some services and get to an agreement

### 3.1.2 Mobility Decisions

Co-location can change the results of computation. But the question is: when is it convenient to move component execution near to the resources needed for computation or near to the components with which interactions are in progress? If it is convenient to move, how does the mobility affect and alter the activity plan? Where should components move (routing problems, also related to security, fault tolerance, etc.) and, *what* will actually move. In particular, in a logic based component, which is composed of a knowledge base and a reasoning part, possibly built on the top of a virtual machine, it is not trivial to decide whether the 'code' to move is the reasoning mechanism, the knowledge base, or the two bits altogether.

## 3.2 Automated negotiation: auctions

Auctions are a typical application of multi-agent systems. We think that they could be an important scenario to take into consideration in the SOCS. In fact, in recent years, the auction scenarios themselves got more and more complicated, and are likely to be a good benchmark for societies of computational entities. Some problems include: coalition formation, dealing with multiple auctions at the same time, participating to auctions with the purpose to achieve multiple resources (a package); an interesting web site is the Trading Agent Competition (TAC) home page of the University of Michigan: `http://auction2.eecs.umich.edu/`. Related links could be found in [Bom01] by Magnus Boman, and in [Pre01] by Chris Preist of HP Labs, Bristol.

An application which is particularly interesting for the project is that described by Collins and Gini in [CG01]. The problem is that of bid evaluation for coordinated tasks in a population of self-interested independent agents. We have to perform a job that involves a set of coordinated tasks within a limited time window. The resource needed to perform these tasks must be acquired from self-interested suppliers. Each supplier provides some tasks, with associated time windows, costs and risk. The aim is to select suppliers that cover all tasks to be performed, ensuring temporal and feasibility constraints, at the minimum cost and the minimum risk.

The optimization side of the problem can be decomposed in a scheduling problem and a set covering application effectively handled by a constraint programming tool. The constraint solver could be enhanced by Operations Research techniques in order to improve solver performances, as shown for example in [FLM01]. It has been proved, in fact, that many optimization problems can benefit from the use of complementary techniques like the two under exam.

The aim of the agent optimization component is to find the optimal solution. However, often time requirements are strict and a sub-optimal (resp. partial) solution is provided. In this setting we can apply metaheuristic techniques for improving (resp. completing) the solution. Metaheuristics [CDG99, Stü99, VMOR99] are approximate algorithms which combine constructive methods, local search strategies, and local optima escaping strategies. In our application, we plan to experiment different approaches for the integration of metaheuristics and constraint programming methods. The integration, in fact, will concern global constraints, search and optimization.

In our application, metaheuristics are useful since some problem hypotheses could change at execution time. For instance, when suppliers can provide services with some delay w.r.t. the expected time. Expected resources could be no longer available or deadlines could become tighter.

In this application, also negotiation, mobility (along with security issues) and learning can be exploited. In fact, the bidding process involves a negotiation phase where a customer asks for a service (or a job) and suppliers should build and provide bids, possibly asking other suppliers. Suppliers also have a goal, i.e., maximize the number of resources under control and the profit. Thus, the customer should first decide the time windows within which tasks should be executed. The larger the windows, the lowest the price of the task and the higher the risk that the schedule cannot be built.

In this setting, mobility can improve the efficiency, fault tolerance and reliability of the interactions between costumers and suppliers. Finally, the customer on the basis of experience can associate a risk to each supplier and change this value after each commercial transaction.

### 3.3 Landscape Models (and beyond)

The dynamic behavior emerging from the agents' interactions can be effectively studied by means of simulations on energy (or fitness) landscapes. Depending on the agents protocols and strategies, different dynamics can emerge and qualitatively different stationary states (final configurations) can be reached.

The following discussion is inspired by the article "A Landscape Theory of Aggregation" [AB93] included in a Robert Axelrod's book on the complexity of cooperation among agents [Axe97].

The landscape theory can model several problems, like coalition formation, task distribution and information flows balance. Some examples will be presented below. Moreover, some extensions of the model lead to the representation of complex scenarios with noise, dynamic environments and different kinds of relation between single and global utility.

In the following we will assume a scenario with $n$ agents, each pursuing a goal which is the minimization of a *frustration* value (or, conversely, the maximization of a *utility value*). The landscape model is defined as follows:

- Each agent $i$ is associated to a *size* $s_i > 0$ which quantifies its importance, power, priority, etc. When agents are not differentiated, $s_i = 1\ \forall i = 1, \ldots, n$.
- For each pair of agents $i, j$ a *propensity* $p_{ij} \in \mathbb{R}$ is defined. The propensity evaluates the tendency of the two agents to collaborate, to share resources, to move to the same node, etc. The propensity is positive if the agents can increase their utility if they collaborate, and it is negative in the other case. Often, propensity is symmetric, i.e., $p_{ij} = p_{ji}$; for instance, in the case of services providers which aggregates. Nevertheless, there are situations characterized by asymmetric propensity, for example where a resource is given by an agent to another one.
- For each pair of agents $i, j$ a *distance* $d_{ij} > 0$ is introduced. $d_{ij}$ may represent the number of links a message from agent $A_i$ needs to pass to reach agent $A_j$, or the level of trust between two agents, or the number of layers separating two jobs in a hierarchical organizational structure. Moreover, by assuming a binary distance $d_{ij} \in \{0, 1\}$, the value 0 can represent the situation where two

