UNIVERSIDAD POLITÉCNICA DE VALENCIA

Depto. Sistemas Informáticos y Computación

MASTER THESIS

Social Based Adaptation in Multi-agent Systems

Author: Juan Ángel García-Pardo Giménez de los Galanes
Supervisor: Dr. Carlos Carrascosa Casamayor

Grupo de Tecnología Informática - Inteligencia Artificial
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia, Spain
Noviembre, 2010

Contents

1

Introduction and Motivation

A dynamic environment whose behavior may change over time presents a challenge that agents located there will have to solve. Changes in an environment —e.g. a market— can be quite drastic: from modifying the dependencies of some product to adding new actions to build new products. The agents working in this environment would have to be ready to embrace this changes to improve their performance, which otherwise would be diminished.

Also, they should try to cooperate or compete against others, when appropriated, to reach their goals faster than in an individual fashion, showing an always desirable emergent behavior.

When it is referred to *adapting* or *adaptation*, it is in the sense of "changing towards a better performance". The evolution of an agent to adapt consists in two processes: actually try some new behaviors (or part of them); and adjust these new behaviors aiming for a better performance. When it is said that the society adapts which is relevant is that the objects which are adapting are the individuals inside the society. This "bottom–up" approach of the sense of *innovation and adaptation* is desirable from the point of view of multi–agent systems: it is totally decentralized, less prone to malfunctions when individual fail, and —of course— it models some problems where the independence of the decision making is crucial.

The society should try to innovate when there are important changes which prevent the agents to feel comfortable. In the opposite case, when agents feel comfortable, the society should try to keep that behavior. Keeping in mind that there is no real object such as “society” between the agents, the need of creating one appears. As said before, it is preferable to do it implicitly; this way the *society concept* would emerge through the interactions between the agents.

First we must have a plausible model for an agent. We will base it in the human psychology, which allows to express its opinion and also to hear the opinion of other agents which are inside the social network of this agent.

The goal of this model is to allow agents to *propagate* the sense of comfort (discomfort) using the opinion, so other agents can feel comfortable (uncomfortable) if the majority of their *contacts* feel this way, despite their own feelings. Some other emotions will come into play, for example *fear*, which in this work will be —later— treated as the uncertainty of the variability of the outcomes of the agent’s work.

Through this view the global opinion can also be understood as a measure of the average happiness of the society. This average can be computed also in smaller communities —these communities can be *clusters* of agents which share something, e.g. same goals—, providing information about which communities are more affected by the changes in the behavior.

A different view of the approach is the one that explains it as a mimic of omnipresence: an agent cannot be in every part of the environment, but it would be desirable for it to achieve it; that way it would realize when a change occurs, even though it is not located in the state where the change is *visible*. If the agent in question *trusts* other agents, these agents can communicate the changes they see to it, achieving *omnipresence* if there are enough agents to cover the states of the environment.

The way of communicating this information can be —as in past works— explicit: there is a communication protocol between the agents, and they use it to *talk* between them. Also the trust issues have to be solved taking into account that not everything an agent says is necessarily true.

Another way to communicate changes is by giving the opinion about how the agent is dealing with the environment. For a stationary environment this opinion should —after some time— represent that the agent is comfortable with the situation, since we expect the agent to adapt easily. After a change, those agents which were affected by the new conditions may feel *uncomfortable*, and their opinion should reflect just that. The trust issues are still here, but with some remarks: the opinion of another agent can be trustfully, but its interests may be not the same ones as this agent’s. Of course, the opposite holds also true: the other agent may have the same interests as this agent, but is trying to deceive through its opinion —so it seems that the agent is not comfortable when it actually is, and vice versa)—. There is no way, without at least observing how well the other agent is really doing, to distinguish between the two cases. There is no need to do so either: the opinion allow this agent to compute the *affinity* to that other agent: if they both share a similar opinion through time, their interests either are the same, or there is some amount of deception in the opinion sharing. Either way, from the point of view of this agent, there is no difference: if they both feel uncomfortable, they should change the way they are doing things. They should keep their behaviors when they feel comfortable.

In this work a method which guides adaptation through social interaction is proposed. The proposal aims to show that adaptation is performed without explicitly reporting by a central authority that changes have occurred. The proposal shows that the system can adapt without the explicit recognition of the changes which should trigger an adaptation, which follows the idea before mentioned: society should change its behavior only when there are significant changes, and not otherwise.

The proposal is shown with an algorithm based on reinforcement learning, a well known technique of learning, and through the interactions of the components of the multi agent system. Much work can be done with the reported opinion of each agent. The opinion represents the happiness of the agent. The reported opinion can be deceptive, but as said before, it will not matter in terms of *similarity* calculus. Through this opinion clusters of similar agents can be computed, which would represent different societies.

This project has been developed inside the GTI-IA[1] (Group of Technology in Informatics – Artificial Intelligence) group which belongs to the DSIC department, Technical University of Valencia.

This work is framed inside a project called *OVAMAH* (Virtual Adaptive Organizations: Architectures and Development Methods, *TIN2009-13839-C03-01*). The ideas supporting the project —and the work done here— started in two other past projects: *THOMAS* (Methods, Techniques and Tools for Open Multi-Agent Systems, *TIN2006-14630-C03-01*) and *I2TM* (Intelligent and Integral Management of Customer Support Centers Exploitation, *FIT-34001-2004-0011*).

[1]www.gti-ia.dsic.upv.es

2

Related Work

Contents

2.1 Introduction to the Related Work

In this chapter several tools and different information is reviewed for the sake of completeness. These *utilities* are the bases on which the main work is supported.

As mentioned in the introduction, the goal of having an emergent way of organization is —partially— the aim of this work. The multi–agent system has to be able to communicate, somehow, the opinion that each individual agent has. In the case the environment changes, the organization that may be inside the multi–agent system may *decide* that there is the need for a change in the way they were carrying out their tasks.

Through the social information the agents will react to these social changes and adapt to the new conditions of the environment, provided that the agents will be able to differentiate *friend of foe* agents. This information will be useful also to make this differentiation possible.

In section 2.2 a short review of what is a complex adaptive system is given. It will be shown that every adaptive system can also be seen as a multi–agent system, capable of learning and adapting.

In section 2.3 a brief introduction to multi–agent systems is given. There the notion of agent is defined, as well as the ideas that are behind the multi–agent paradigm. This section is of key importance due to the nature of the work: the system will allow any kind of agent in the environment, since the system does not control it, but the system will only *work* with those agents which are capable of social communication. Also is to note that every agent is situated in an environment, which can also be read as "every agent could have its own environmental representation", which is of key importance in other fields as ontology alignment. In this work the representation of the environment will be treated with more detail in 3.

The differentiation between learning and adapting is explored in section 2.4. When talking about evolutionary systems, for example, two different —at least two— phases can be identified: the innovation in the system which has evolved, and the adaptation that the environment will evaluate in an automated fashion. These details are explored very superficially there.

When talking about innovative behaviors and adaptation, some work done in psychology is useful. Innovation, adaptation and a measure of them both are introduced in section 2.5. These ideas are moved to the field of multi–agent systems further on by defining a model of agent including these ideas in 4.