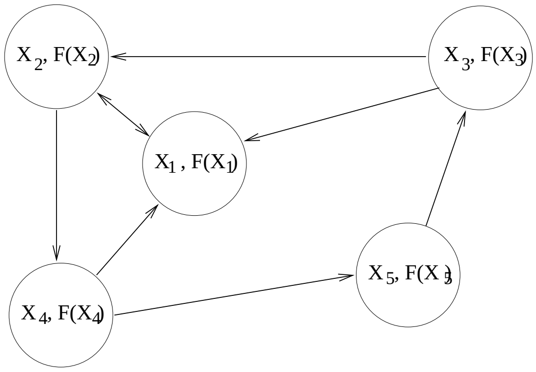


Figure 1. *A graph representing a landscape.*

agents are in the same coalition, and 1 the other case. Since in general the distance can change depending on the state the agents are in, we introduce the state of the system $X$ and define $d$ as a function of $X$: $d_{ij} = d_{ij}(X)$.

- A *frustration* function is associated to each agent:

$$f_i(X) = \sum_{j \neq i} s_j p_{ij} d_{ij}(X).$$

  The agent tries to minimize its frustration by using a strategy that guides the agent's actions which thus modify the state $X$ of the whole system. The agent's strategy may also involve other agents and the protocol adopted can be critical for the agent to reach its minimal frustration.

- A *global frustration* is introduced:

$$F(X) = \sum_i s_i f_i(X) = \sum_{i,j,i \neq j} s_i s_j p_{ij} d_{ij}(X).$$

  This global function measures the system's welfare, as it is an averaged sum of the individual frustration.

The mapping $X \to F(X)$ is the *landscape*, which is usually represented as a labeled graph (see Figure 1), where nodes represent the states of the system, arcs represent the possible transitions from a state to another and labels are the values of $F$. A pictorial idea of a landscape is given in Figure 2, where each point of the 2$D$ surface represents the value of $F$ of the corresponding state.

The states we usually want the system to reach are the *global minima* of the global frustration function $F$. In these states, all agents have their minimal frustration. Local minima of $F$ corresponds to states of the system where some agent has a

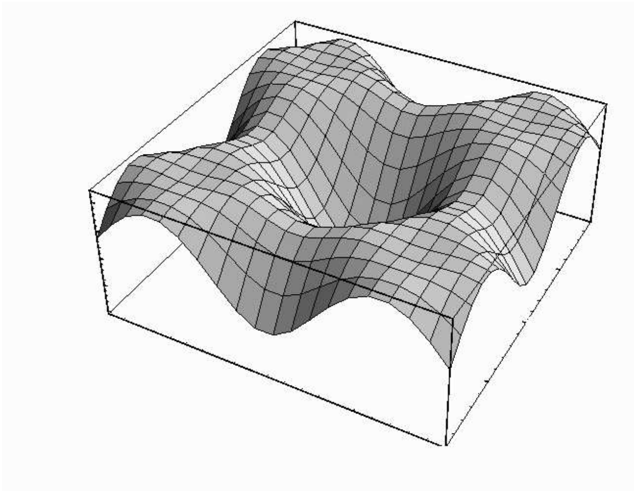

Figure 2. *A pictorial landscape representation.*

frustration higher than the one it could reach. In this model the agents behavior corresponds to a trajectory on the landscape and this dynamics is the result of protocols and strategies of agents. For example, we suppose that the protocol states that only one agent at a time can act and the strategy (for simplicity the same for every agent) guides the agent to change its state in the direction of the maximal decrease of $f$. This combination protocol/strategy is nothing else than a gradient descent in a discrete surface. Agents start at a configuration $X$ and, action by action, will eventually reach a steady state which is a minimum of $F$. But depending on the initial state, this can be either a global or a local minimum. If the protocol allowed two or more than two agents to act at the same time, the final state could be different. Similarly, a more complex strategy would lead to different dynamics and consequent final states.

The landscape model can be used as a testbed for the dynamics emerging by the protocols and strategies that the agents follow. Given an instantiation of the model (thus referring to a particular real case) we can analyze the landscape and extract useful properties (number of local minima, size of basins of attraction of global minima, auto–correlation, etc.) and observe the resulting dynamics. Most of the knowledge and results available about optimization on fitness landscapes can be directly applied [Kau93, Kau95, Sta95, KM95].

The following questions can be answered:

- Given the strategies, does the protocol produce a dynamics such that the agents reach always/often/never a global minimum?

- Dually, given the protocol, can the strategies lead the agents to reach their goal?
- In case of heterogeneous strategies, are there dominant strategies (i.e., agents reach their minimum at the expense of other agents' success)? Are altruistic strategies needed to lead the system to a global minimum?
- Are protocols and strategies robust to noise and unforeseen events?
- How protocols and/or strategies should be modified as to let the system reach a global minimum?
- Are there "dangerous" initial configurations that lead the trajectory to non–desirable final states? In other terms, what are the basins of attraction to avoid?

Moreover, other issue can be addressed:

- What are the final states that the system can reach if agents are divided in partitions and the agents of each partition (coalition) pursue only the benefit of the partition they belong to?

  This question is strongly related to the studies of Kauffman about adaptive multi-agent systems [Kau95, KM95] and it opens the door for the introduction of computational agents in the domain of social agents simulation and optimization.
- With the given definition, each action which moves an agent in a better (i.e., less frustrated) state, leads the whole system to a better configuration. However, this is no more valid assuming asymmetric propensity $p$, thus extending the possible scenarios and dynamics to investigate. In particular, a main point is the investigation of the relation between single and global benefit.

  Moreover, $p_{ij}$ can be a function of time: $p_{ij} = p_{ij}(t)$, for example when an agent $A_i$ has no more interest in collaborating with agent $A_j$ as soon it has obtained from $A_j$ the resource it needed. The introduction of the dependence from time generates dynamic landscapes, which changes in time.
- The global function can be different from the one given in the definition. For instance, the function $F(X) = min\{f_i|i = 1, \ldots, n\}$ is the leading force to a monarchy with subjects trying to accomplish and pursuing the queen benefit (see for example the model of some social bees). Conversely, the function $F(X) = max\{f_i|i = 1, \ldots, n\}$ leads the system to the support and help of the "most frustrated" (or less fit) agent (the strongest manifestation of cooperation). Furthermore, the *size* $s_i$ of agents can be a function of time and then generating a scenario where the relative importance of agents varies.

Several problems can be represented by the landscape model. In the following, we briefly mention few of the most typical.

### 3.3.1 Coalition formation: the case of service providers

The system is composed of $n$ agents, each providing a set of services $\{\sigma_1, \sigma_2, \ldots\}$. Depending on the customers requests, a set of services has to be offered and agents must form coalitions in order to provide such composition of services. Each agent has high propensity to join agents which provide services it does not provide by itself. Conversely, it can have null or even negative propensity to join an agent which has no lacking services to offer. The distance between two agent is 0 if they belong to the same coalition and 1 if they belong to different coalitions. Assuming an initial coalitions structure, agents can change coalition by obeying protocols and following their strategy, with the goal to minimize their frustration, i.e., maximizing the number of services needed to fit the customer's requests.

### 3.3.2 Choosing sides: the emergence of a norm

We suppose to have a set of agents and each agent can accept or refuse a norm (e.g., to refuse services to a group of customers, to move to another node, to join a malicious coalition in an auction, etc.). Depending on the interests (profit, utility, frustration) and power (priority, importance, resources availability) it is possible to evaluate respectively propensities $p_{ij}$ and sizes $s_i$. Distances are evaluated as in the example above. Once defined protocols and strategies it is possible to know the final acceptance/refuse of a norm as steady state of the dynamic. This example also fit the case of high order protocols emergence.

### 3.3.3 Cooperation: cooperative problem solving through a network

Finally, we consider a case where distances are measured as number of links a message has to pass to reach the receiver agent from the sender. Agents must reach a goal (for example completing a planning task) and they share resources, information and sub-goals. Propensities are measured as the utility that agents can achieve by sharing information and exchanging sub-goals. This example is particularly meaningful in the analysis of the protocols and strategies the agents follow, because frustrated situations indicate that some agents can not reach their goal and protocols and/or strategies must be changed. Moreover, given the intrinsic dynamic character of the model, when it is not possible to change protocols and strategies, system designers can move the system to a better (hopefully the best) situation by introducing agents with the appropriate strategies (for example, altruistic agents or agents whose goal is to help negotiation).

The sketched examples point out that negotiation assumes in the landscape model a very important role, as it provides advanced strategies to lead the system to the maximal overall utility.

Landscape models are related to coalition formation and game theory, but they are not aimed at generating coalition structures, neither at finding dominant strategies. They represent a flexible and general testbed for the analysis of dynamic properties of multiagent systems.

## 3.4 The Predator/Prey ("Pursuit") domain

The Predator/Prey or Pursuit domain (pursuit domain in the following was introduced in [BJD86] and it is used in [SV00] to illustrate the various features of Multi-Agent Systems (MAS). Even if it is a toy domain, it can be used for testing the basic features of a Society of Computees (SoC). In this domain there is a number of predators and a prey that move on a 2-dimensional world (see figure 1a). The predators are usually four and their aim is to capture the prey or surround it so that it can not move to a free position. A capture position is shown in figure 1b.

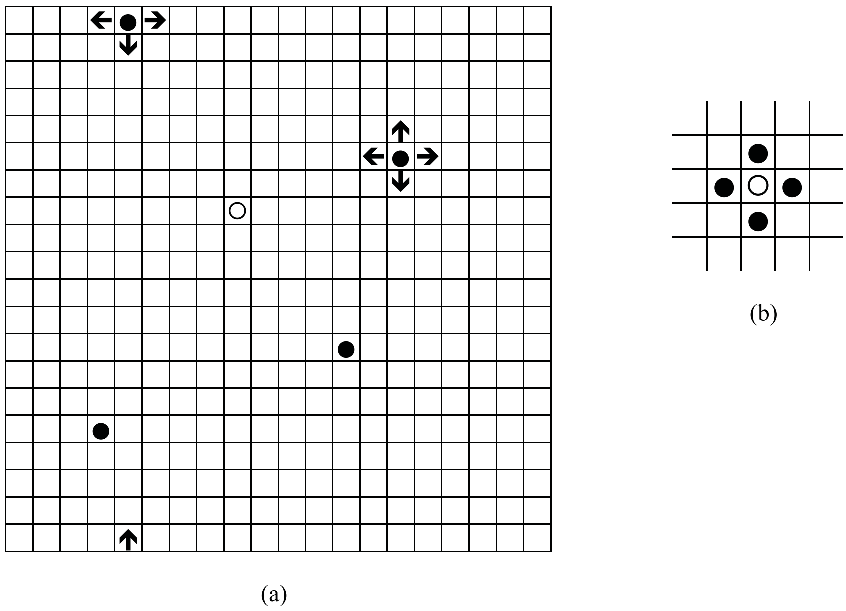


Figure 3. *An instantiation of the pursuit domain.*

The predators and the prey can move horizontally and vertically and they move in turns. The world is toroidal, i.e., a player moving off one end of the board comes