Also background of two different —but very closely related— techniques is provided in sections 2.6 and 2.7. These techniques are in the core of both the proposed model and the learning algorithm for adaptation.

Lastly some existing in the literature approaches to multi–agent learning for changing environments, emotional models and social communication inspired learning algorithms are very briefly reviewed in section 2.8.

2.2 Adaptive Systems

An adaptive system is a set of interacting or interdependent entities, real or abstract, forming an integrated whole that together are able to respond to environmental changes or changes in the interacting parts.

Following this definition the entities or agents must form a "whole" which will respond to environmental changes or changes in the agents themselves. This whole, which is the system, does not need to be explicitly specified by, e.g., a set of rules or a topology. When the design of the system is made from the point of view of the agents located there we can talk about a bottom-up designed system.

Another definition, very similar in spirit to the former one, is given to the term *complex adaptive systems*. According to John Henry Holland [Waldrop, 1992]:

> A Complex Adaptive System (CAS) is a dynamic network of many agents (which may represent cells, species, individuals, firms, nations) acting in parallel, constantly acting and reacting to what the other agents are doing. The control of a CAS tends to be highly dispersed and decentralized. If there is to be any coherent behavior in the system, it has to arise from competition and cooperation among the agents themselves. The overall behavior of the system is the result of a huge number of decisions made every moment by many individual agents

The definition of a complex adaptive system is somewhat more restrictive than the definition of a classical multi agent system, where the control of the system or the global behavior does not need to be an emergent effect of the interactions between the agents. In a multi-agent system the global behavior can be planned ahead, dividing certain aspects of the system to different kinds of agents, in a goal directed fashion [Russell and Norvig, 2009]. This is also the view used in a wide range of fields such as economics, where the agents' behavior should be rational in the sense of *rational choice theory* [Allingham, 2002], such as they pursue their goals by the means they have available.

Traditional examples of problems which are appropriate to multi-agent systems research include online trading, disaster response, and modeling social structures. Although every CAS can be seen as a multi-agent system, the traditional examples for complex adaptive systems are focused on the adaptive part of the system: stock market, ant colonies, ecosystem, immune system, manufacturing businesses and social systems. As seen in the examples, the idea of emergence is the key of those systems. For instance, in a social system simulation agents are modeled according to certain design criteria. The result of the interaction of the individuals is highly unpredictable beforehand due to the chaotic nature of the system [Holland, 1999]. Small variations on the interactions between the individuals or in the initial conditions can develop in a totally different future of the simulated system.

When the adaptive system can self adjust some of its parameters (given that it has some), the value of the parameters depends only on the history of the system dynamics. This way the system can implicitly learn which values for these parameters are better suited to which circumstances. These vague concepts will be stated more precisely later, when the difference between learning and adapting is well established.

2.3 Multi-Agent Systems

Quoting from [Ralston et al., 1993]:

> Multi-agent systems are computational systems in which several artificial "agents", which are programs, interact or work together over a communications network to perform some set of tasks jointly or to satisfy some set of goals. These systems may consist of homogeneous or heterogeneous agents. Examples of agents would be ones for detecting and diagnosing network problems occurring on a segment of a local area network; for scheduling the activities of a group of machines in a workcell on a factory floor; or for locating agents that are selling a specific product and deciding on what price to pay. Agents may be characterized by whether they are benevolent (cooperative) or self-interested. Cooperative agents work toward achieving a set of shared goals, whereas self-interested agents have distinct goals but may still interact to further their own goals. For example, in a manufacturing setting, where agents are responsible for scheduling different aspects of the manufacturing process, agents in the same manufacturing company would behave in a cooperative way, while agents representing two separate companies, where one company was outsourcing part of its manufacturing process to the other company, would behave in a self-interested way. Agents often need to be semi-autonomous and highly adaptive due to their "open" operating environments, where the configuration and capabilities of other agents and network resources change dynamically. Agent autonomy relates to an agent's ability to make its own

> decisions about what activities to do, when to do them, and to whom information should be communicated. Scientific research and practice in this area, which is also called Distributed Artificial Intelligence (DAI), focuses on the development of computational principles and models for constructing, describing, and analyzing the patterns of interaction and coordination in both large and small agent societies.

A shorter, convenient definition is also given by [Shoham and Leyton-Brown, 2008]:

> A multiagent system is a system that include multiple autonomous entities with either diverging information or diverging interests, or both.

Multiagent Systems (MAS) are a general software paradigm focused on fundamental questions about autonomy, cooperation, coalition formation, etc. Currently they are being applied in a wide range of domains, with significant results. Every MAS can be classified into two different categories: open MAS and closed ones. The main difference between them lays in the fact that a closed MAS is created with a fixed structure and goals, where an open MAS allows agents to freely enter and leave the system dynamically, and they don't necessarily share goals, or not the whole of them. This type, open MAS, is of special interest to this work.

Open MAS exist in dynamic operating environments, where new components are aggregated, existing components leave the system, or both of the former ones. Operating conditions can change also dynamically, in a unpredictable manner. These open MAS have as their characteristics the heterogeneity of their participants, the limited trust between them and very probably a not negligible subset of conflicting goals.

Agent Oriented paradigm, and MAS have become in one of he most important techniques applied in the resolution of complex problems, belonging to the artificial intelligence field. This is so, that it is considered as *the next most significant advance in software development* [Sargent, 1992] and *the new revolution in software* [Ovum, 1994]. Several applications using this paradigm exist and are being used in a wide variety of areas [Jennings and Wooldridge, 1998], such as:

- *Industrial applications*, dealing themselves with:
  - *Process Control*: applied to power transmission (northern Spain), particle accelerator control, nuclear plant monitoring and diagnosing of failures and also in the steel coiling process.

  - *Production*: successfully applied in systems dealing with part assembly, painting, product storage, etc...
  - *Air traffic control*: applications deployed in airports like Sydney, Australia, for air traffic management.

- Also being applied in *commercial applications* to:
  - *Information Management*: such as intelligent email filtering, news groups or automatic retrieval of information on the web.
  - *Electronic commerce*: for tasks such as automatic product purchase or sale and product search taking into account user preferences.

- *Medical applications*:
  - *Intensive Care Unit patients monitoring*: used to control the condition of the patients in the ICU.
  - *Patient assistance*: systems developed to follow the treatment of patients, taking into account every relevant aspect of their diseases or conditions.

- Lastly, also used in *entertainment*, such as:
  - *Games*: the agents technology allows more complex games, with intelligent features in the characters which can autonomously coexist in the virtual environment with their human controlled counterparts.
  - *Interactive theater*: a user can play the role of a character, while the other characters in the play are virtual, controlled by agents.

2.3.1 Agent Definition

Several proposals to define what an agent can be found in the literature; none of them is fully accepted by the whole scientific community as being the *definitive definition*. One of the simplest ones is the one from Russel [Russell and Norvig, 1995], which conceives an agent as an entity which senses the environment and acts on it.

Grounding in this definition, agents can be characterized accordingly to their attributes [Botti et al., 1999], which will define their behavior to solve a given problem. Thus we can speak of a social agent, an adaptive agent, ...depending on their attributes.