| | |
|---|---|
| Definition of capture | Visible objects and range |
| Size and shape of the world | Predator communication |
| Predators' legal moves | Prey movement |
| Simultaneous or sequential movement | |

Table 1. *Parameter of the pursuit domain.*

back on the other end. Usually the prey moves randomly with a certain probability of standing still in order to model the fact that it is slower than the predators. The predators have a complete view of the world and they can communicate with each other. Variations of this domain can be obtained by varying a number of parameters. The world can be non-toroidal, it can have a grid with other shapes as spaces (for instance hexagons) or it can be continuous. The dimension of the world is a parameter of the domain and it can also be infinite. The players can move not only horizontally and vertically but also diagonally. In the case where the world is not toroidal, the prey can be captured by less than four predators by trapping it against an edge or a corner. Another possible criterion for capture is that a predator occupies the same position as the prey. The prey can move in order to escape the predators instead of moving randomly. The player can move simultaneously instead of in turns. The predators can have only a limited view of the world and the may not be able to communicate. A summary of the parameter in the pursuit domain is shown in table 1.

### 3.5 Robotic Soccer

Robotic soccer was first presented in [Mac93]. Robotic soccer can be played either with real robots or with a simulator. We will concentrate here on the simulation version. A particularly good simulator is the "soccer server" developed by Noda [Nod98]. This simulator is realistic in many ways: the players' vision is limited; the player can communicate by posting messages to a blackboard that is visible to all players; all players are controlled by separate processors; each team has 11 players; each player has a limited stamina; actions and sensors are noisy and play occurs in real time. A key aspect of robotic soccer is that the players not only need to control themselves but also the ball, which is a passive part of the environment, and their opponents, which are active parts of the environment. It can be used to evaluate different MAS techniques in a direct manner: teams implemented with different techniques can play against each other. The server provides support for users wishing to build their own agents. A number of teams is available on the Internet.

### 3.6 Information Gathering

The amount of information currently available via the Internet and the World Wide Web has grown so much that automated methods for gathering information on behalf of the user have become needed. Information gathering is an activity involving pro-active acquisition of information, from possibly heterogeneous sources, in response to a complex query that may require the system to possess capabilities like reasoning, representation and inferencing. In [ONL94] the authors propose the application of MAS to the task of information gathering: in response to a query, multiple semi-autonomous agents can be released to search the distributed "information space" in a cooperative manner for relevant items. Cooperation between agents involves the management of inter-dependencies between their activities so as to integrate and evolve consistent clusters of high quality information from distributed heterogeneous sources.

### 3.7 More scenarios

In addition to such more practical or commercial scenarios, there are some interesting problems that are derived from game theory, such as those related to norms, cooperation, the prisoner dilemma, coping with noise, choosing sides, landscapes and frustration, dissemination of culture, etc.

More applications can be found following the links from the UMBC Agents web page, maintained by Tim Finin: `http://agents.umbc.edu`; they mostly refer to the e-commerce context.

## 4 Communication in Multi-agent Systems: state of the art

In [SV00] a survey on the state of the art in MAS is presented. The authors divide the systems in four classes: homogeneous non-communicating agents, heterogeneous non-communicating agents, homogeneous communicating agents and heterogeneous communicating agents. We will discuss in the following the last two classes since they are more relevant to the project. Homogeneous agents share the same goals, actions and domain knowledge. They differ only in the values returned by their input sensors in a given situation. Heterogeneous agents instead can have different goals, actions and domain knowledge.

### 4.1 Homogeneous communicating agents

In the pursuit domain described in 3.4, [SM90] defined a strategy for an homogeneous non-communicating MAS. The strategy consists in a simple heuristic behaviour for each agent based on local information. They define capture positions as the four positions adjacent to the prey. Each agent moves to the capture position that is closest to itself. Since the agents can not possibly see each other, they can move to the same position and block each other as they approach, thus failing to capture the prey. In the case of homogeneous communicating agents, Stephens and Merx propose another strategy for the predators [SM90] where they communicate to ensure that they aim at different capture positions: the predator that is farthest from the prey (how does it knows it is the farthest?) chooses the capture position closest to it and announces that it will approach that position. Then the next farthest predator chooses its capture position and announces it to the others, and so on. In this way the agents try to approach the prey from different sides. However, this protocol does not always succeed.

When considering communicating agents, the issue of *what* they should communicate arise. The agents can communicate their sensed state of the world. However, the agents can also exchange information regarding their internal states. Balch and Arkin [BA94] study the effect of allowing agents to communicate their states and their goals with one another in three multi-robot applications. They found that agents that communicate goal information performed slightly better than those that communicated state information. Both conditions exhibited far superior behaviour when compared with non-communicating agents. It also been observed that allowing agents to communicate internal state information can be very effective in the robotic soccer domain [SV99].