On the other hand, some other definitions can be found which restrict the image of the first definition, requiring an agent to fulfill some of those attributes in their basic definition.

Franklin [Franklin and Graesser, 1996] includes the autonomy attribute inside the basic definition for an agent. This way the definition formalizes an agent as a system situated inside and being part of an environment, which is able to sense and act on this environment through time, pursuing its own agenda to, this way, act on what it will sense in the future.

Another similar definition is given by Huhns [Huhns and Singh, 1998], where agents are active and persistent components which are able to sense, reason, act and communicate.

A definition which tries to reconcile all these differences, based on a set of attributes is provided by H. Van Parunak [Van Parunak and Odell, 1999]. An agent is defined as an evolution, an increment, of an active object which may or may not have a set of additional attributes —as many and of the kind as needed—. This way he compares an agent to a swiss army knife, where the basic definition is the knife itself, and if it is needed there can be added accessories to it. If not required, there is no need to carry all of the accessories (figure 2.1).

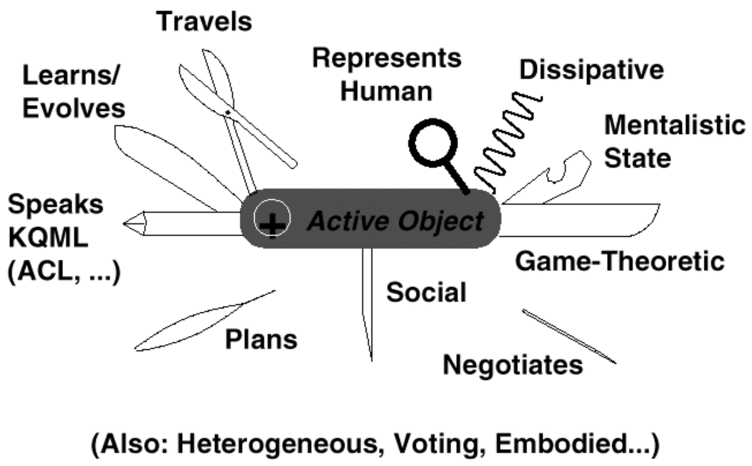


Figure 2.1: An agent's accessories (Parunak)

Finally, the most accepted definition nowadays is the one proposed in [Wooldridge and Jennings, 1995], whereby an agent is a computational system capable of autonomous action, flexible, in an environment, categorizing flexible as:

- *Reactive*, which responds to the environment in which the agent is situated.
- *Proactive*, capable of pursuing its own plans or goals.
- *Social*, capable of communicating with other agents, using some kind of language and communication channel.

Attending to this idea, to name a system "agent" it has to fulfill the former requirements. Currently only a small percentage of the existing software accomplishes this definition.

2.3.2 Agent's Attributes

Some of the properties or attributes which are granted to agents —at different degrees— to solve particular problems, and which are described by some works like [Franklin and Graesser, 1996, Nwana, 1996], are:

- *Temporal Continuity*: an agent is considered as a never ending, continuously executing process, always carrying out its own function.
- *Autonomy*: according to [Castelfranchi, 1995] an agent is autonomous if it can operate without direct human intervention, and has some kind of control over its own actions and internal state.
- *Sociability*: this characteristic allows an agent to communicate with other agents, or even with some other entities.
- *Rational*: the agent is able to reason about the sensed data from the environment, and compute the optimal outcome. No other outcome should be better than the one computed by the agent, given the information it has —the sensed data—.
- *Reactivity*: an agent acts because of changes on the environment it is situated. In this case, the agent senses the environment and the changes in it conduct the agent's behavior.
- *Proactivity*: an agent is proactive when it is able to pursue its goals, despite the changes that may occur in the environment. This definition is not in conflict with *reactivity*. The behavior of the agent is the result of two different types of behaviors: the *receptive* behavior and the *discoverer* behavior. While following a receptive behavior the agent is guided by the environment. On the opposite, in a discoverer

behavior the agent adds up internal processes to obtain its own goals —or subgoals—. The final behavior of the agent is in between being totally receptive or being totally discoverer; it has some degree of both behaviors (more about that on this work, in 2.5 and 4).

- *Adaptivity*: related to the learning tasks an agent is able to carry out, and related with the ability of an agent to change its behavior according to what it has learned.
- *Mobility*: ability of an agent to relocate itself using a computer network.
- *Veracity*: assumption in which an agent does not give false information on purpose, nor tries to deceive another agent or entity.
- *Benevolence*: assumption in which an agent is willing to *help* other agents if the help is not in conflict with its own goals. This attribute does not prevent the *rationality*: acting to help other agents can have zero reward for the agent; if it is benevolent, it will act to help.

Even though there is no consensus about the degree of significance of each attribute, these properties distinguish an agent from a mere computer program.

2.4 Learning and Adapting

What is the difference between learning and adapting for an agent?

Learning can be defined as the process of acquiring new knowledge, behaviors, values or skills during time. Specifically for a machine, this means that through certain algorithms the machine is able to generalize key aspects of the environment where this machine is located, using data gathered along this time. The machine or agent, ideally, will be able to recognize certain *patterns* in the data and generalize from them. When a new time step arrives the agent will be able to compare the datum obtained from the environment to the prediction that the algorithm made so far. This process can go (forever if needed) until the differences between the observed data and the predicted ones are small enough.

In the former section an adaptive system was defined to be a system able to respond to changes in the environment or in the system itself. By *able to respond* we have to clarify that not every response would be valid. Intuitively giving a valid response, or being able to respond to changes, is giving an output which improves the overall outcome of the system compared to the outcome that the system would obtain if the entities did not adapt to the changes.