Examples of applications of homogeneous communicating agents are in the domain of distributed sensing. Matsuyama [Mat90] describes a cooperative distributed vision project a that aims at understanding a three dimensional scene using multiple cameras, either stationary or on mobile robots. An example application of the system is the problem of tracking an individual car using cameras mounted at urban intersections: when the car leaves one camera's range and enter another's, there needs to be a way of identifying the two images as representing the same car, even though it probably looks different in the two cases. The project combines active sensing, the ability to shift attention towards an area of higher uncertainty or interest, and communication among multiple sensing agents. Another distributed sensing project is the *trafficopter* [MM97] where car themselves collect and propagate traffic information to help each other decide on the best route to a given location.

## 4.2 Heterogeneous communicating agents

When considering heterogeneous communicating agents a number of issues and applications have been investigated. We present some of them taken from [SV00].

*Understanding each other.* When agent are heterogeneous, they must agree on a common protocol. Independent aspects of the protocol are information content, message format and coordination conventions. Among other, existing protocols for these three levels are KIF for content [GF92], KQML for message format [FMFM94] and COOL for coordination [Mat95].

*Planning Communicative Acts.* Communication actions might alter the beliefs of other agents and therefore modify the external world. As a consequence, their execution is planned in the same way other actions are. The theory of communication as action is called *speech act* [CL95, LS95].

*Benevolence vs Competitiveness.* A society of agent can have a common goal, can be composed of self-interested agents or can be composed of groups of agents sharing the same goal. Agents belonging to different groups compete in order to achieve their respective goal. Agents in the same group can cooperate or compete to achieve their goal. An example of competitive agents striving to perform a common task is [Wei95] where each agent is a Q-learner that tries to earn the right to control a single system. The highest bidder pays a certain amount to be allowed to act, then receives any reward that results from the actions. An example of cooperative Q-learners is [Clo96] where one agent teaches another through communication: a learner is rewarded for mimicking the trainer.

A form of competition is *negotiation* or competition based on the law of supply and demand. For example in *contract nets* [Smi80] each agent has its own self-interested goal and have limited reasoning resources. They bid to accept tasks from other agents and then they can either perform the task (if they have the proper resources) or sub-contract them to other agents. When sub-contracting, they must pay for having the task executed and therefore shop around for the lowest bidder.

*Planning* Agents that must achieve a common goal have to plan in coordination. An example of coordinate planning is Partial Global Planning (PGP) [DL95] where heterogeneous agents can post constraints, or commitments to do a task by some time, to each other local scheduler.

*Commitment* Agents in a society, whether sharing a common goal or self-interested, may decide to commit to each other, i.e., agree to pursue a given goal, possibly in a given manner, regardless of how much it serves's one own interests. [Had95]. The theory of belief-desire-intention (BDI) can help agents to reach commitments to each other: each agent models other agents by modeling their domain knowledge (beliefs), their goals (desire) and their intentions, i.e., the goals they are currently trying to achieve and the methods by which they are trying to achieve them. In [RG95] BDI is used in order to build a system for air-traffic control.

# 5 Group activity and background

The scenarios presented above gave a flavor of the directions of research that we can undertake. In fact, they showed the need for some particular features (ability to learn, to negotiate, and so on) that SoCs must take into account if we want them to tackle some specific problems and application domains, giving a motivation to the work, and representing a possible reference for making future design choices.

We would like now to explain in more detail the ideas that the various groups of UNIBO and DIFERRARA would like to pursue in the project, at this first stage, and provide an essential bibliography to the specific topics. Most of the cited papers are available from the web; a subset of them can be downloaded from `http://lia.deis.unibo.it/ Research/ Projects/ SOCS/ partners/ papers/`

## 5.1 Abduction, dialogue and agent negotiation

**Paolo Torroni, Anna Ciampolini and Paola Mello**

The idea is to study a logic-programming based approach to agent dialogue. Some relevant previous work by Sadri, Toni, and Torroni [STT01, STT02] aims at producing dialogues from the agents' knowledge bases (including their goals), by means of an abductive proof procedure. The important point in solving this problem with logics is that it is possible to formally prove properties of the system, and that it is possible to obtain an implementation, although prototypical and inefficient, by giving high-level, declarative rules and constraints (those used to prove the properties).

Such work is pretty much focused on negotiation and resource allocation issues. But dialogues can also be of other types: persuasion, inquiry, information seeking, deliberation. It would be very interesting to study how and why agents could have dialogues of different types. A good and short classification of dialogues, from [WK95], is in [PSJ98], Section 5.3 ('Other views of dialogue'). It is worth reading Section 6 of the same paper ('Related work'), for more references. [STT02] also contains a good discussion about dialogue, but again, it is especially focused on negotiation issues. An important argumentation-based approach to agent dialogue is in [KSE98], but it is not straightforward to translate an argumentation system into an operative implementation.

The agent (computee) architecture: a right balance between reactivity and rationality has also been widely discussed, from a Logic Programming perspective. 4 different approaches from 1992 are briefly discussed in [BDM+99], Section 1.3.1; the paper is also very interesting to understand how to combine LP and MAS in general. It makes use of an extension of Linear Logic Programming: $\mathcal{E}_{hhf}$, providing multi-conclusion clauses. On our opinion, the two references that we should focus on, to start with, are (1) that one cited above, and (2) the Kowalski-Sadri agent cycle, whose last version is very clearly depicted in [KS99].

Our impression is that it is very important to build an architecture that can both exhibit a reactive behaviour, and that can be implemented starting from a high-level declarative language, grounded on a solid semantics basis, that allows proving properties of the system. Those approaches that do not take into account the practical side of the problem are not easily motivated (much more expressive logic formalisms exist, such as modalities, which in most cases lack an operational semantics).

### 5.2 Constraints and planning

**Michela Milano**

Constraints [Hen89] can be very useful both for modelling and solving optimization and feasibility problems. Concerning the modelling side, constraints are relations which can be declaratively imposed among objects, i.e., entities of the application domains. In addition, constraints have an operational semantics, i.e., a propagation algorithm that performs domain filtering by removing those combinations of assignments that cannot lead to any consistent solution. The more powerful the filtering algorithm, the higher the reduction of the search tree and the efficiency of the solver. After propagation, the remaining part of the search tree should be explored. In general, constraint solvers perform a complete search.

An important modelling tool supported by all constraint programming systems are the so called global constraints [BC94], i.e., relations among several variables representing suitable abstractions that are subproblems of many applications. In this way, real life problems can be modelled by composing subproblems represented in terms of global constraints. Beside the modelling facilities, global constraints embed powerful propagation algorithms that remove a considerable portion of the search space. Constraints have been successfully used to solve scheduling and planning applications.

Quite recently, constraints have been used in multi-agents applications. The reference publication for agents and constraints is the book by Nareyek [Nar01], that looks at computer games as an application field. The author claims that 'The agents must be able to exhibit sophisticated behavior and find solutions even for unforeseen situations. A planning system is therefore needed to guide an agent's behavior. The system will be fully based on the constraint programming paradigm. This allows the agent's behavior to be represented by a declarative high-level specification that can be easily modified and maintained. The application of high-level constraints allows domain-specific knowledge to be exploited for the search process.'

The link between agents and constraint programming will be further explored during the project. An application which is particularly interesting for the project is that sketched in 3.2, taken from [CG01].

### 5.3 Learning and agents

**Fabrizio Riguzzi, Evelina Lamma**

A computee should be able to modify its knowledge base in order to adapt to the environment.

*Belief revision* is the simplest form of knowledge base updating. This process is started when the computee acquires new knowledge (typically in the form of facts about the world) that is contrasting with the knowledge it already has. In this case, the computee should modify its knowledge base in order to restore consistency. In particular, in belief revision the consistency is restored by modifying the truth value of a set of beliefs stored in the knowledge base in the form of facts. The predicates whose facts can be revised are called revisables. A number of procedures for performing belief revision have been proposed in the literature, among them we cite REVISE [DPS97] and GBR [LRP01].

A more complex type of knowledge updating consists in modifying the clauses in the computee knowledge base. This type of updating is called *learning* because general knowledge is modified in this case. Learning is usually performed by starting from a set of positive and a set of negative examples for one or more predicates whose definition is to be learned or revised. Two main type of learning tasks of this kind exist: batch learning and incremental learning. In batch learning all the examples are given at then beginning of the computation while in incremental learning the examples are given one at a time. In batch learning, a set of clauses is found so that, together with the available knowledge base, they cover all the positive examples and none of the negative ones. Examples of batch learning systems are Progol [Mug95] and Foil [Qui90] In incremental learning, every time a new example is obtained, it is tested against the available knowledge and operations are performed depending on the result of the test: if the example is positive and it is covered by the theory or if it is negative and is not covered, then nothing is done; if the example is positive and it is not covered, then the theory is generalized, while if the example is negative and it is covered then the theory is specialized. Examples of incremental systems are INTHELEX [ELMS96] and RUTH [DR92].

A computee should be able to update its knowledge base both by means of belief revision and by means of learning. A possibility to integrate the two would be the following: when a new fact is agent that is in contradiction with the available knowledge, the computee first tries to change the truth value of revisable literals in order to eliminate contradiction. It it fails, then it can try to update its general knowledge by means of learning.

In the context of computees, it is very important to define what is the role that is to be assigned to knowledge updating. A possibility would be to have a separate knowledge updating component or to have knowledge updating interleaved with planning and acting, the other capabilities of the computee. This second approach is probably to be preferred: a fraction of the computee time should be dedicated to learning so that it is interleaved with the other main functions of the computee, namely planning and acting. The amount of time devoted to learning can be fixed beforehand or, better, can be dynamically modified by the computee: in situations where there is no immediate action to be taken, the computee may prefer to devote its time to updating the knowledge base with the evidential knowledge gathered up to that point, while in more dynamic situation the agent may decide to postpone learning and just collect information without integrating it in the knowledge base.

Batch and incremental learning require examples, i.e. input/output pairs of the function to be learned. This kind of learning is called *supervised learning*. This method is appropriate when there is a teacher that supplies the correct input/output pairs or when the function's output represents a prediction about the future that can be checked by looking at percepts in the next time step. However, when no teacher is available and when the effect of action can be verified only at later times, different techniques have been developed that go under the name of *reinforcement learning*. In reinforcement learning no examples for concepts are provided but only a delayed feedback is available in response to the actions of the agent. This feedback is a real number that expresses the reward obtained by the agent in its course of actions. One of the simplest form of reinforcement learning is Q-learning where the aim is to learn an action-value function that provides the expected utility (in the form of a real number) of taking each action in each state. This function is called Q-function. In [DRB98] an approach is presented for learning a Q-function in the form of a first-order regression tree.