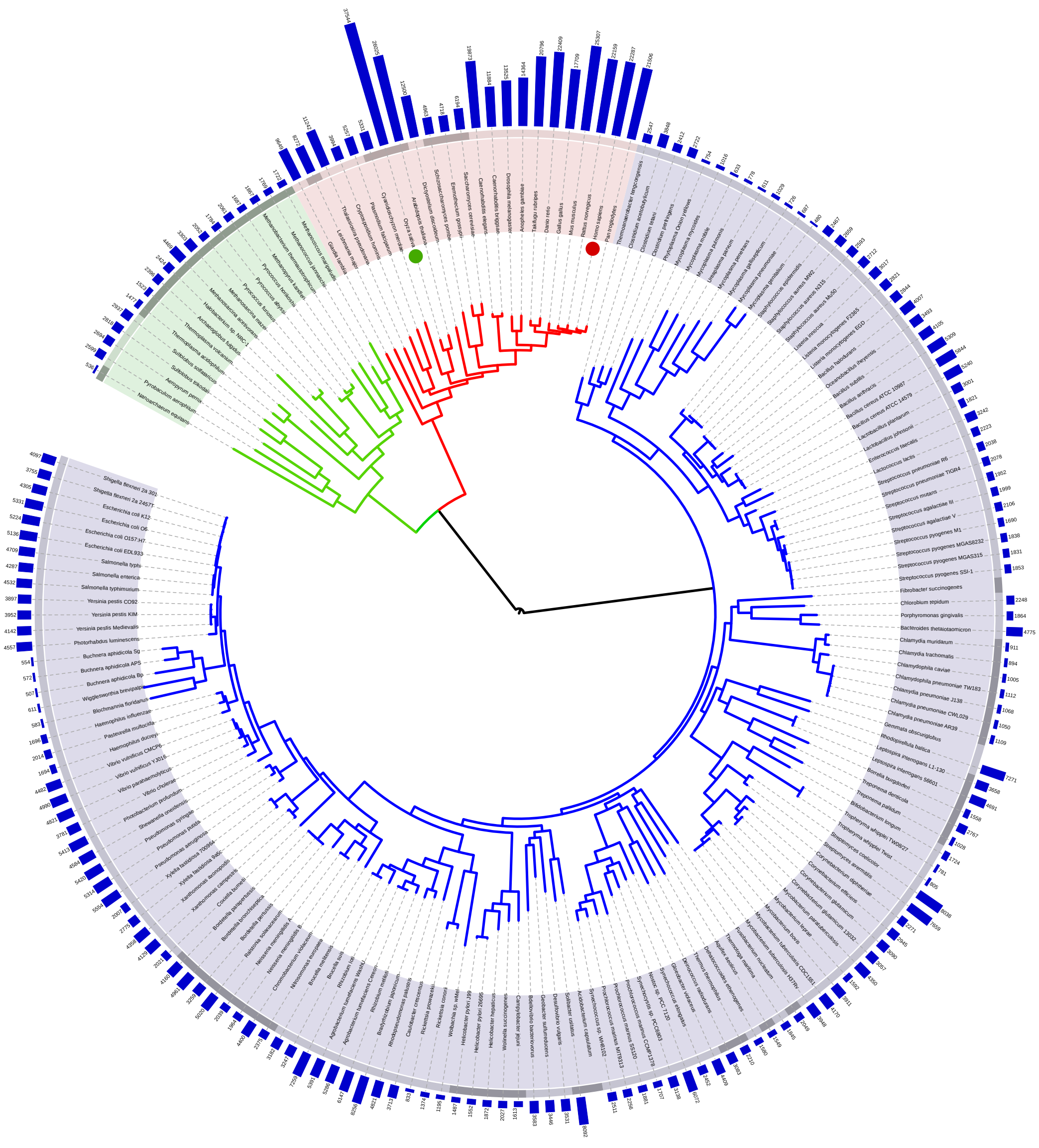


Figure 2.2: Evolution of species is an algorithm which enables adaptation but requires no learning.

A bit more rigorously: let E be an event that will trigger a change or adaptation in a system S. Let the adaptation itself be noted simply as $S \rightarrow S'$. The probability of the system making a change has to be greater when the event E is given that when that event does not occur, but the system must converge to a state when all stimuli ceases. So first we can say that $P(S \to S'|E) > P(S \to S')$ to ensure changes when events occur, and we can later say that $\lim_{t\to\infty} P_t(S \to S'|E) = P_t(S \to S')$ to force convergence of the system to a stable state (when no other events arrive). This way we can define when a change is more probable in an adaptive system.

Let the function $F : S \to \mathbb{R}$ be a measure of the outcome (or *fitness*) of a system, the higher the value F has the better. We can write that given a change in the system $(S \to S')$ it will be considered a valid change iff $F(S) < F(S')$, that is if the change made the system better suited to the new conditions.

So from these definitions we do not need to enforce an agent to have the capability of learning to be able to adapt as part of an adaptive system. But in an intuitive way, is it necessary for an agent to learn to be able to adapt? We can find examples in nature, where evolution has brought different species to different niches very successfully, without the explicit need for them to learn anything (figure 2.2). Ant colonies behave socially because evolution determined a greater success ratio for them when they did compared to when they act on their own. Wolves hunt in packs because they are more successful as well, not because they learned to do so. When a wolf is born he automatically has *hard wired* that when he is mature enough he will not go hunting alone, if possible, because it is more probable to return without a prey than when hunting in a pack of wolves. Those are *rules* which they follow, rules which were inherited and which were created by evolution of this species.

The opposite question can also be asked: is learning a form of adaptation? The definition used for learning does not imply the use of the knowledge that the agent has gained, only that it improves its knowledge, skills, and so on. Thus it can be viewed as a good mechanism which would enable an agent to improve itself, thus, adapt. By conveniently using learning we can have adaptive agents, which is what intuitively we were expecting.

So between changes (events that trigger adaptation) the system must converge to a steady state. In this state the fitness function should be as closest to the optimum one as possible. The question of how many samples must be seen to ensure appropriate learning was already addressed by E. M. Gold [Gold et al., 1967] and we know from [Clark, 2001] that under some light assumptions the learning agent can converge, in the limit, to a sequence where the difference between the observed data and the predicted data is negligible, with probability 1.

The possibility of adapting quickly enough to changes before a new set of changes modifies the environment is of course problem dependent, but since evolution has created a wide range of systems able to deal with changes quite well, it seems plausible to find models —such as the ones found in nature— to deal with the range of changes that we see in natural systems (which is astonishing). On the opposite, for a given system or agent, there will always be an environment receiving changes quickly enough that would not allow the agent or system to adapt to them. Whether that kind of environments are interesting or not is out of the scope of the work.

2.5 Psychological Preliminaries

Two different items are borrowed from psychology in this work. The first one is a measure, called KAI, and the second one is a model for emotions, called OCC. In this section a brief description of both of them is provided to understand the benefits that using them could bring back.

Cognitive psychology studies how humans perceive, think, solve and remember problems, which explicitly acknowledges the existence of internal mental states (intentions, desires, beliefs, . . .). It is closely related to its neurological approach, cognitive neuroscience. We will be focusing in this section in *problem solving*.

In the psychology literature the widely adopted Kirton Adaption-Innovation Inventory (KAI) [Kirton, 1976] proposes that each individual is located in a continuum between "doing things better" to "doing things differently"; i.e. being "adaptive" or "innovative" (figure 2.3). This means that each person has, in the cognitive level, a degree of susceptibility or inclination to be innovative in the way they think.

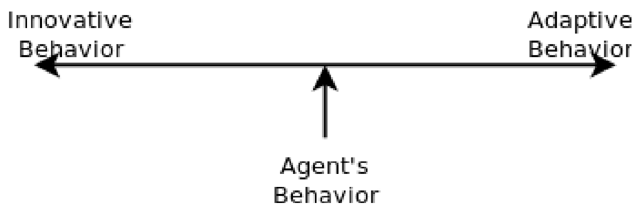


Figure 2.3: KAI index continuum. Any agent's behavior is located in between the two extreme values.

Plenty of work has been developed in cognitive psychology using the KAI theory (e.g. Chamberlain's theory of strategy [Chamberlain, 2010], Modes of Leadership towards uncertainty [Wilkinson, 2006]).