In the context of computees, it is our opinion that reinforcement learning option should be discarded, since it results in planning future actions, while we would like planning to be a central point in our study, independently of the learning techniques adopted.

In [SV00] a survey of machine learning techniques for multi-agent systems is presented. MAS are divided into 4 classes: homogeneous non-communicating agents, homogeneous communicating agents, heterogeneous non-communicating agents and heterogeneous communicating agents.

## 5.4 Metaheuristics, complex systems and emergent behaviour

**Andrea Roli**

The idea is to adopt the agent metaphor for building a multi-level framework for metaheuristic algorithms, and put together and possibly coordinate different algorithms [RM01]. The expected result of such approach is both in the direction to obtain a better understanding of metaheuristics, and to suggest directions for SE-oriented implementation. Moreover, this framework permits to directly connect metaheuristics as multiagent systems and complex systems. The application of models and metaphors of complex systems to the analysis and synthesis of multiagent systems is a quite new and promising research direction. The focal point is the system dynamics and its properties, like the possible presence of emergent behaviors.

The contribution to the project of such approach is multiple. In fact, the layered architecture could be a possible scheme for certain societies of computees. Computees could be characterized by having different capabilities (metaheuristic algorithms), and it could be interesting to study the way interactions among such kind of entities should be modeled. Once identified a fitness landscape for a society of computees, it could be interesting to study what it means 'fitness' for a single computee, for a society of computee, in which way the benefit of the single would impact on the so-called *social welfare*. Such kind of problems has been deeply investigated in the multi-agent literature, but we think that the modeling of the welfare function as a fitness landscape to be explored by a society of computees could lead to new interesting results.

The computees paradigms, on the other hand, could add value to the above cited work, in that the higher levels of meta-reasoning about algorithms have not yet been explored. With reference to the planning issues introduced in 5.2, planning computees could manage the meta-level management of computees incorporating metaheuristic algorithms, and with reference to 5.3, learning computees could improve such meta-level in terms of adaptation.

# References

[AB93] R. Axelrod and D.S. Bennett. A landscape theory of aggregation. *British Journal of Political Science*, 23, 1993.

[ABDS01] M. Aiello, P. Busetta, A. Donà, and L. Serafini. Ontological overhearing. In *Proceedings ATAL'01, Seattle, WA*, number VII in LNAI, Intelligent Agents. Springer Verlag, 2001.

[Axe97] R. Axelrod. *The Complexity of Cooperation*. Princeton studies in complexity. Princeton University Press, 1997.

[BA94] T. Balch and R. C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):1–25, 1994.

[BC94] N. Beldiceanu and E. Contejean. Introducing global constraints in CHIP. *Mathematical Computer Modelling*, 20(12):97–123, 1994.

[BDM+99] M. Bozzano, G. Delzanno, M. Martelli, V. Mascardi, and F. Zini. Logic programming and multi-agent system: A synergic combination for applications and semantics, 1999.

[BJD86] M. Benda, V. Jagannathan, and R. Dodhiawala. On optimal cooperation of knowledge sources - an empirical investigation. Technical Report BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computer Services, 1986.

[Bom01] M. Boman. Trading agents. *AgentLink News Newsletter*, 6:15, January 2001.

[CDG99] D. Corne, M. Dorigo, and F. Glover, editors. *New Ideas in Optimization*. Advanced topics in computer science series. McGraw-Hill, 1999.

[CG01] J. Collins and M. Gini. An integer programming formulation of the bid evaluation problem for coordinated tasks. *Non so dove e' pubblicato*, 2001.

[CL95] P. R. Cohen and H. J. Levesque. Communicative actions for artificial agents. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 65–72, Menlo Park, California, June 1995. AAAI Press.

[Clo96] J. A. Clouse. Learning from an automated training agent. In G. Weiss and S. Sen, editors, *Adaptation and Learning in Multiagent Systems*. Springer Verlag, 1996.

[DL95] K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80, Menlo Park, California, June 1995. AAAI Press.

[DPS97] C. V. Damásio, L. M. Pereira, and M. Schroeder. REVISE: Logic programming and diagnosis. In *Proceedings of Logic-Programming and Non-Monotonic Reasoning, LPNMR'97*, volume 1265 of *LNAI*, Germany, 1997. Springer-Verlag.

[DR92] L. De Raedt. *Interactive Theory Revision: An Inductive Logic Programming Approach*. Academic Press, 1992.

[DRB98] S. Dzeroski, L. De Raedt, and H. Blockeel. Relational reinforcement learning. In *ICML98*, 1998.

[ELMS96] F. Esposito, A. Laterza, D. Malerba, and G. Semeraro. Refinement of Datalog programs. In *Proceedings of the MLnet Familiarization Workshop on Data Mining with Inductive Logic Programing*, pages 73–94, 1996.

[FLM01] F. Focacci, A. Lodi, and M. Milano. Optimization-oriented global constraints. *Constraints*, 2001. to appear.