Quoting from [Kirton, 1989]:

> The Adaption Innovation Theory is founded on the assumption that all people solve problems and are creative. The theory

> sharply distinguishes between level and style of creativity, problem solving and decision making and is concerned only with style. Both potential and evident capacity aside the theory states that people are different in cognitive style in which they are creative, solve problems and make decisions. These style differences lie on a normally distributed continuum, ranging from high adaption to high innovation. The key to the distinction is that the more adaptive prefer their problems to be associated with more structure and more of this structure to be consensually agreed than do the more innovative. The more innovative are comfortable solving problems with less structure and are less concerned that the structure be consensually agreed than are the more adaptive.
>
> Those scoring as more adaptive (the terms adaptive and innovative are relative), as measured by the Kirton Adaption Innovation Inventory, approach problems within the given terms of reference, theories, policies, precedents and paradigms and strive to provide "better" solutions (e.g. continuous process improvement). By contrast those more innovative tend to detach the problem from the way it is customarily perceived and, working from there, are liable to produce less expected solutions that are seen as being "different" (e.g. re–engineering). Styles of creativity produce different patterns of behavior. All styles are absolutely essential to deal successfully with the wide range of problems faced by individuals and groups, over time.

A very adaptive person is one which tries to excel at their job, getting the best result out of the known situation. On the contrary a very innovative person would achieve a task by trying different methods, even though some of them would yield dreadful results. We as persons are located somewhere in between the two ends of this continuum. In the bakery example given before, the baker who tries to make bread more efficiently out of the same flour is closer to the "adaptive" side of the KAI continuum. The baker who looks for a different way of making bread —or bakes bagels, for example— is closer to the "innovative" side.

Also extracted from psychology —and independently of the KAI index— we find a model for emotions, called OCC [Ortony et al., 1988] (the acronym comes from the names of the authors: A. Ortony, G. L. Clore, and A. Collins). This model is widely used because it specifies which actions, events or objects will elicit any of the emotions. It is a cognitive approach to emotions which embraces twenty two of them, in eleven different dimensions (see figure

2.4 for a very clarifying diagram). They are eleven dimensions, since all of the emotions are couples of complementary ones, e.g. *hope* and *fear*. The intensity of the emotions is dependent on four variables:

- proximity
- sense of reality

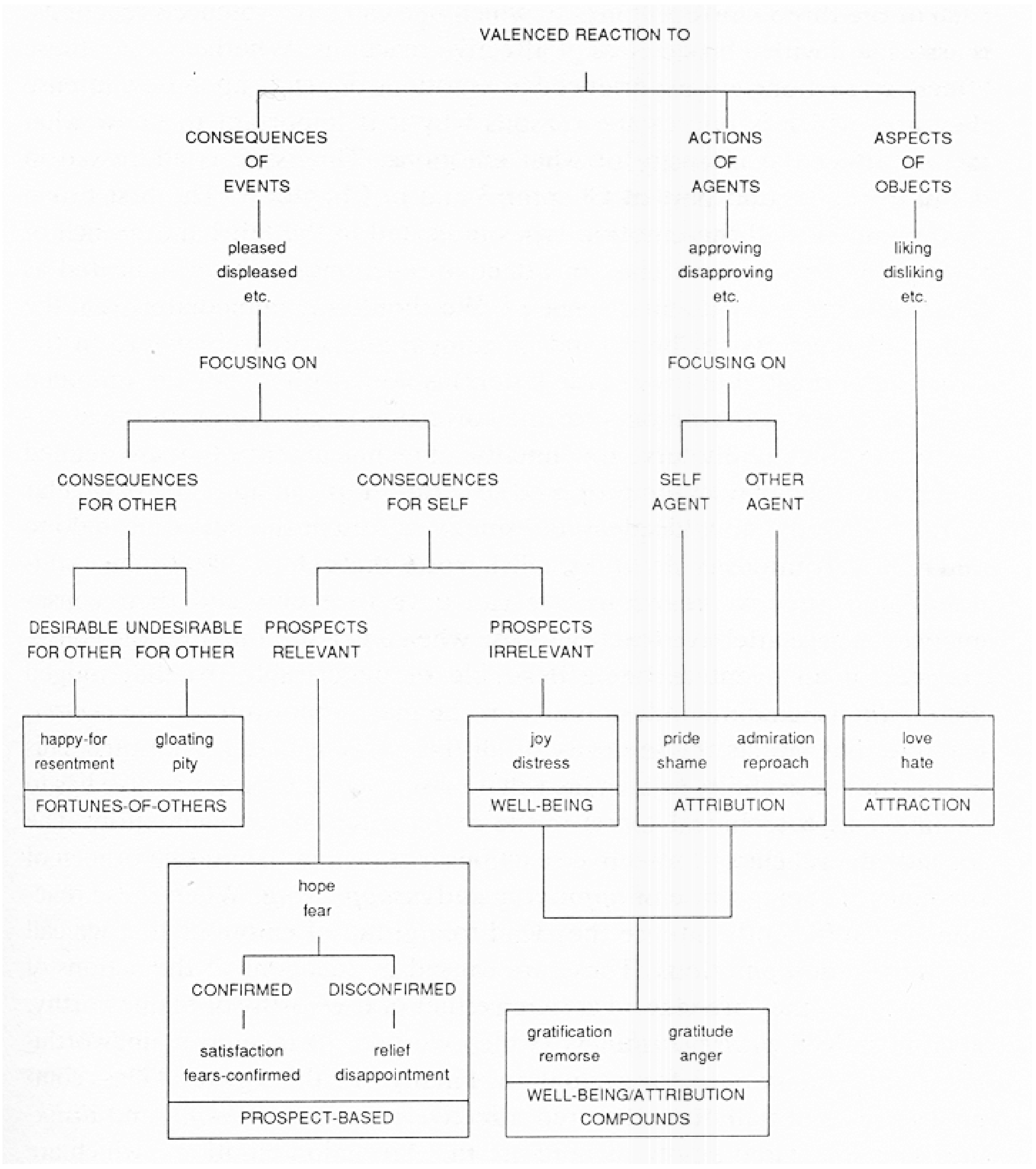


Figure 2.4: Hiearchy of emotions according to the OCC model.

- unexpectedness
- arousal

These four variables would modify the weight of the value for each of the 22 emotions.

The 11 pairs of emotions are:

- <happy–for,resentment>
- <gloating,pity>
- <hope,fear>
- <satisfaction,fears–confirmed>
- <relief,disappointment>
- <joy,distress>
- <pride,shame>
- <admiration,reproach>
- <gratification,remorse>
- <gratitude,anger>
- <love,hate>

Some of the enumerated emotions will be of particular importance, such as <satisfaction,fears–confirmed>, which will give a measure for each individual agent of how well it is carrying its task.

Psychological models such as the OCC model or the KAI index will be used to propose an initial approach to the human relationship to the surrounding opinion, and this model will be used latter as the social agent model. All these in chapter 4.

2.6 Case Based Reasoning

Case-Based Reasoning, or CBR, is basically the process of solving new problems by analogy to others already solved. By remembering similar situations, a CBR system is able to find a situation which is very similar to a new one and apply a solution which is also similar to the solution used in that *remembered* situation. The system approaches the idea that human minds work alike:

when presented with a new problem, a human will try to find another past situation which is similar to the one presented [Aamodt and Plaza, 1994]. The field of CBR systems extends also to cognitive science through the relationship with *Prototype theory* [Lopez De Mantaras et al., 2005].

As an example: lets imagine a person which is confronted with the problem of solving a blockage in one door. The blockage consists in one medium size object sitting right in front of the door, preventing the door to be opened. The person knows that tables can be pushed and move away from their original position, but knows nothing about this new object. Nevertheless the most similar example that is found is the one related to moving tables, so the person reasons that the solution has to be similar to the one of pushing away tables.

For a CBR system, a case is an instance of a problem. If the case was already solved —or learned to solve, or simply seen and marked as unsolvable— it is usually called a past case, stored case or *retained* case. Analogously, a new case is simply a problem which the system is confronted with for the first time (in the general case).

As mentioned in [Aamodt and Plaza, 1994], *the driving force behind case-based methods has to a large extent come from the machine learning community, and case-based reasoning is also regarded a subfield of machine learning.* Learning is important in CBR systems since the idea of analogy learning arises from the very own description of CBR: once the system has given a solution for a new case, it is retained both the case and its solution. Correspondingly is done for *solutions* that were proven to be bad ones. This information storing process can be seen as learning for those specific stored cases. If, by chance, the system confronts a new case which is exactly the same one as a stored case, the system *knows* that it can use it, because the system *learned* that that solution was good for that particular case. The trick comes when, as in real life happens, the stored cases are similar but not exact to the new ones. The CBR paradigm addresses this problem.

The basic CBR cycle (figure 2.5[1]) is described by these four processes:

1. Retrieve

2. Reuse

3. Revise

4. Retain

[1]Taken from [Aamodt and Plaza, 1994]

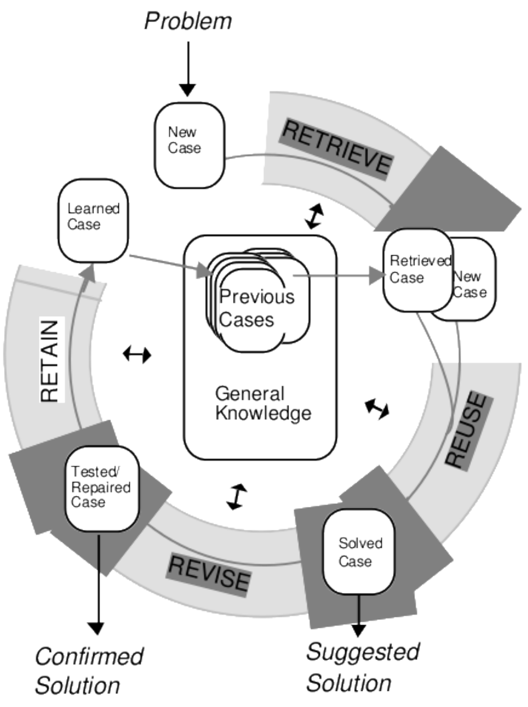


Figure 2.5: The CBR Cycle (the *4 REs*)

The retrieve phase searches one or more past cases to be used as the *analogy* with the new case. This task should also take into account several subtasks, such as identify the descriptors of the cases, search the database based on the values for the descriptors, and most important, compute the similarity from these past cases to the new one. The descriptors can be obtained via an expert on the domain, or inferred —via statistical methods, for example—. The similarity measure implies a distance function which can be given, one more time, by an expert or computed as well —using some underlying model, for example— [Lopez De Mantaras et al., 2005].

The reuse stage is responsible for transforming the solution stored with the past case to a useful one for the new case. When the task is similar to classification, to transformation is needed: the solution is itself the outcome of the similarity function between the retrieved cases and the new one; the highest similarity implies belonging to the same class as that stored case. When the task is of other nature, some adjustments have to be made to the retrieved solution. Usually, as the similarity decreases, the adjustments get more intense. The reuse stage should map the retrieved solution — the solution belonging to the most similar retrieved case— to one inside the solution space, with the property that the solution proposed by the map function should be at least as good as the retrieved one. Ideally, the retrieved solution would be the optimal one for the new case. In the example relating the obstacle blocking the door, the solution should be adapted to the new object: if the variables describing the case of blocking objects include the mass of the object, and the solution includes the force to apply to that object, the mapping function in the reuse stage should adapt that force to the estimated one in the new blocking object.

Revising the proposed solution is when the CBR system has the opportunity to learn new examples. In case the proposed solution wasn't satisfactory, a new solution can be provided using domain dependent knowledge. This new case would include the newly created solution. Learning from the failures is, for a CBR system, as important as learning from the successes.

The retain stage keeps track of the data that need to be updated in the case base. The CBR systems must update their case base regardless of the initial outcome of the proposed solution. The well-behaved solutions must be kept so future new cases can adapt their solutions using these ones as basis. The solutions which failed when they were applied must be kept also, to *remind* the difference between a success and a failure in the reuse stage: when mapping a retrieved solution, if it is more similar to a failure than to a successful solution, the system should prevent, at first stage, giving that solution as the correct one. Also, the system can always improve the understanding of the failure when presented another one.

Using again the blocking object example, if we apply the transformed amount of force (computed at the reuse stage) and observe that the object moved much further than expected (and assuming that is not desirable), the revision of the solution and the retain stage should realize that the variable *number of wheels* does in fact matter for the computation of the proposal. The retrieve phase, when entered again once a new case arrives, would emphasize the weight of that variable when determining the similarity to other objects: objects which have three or more wheels and approximately the same mass are similar to the object mentioned before, more than objects which do not have wheels but have exactly the same mass. The reuse stage should, as well, map the solution taking into account the differences between *number of wheels* and *mass* of the new object and the retrieved ones.

2.7 Reinforcement Learning

Reinforcement Learning, or RL, is a technique that allows an agent to learn which chain of actions is best suited for its goals. RL takes into account that an agent is situated in an environment, thus the actions modify the state of this environment. The relationship between the environment and the agent itself is exploited by this technique.

Every action that the agent performs in the environment may have consequences. These consequences may be observable at the time the action was executed or may be in the future. The problem the agent is facing is one of decision making: which chain of actions will allow the agent to reach its goal? The data of how well an action performed is usually available in the future, not at the time when the agent has to make the decision. RL will allow —after a very important learning stage— to decide the best next action, by only observing the current state of the environment. This is the learning task.

If we consider to build a robot, which has to carry out some duty, such as surveillance, we may build this robot with several actions available: to move forward and backward, to turn, and to stay still. Also, we want the robot to sense the environment, so we build some sensors in it, such that they can give the robot's position, and also a camera to detect objects.

The robot has to patrol some area, but also has to recharge batteries when the battery level is low. It would be desirable to have this robot learning for some time how is the environment, so afterwards it could move and recharge. This way the robot would be *scenario agnostic*, or independent of the different environments where we decide to put it in.

Of course the robot is an agent: it fulfills the requirements of being proactive and reactive, being independent and it could communicate with some other robots if we design it to do so. How can this agent gather the knowledge to reason about which strategy should follow in every case? RL will address that issue.