[FMFM94] T. Finin, D. McKay, R. Fritzson, and R. McEntire. KQML: An information and knowledge exchange protocol. In K. Fuchi and T. Yokoi, editors, *Knowledge Building and Knowledge Sharing*. Ohmsha and IOS Press, 1994.

[GF92] M. R. Genesereth and R. E. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-02-1, Computer Science Department, Stanford University, 1992.

[Had95] A. Haddadi. Towards a pragmatic of interaction. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 133–139, Menlo Park, California, June 1995. AAAI Press.

[Hen89] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming*. MIT Press, 1989.

[Jen00] N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.

[Kau93] S.A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[Kau95] S.A. Kauffman. *At home in the universe*. Oxford University Press, 1995.

[KM95] S.A. Kauffman and W. Macready. Technological evolution and adaptive organizations. *Complexity*, 26(2):26–43, March 1995.

[KS96] R. A. Kowalski and F. Sadri. Towards a unified agent architecture that combines rationality with reactivity. In D. Pedreschi and C. Zaniolo, editors, *Proc. International Workshop on Logic in Databases, San Miniato, Italy*, number 1154 in LNAI. Springer Verlag, 1996.

[KS99] R. A. Kowalski and F. Sadri. From logic programming towards multi-agent systems. *Annals of Mathematics and Artificial Intelligence*, 25(3/4):391–419, 1999.

[KSE98] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation; a logical model and implementation. *Artificial Intelligence*, 104:1–69, 1998.

[LRP01] Evelina Lamma, Fabrizio Riguzzi, and Luís Moniz Pereira. Belief revision by lamarckian evolution. In *First European Workshop on Evolutionary Learning (EvoLEARN2001)*, LNCS. Springer-Verlag, 2001.

[LS95] A. Lux and D. Steiner. Understanding cooperation: An agent's perspective. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 261–268, Menlo Park, California, June 1995. AAAI Press.

[Mac93] A. K. Mackworth. On seeing robots. In A. Basu and X. Li, editors, *Computer Vision: Systems, Theory and Applications*, pages 1–13. World Scientific Press, 1993.

[Mat90] T. Matsuyama. Cooperative distributed vision. In *Proceedings of the First International Workshop on Cooperative Distributed Vision*, Bandera, Texas, 1990.

[Mat95] T. Matsuyama. COOL: A language for describing coordination in multi agent systems. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 17–24, Menlo Park, California, June 1995. AAAI Press.

[MM97] A. Moukas and P. Maes. Trafficopter: A distributed collection system for traffic information. At URL `http://trafficopter.www.media.mit.edu/projects/trafficopter/`, 1997.

[Mug95] S. Muggleton. Inverting entailment and Progol. In *Machine Intelligence*, volume 14, pages 133–188. Oxford University Press, 1995.

[Nar01] A. Nareyek. *Constraint-Based Agents*, volume 2062 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[Nod98] I. Noda. Team GAMMA: Agent programming on GAEA. In H. Kitano, editor, *RoboCup-97: Robot Soccer WOrld CUp I*, pages 500–507. Springer Verlag, 1998.

[ONL94] T. Oates, M. V. Nagendra, and V. Lesser. Cooperative information gathering: A distributed problem solving approach. Technical Report 94-66, Department of Computer Science, University of Massachussets, 1994.

[Pre01] C. Preist. Agent Mediated Electronic Commerce at HP Labs, Bristol. *AgentLink News Newsletter*, 7:3–6, July 2001.

[PSJ98] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[Qui90] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[RG95] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 312–319, Menlo Park, California, June 1995. AAAI Press.

[RM01] A. Roli and M. Milano. Metaheuristics: A multiagent perspective. Technical Report DEIS-LIA-01-006, University of Bologna (Italy), 2001. LIA Series no. 50.

[SM90] L. M. Stephens and M. B. Merx. The effect of agent control strategy on the performance of a DAI pursuit problem. In *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence*, Bandera, Texas, 1990.

[Smi80] R. G. Smith. The contract net protocol: High-level communication and conterol in a distributed problem solver. *IEEE Transaction on Computers*, C-29(12):1104–1113, December 1980.

[Sta95] Peter F. Stadler. Towards a theory of landscapes. In R. Lopéz-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertuche, editors, *Complex Systems and Binary Networks*, volume 461 of *Lecture Notes in Physics*, pages 77–163, Berlin, New York, 1995. Springer Verlag. SFI preprint 95-03-030.

[STT01] F. Sadri, F. Toni, and P. Torroni. Logic agents, dialogues and negotiation: an abductive approach. In *Proceedings AISB'01 Convention, York, UK*, March 2001.

[STT02] F. Sadri, F. Toni, and P. Torroni. Dialogues for negotiation: agent varieties and dialogue sequences. In *Proceedings ATAL'01, Seattle, WA*, volume to appear of *LNAI, Intelligent Agents VIII*. Springer Verlag, 2002.

[Stü99] T. Stützle. *Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications*. DISKI - Dissertationen zur Künstliken Intelligenz. infix, 1999.

[SV99] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwith communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

[SV00] Peter Stone and Manuela M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[VMOR99] S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors. *Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, 1999.

[Wei95] G. Weiss. Distributed reinforcement learning. *Robotics and Autonomous Systems*, 15:135–142, 1995.

[WK95] D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press, Albany, NY, 1995.