Let's consider the example of the robot: moving forward or backward, as well as turning or recharging do not always provide a feedback of "how well the action went". Nevertheless the aggregation of some actions (called *policy*) will provide a positive outcome, or reward, when carried out in certain states of the environment. If we assume that this positive outcome depends on the policy, which started in a certain state of the environment, the agent can try the whole set of policies starting in the whole set of states of the environment. In the robot example, that could lead to an endless loop of useless chained-together actions, such as repeatedly moving forward and backward forever. Instead, if modeling the learning problem as a time dependent, transition problem, we can figure out that some states are actually reachable by different means: it is only a matter of time until the agent finds a way to reach certain states —always assuming that these states are somehow reachable—, and the agent, through a discoverer behavior, will also find the most satisfying way to reach these states. Here is where the problem is modeled as a Markov Decision Process (MDP), as in figure 2.6.

The MDP is modeled as a 4 tuple (S, A, δ, R), where

- S is the set of states of the environment. It has to be finite.
- A is the —also finite— set of actions.
- $\delta : S \times A \times S \to \mathbb{R}$ is the transition function, which gives a probability such that $\delta(s, a, s')$ is the probability to transit to state s' being in the state s and performing action a.
- $R : S \times A \to \mathbb{R}$ is the *reward function*, which assigns rewards to actions performed in certain states. $R(s, a)$ is the reward —immediate reward— obtained by executing the action a in the state s.

In the figure 2.6 the only actions performed are the a_i, a_{i+1}, a_{a+2}, from the states s_i, s_{i+1}, s_{i+2}, and the rewards where $R(s_i, a_i) = r_i$, $R(s_{i+1}, a_{i+1}) = r_{i+1}$ and $R(s_{i+2}, a_{i+2}) = r_{i+2}$.

The transitions are deterministic —for now, since there are no other samples but the one shown in the figure— thus $\delta(s, a, s') = 0$, $\forall s, a, s'$, excepting $\delta(s_i, a_i, s_{i+1}) = 1$ and $\delta(s_{i+1}, a_{i+1}, s_{i+2}) = 1$

In this context the learning task is now more formally defined as to learn the best policy for the environment (π^*). The policy, $\pi : S \to A$ is sometimes called *response policy*, when in the context of game theory. Then the policy function π will provide the agent with the next action a_t to execute at time t while the agent is in the state s_t. Chaining the functions π and δ the agent has its behavior defined. Its goals have to be represented through the MDP model, and the reward function R is critical to define *what the agent wants.*

The agent must learn the optimal action policy π^* only through the rewards over time. To do so the agent will take into account that the future rewards are important, so the concept of *discounted cumulative reward* is defined:

$$V^\pi(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{2.1}$$

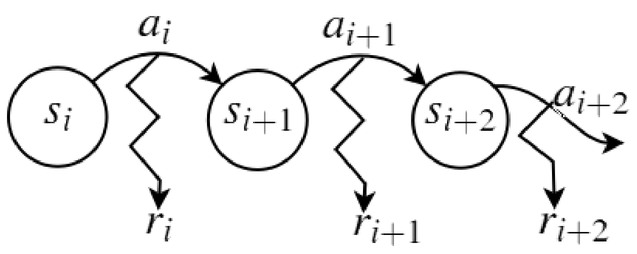


Figure 2.6: A simple Markov Decision Process

In equation (2.1) V^π is called discounted cumulative reward because: it is the sum over time of all the rewards that the agent is going to get, and because every future reward is discounted by a small constant factor γ $(0 \leq \gamma < 1)$, giving more importance to rewards obtained near the present that further in the future. The closer γ gets to 1, the more importance will get future rewards.

Now, the agent has simply to find the policy which maximizes $V^\pi(s_t)$, $\forall s_t$ in the environment. This will be the optimal policy (π^*):

$$\pi^* \equiv \underset{a}{\operatorname{argmax}} V^\pi(s), \ \forall s \tag{2.2}$$

How to find π^* is what the RL algorithms deal with. The most famous and well known is called *Q-learning*.

Q-learning takes advantage of the Bellman's equation for optimization: it breaks the search for the policy maximizing the value of V^π into two pieces: the immediate reward one, and the cumulative part. The algorithm uses a function called $Q : S \times A \to \mathbb{R}$ following the spirit of V^π. To simplify notation we can use $V^* \equiv V^{\pi^*}$ as the discounted cumulative reward for the optimal policy, and redefine δ as $\delta(s,a) = \operatorname{argmax}_{s'} \delta(s,a,s')$. This way δ is now the function which returns the most probable state for the transition starting in s and executing a. The Q function is defined as:

$$Q(s,a) \equiv R(s,a) + \gamma V^*(\delta(s,a)) \tag{2.3}$$

The first piece, $R(s,a)$ is obtained instantly. The rest can be computed by dividing again the value of V^* into a definition related with the Q function. By definition, and using equation (2.3), we know that:

$$V^*(s) = \max_a Q(s,a) \tag{2.4}$$

Then, plugging in the new definition for V^* into equation (2.3) the recursive definition of the Q function is:

$$Q(s,a) = R(s,a) + \gamma \max_{a'} Q(\delta(s,a),a') \tag{2.5}$$

Now the Q–learning algorithm has to estimate the values of each possible $Q(s,a)$ for every pair state–action. It can do so by exploring the environment for an important amount of time. The algorithm can use a tabular representation for Q and update the value $Q(s,a)$ every time the agent executes a while in state s. The updating process for Q–learning is defined as:

$$\hat{Q}(s,a) \leftarrow R(s,a) + \gamma \max_{a'} \hat{Q}(\delta(s,a),a') \tag{2.6}$$

The updating process is using the $\hat{Q}$ values, which are the estimates that the algorithm keeps for the Q function. With enough iterations —visiting infinitely often all the state–action pairs— the Q–learning algorithm will make $\hat{Q}$ converge to the real Q, if the environment can be modeled as a deterministic MDP. A very clarifying proof can be found in [Mitchell, 1997], pp. 378–379.

When the environment is not deterministic the transition function δ —which we had redefined as $\delta : S \times A \to S$— and the reward function $R : S \times A \to \mathbb{R}$ have an *associated probability distribution.*

The discounted cumulative reward has to be redefined as the expected value of the "old" V^π:

$$V^\pi(s_t) \equiv E\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right] \tag{2.7}$$

And the definition of the Q function is the generalization of equation (2.5):

$$Q(s,a) = E[r(s,a)] + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a) \tag{2.8}$$

The analogous updating process for the nondeterministic case follows the rule:

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n \left[\max_{a'} \hat{Q}_{n-1}(s',a')\right] \tag{2.9}$$

The term α_n, $0 \leq \alpha_n \leq 1$ is used to update the $\hat{Q}$ value averaging the old value and the new one. The term α_n can be non–constant, i.e. computed by a function; this is the case needed to assure that the method will converge, to have α_n increasing as the number of updates increases as well. A proposal for α_n [Mitchell, 1997] is given by:

$$\alpha_n = \frac{1}{1 + \text{updates}_n(s,a)} \tag{2.10}$$

The function $\text{updates}_n(s,a)$ returns a scalar representing the number of times that the pair (s,a) has been updated by rule (2.9). This way the values of $\hat{Q}$ are updated "less and less" —meaning that the change tends to 0, thus it is useful for convergence—.

The convergence of the learning process is also guaranteed provided that the problem can be modeled as a nondeterministic MDP —that is in this case that the probability distributions depend only on s and a and not, for example, on previous states or actions— and it is guaranteed that $\hat{Q} = Q$ when $t \to \infty$. The proof can be found also in [Mitchell, 1997], pp. 382–383.

The agent must explore the pairs state–action as many times as possible to find out the Q values. But the agent's goals probably do not include to explore the environment, but to carry out some other task. The exploration is an *innovative* action needed in order to be able to exploit the environment for the agent's profit. The exploration–exploitation dilemma addresses just the question of how much to explore and when to start exploiting the information of Q.

Intuitively the probability of choosing an "optimal" —accordingly with Q— must increase with time, since it is supposed that the information of Q becomes more accurate with time. In [Mitchell, 1997] one proposal to choose an action from a given state is provided:

$$p(a|s) = \frac{k^{\hat{Q}(s,a)}}{\sum_{a'} k^{\hat{Q}(s,a')}} \tag{2.11}$$

As the scalar $k, k > 0$ increases the equation assigns higher probability to actions exploiting the environment. Of course other functions for k can be provided (as it is provided in section 5.2).

2.8 Previous Approaches to Adaptive MAS

Different approaches to Multiagent Learning (MAL) are examined here.

Most of the literature surrounding the field of learning in MAS's is focused on reinforcement learning, and adaptive MAS are not an exception. Thus it is interesting to see first a set of some of the algorithms used by the agents to learn about the rules in the environment.

The fields of *Temporal-Difference RL* (TD) [Sutton and Barto, 1990] and *Game Theory* [Von Neumann and Morgenstern, 1967] are usually involved in these algorithms. From the point of view of TD, the algorithm relies on dynamic programming to find the best suited policy for the given environment, following the Bellman's equation [Bellman, 1957]. No modeling of any other agent is done. The algorithms usually aim for convergence on their rewards.

On the other hand, the Game Theory approach focuses on Nash equilibria. That means that no agent can do better by changing its action unilaterally. It is called equilibrium because it is supposed that agents will try to do as best as they can —property which is called rationality—. Since they cannot do better by changing its behavior, they will not change it. The problem arising is that for any given environment there may be many Nash equilibria, thus either all of the agents find themselves in one of those, or the equilibrium from the point of view of the whole system is not reached.

Some of these algorithms make use of an explicit coordination mechanism. To do this, some assume certain behavior of the agents —such as that they always behave rationally. Of course, every algorithm has its limitations, and its advantages.

There are some algorithms which through an explicit coordination mechanism deal with the problem of MAL. One of them is Q-Learning with SSA and ABAP [Melo and Ribeiro,]. The core of the algorithm deals with the problem of finding the same Nash equilibrium point out of the many possible equilibria. It assumes the agents will behave rationally to achieve this goal, but does not need assumptions on the environment.

Some other algorithms based on game theory —just to mention some of the well-known ones— and Nash equilibria are: Nash-Q [Hu and Wellman, 2003] —which needs the agents to be rational and has some convergence problems [Bowling, 2000]—; Nash-DE algorithm [Akchurina, 2009] —which converges to a stationary Nash equilibrium in the general case, with some assumptions, but still needs the player to behave rationally—. There are some other approximations for the *general sum* games but they all need the agents to behave rationally.

On the other side from rational behavior there are some approaches based in emotion modeling. These modeling can be usually seen in rule-based systems, expert systems. These have the advantage to behave more human-like than rational agents —property which is sometimes needed—. Furthermore emotions can bias an agent decision, thus emotions can be seen as heuristics as well. The work described in [Steunebrink et al., 2007], based on *2apl*, is very clarifying. Emotions do not provide a mechanism for cooperation, but it doesn't prevent it either. The environment is not restricted here, but since it has a strong basis on psychological models, the agents are supposedly able to observe other agents actions and emotions, to develop their own.

Also in the reinforcement learning (RL) literature we can find an algorithm which explicitly tackles the problem of exploration - exploitation: how much to explore and when should the agent use the gathered information. The algorithm E^3 —or extended version $MDP{-}E^3$ [Kearns and Koller, 1999]— does not have strong restrictions on the environment, only that it should be static. That —and the fact that it does not work with multiple agents— renders the algorithm unsuitable for MAL and adaptation. Still is a very significant approach to the specific treatment of the probability of exploration.

The closest algorithm in spirit to the work presented here is the WoLF algorithm [Bowling and Veloso, 2002a] —there is a revision of the algorithm to allow high-dimensionality environments [Bowling and Veloso, 2002b]. Both require the agent to know its payoff matrix, and both allow dynamic environments since it *learns faster* when results are not as expected. There is, though, a problem with the convergence of learning: if the agents do not follow stationary strategies the algorithm will not ensure convergence.

Although there is some literature on *Social Reinforcement* [Mataric, 1994, Fabregat et al., 2008], our approach is completely different. The reinforcement will always and only be given by the environment, and the society will just give its opinion, aiming this opinion towards changing the behavior of the rest of the agents.

3

Composed Environment

Contents

3.1 Significant Changes and Adaptation

In any open multi-agent system that allows heterogeneous agents, the adaptation of part or of the whole society is a difficult task [Helleboogh et al., 2007]. Innovation should occur when changes that arise are significant enough to yield losses higher than the cost of the adaptation itself to the new conditions. Detection of the importance of changes in the environment is not naive, since the best sequence of actions —best policy— is unknown a priori; we cannot obtain reference values to compare neither the individual behavior of each agent nor the one of the whole society.

For all non-static environments, we can say that there are certain changes that require innovation from the agents. The changes that could lead to that innovation on the agents can be divided into the following categorization:

Mechanical Changes these happen in the physical environment, such as changes in rules or characteristics of the environment where agents are located.

Social Changes these happen in the organization (in the set of agents). These changes include, but are not limited to:

- A new social agent appears.
- An existing social agent disappears.

Any agent in the system that is not a social one will represent changes in the environment, as social agents cannot detect its presence. These *non-social agents* will introduce a *noise* both in the reward and transition functions of the environment, and since there is no constraint about the observability of any outcome of the actions of any agent, this change will make deterministic environments behave like non-deterministic ones —always from the point of view of the agents situated in this environment. This is essentially the reason why classic reinforcement learning algorithms —such as Q–learning— cannot obtain the optimal policy in a MAL.

Despite our efforts to find a suitable representation, changes on how the environment responds to an agent's action occur. Some representations will be more sensible than others to those changes, but for a suitable mapping function (from the environment to the agent's representation) there always be some changes that would alter (and should alter) the agent's view of the environment, and its behavior. This is notorious when the environment changes physically: e.g. a known set of rules of production change in a supply chain management problem. As an example —in the context of supply chain management—, in a market there may constantly appear new ways to build new products, such as by introduction of new objects (functions) which allow actions unknown until that moment, or by changing prices, timings or dependencies. All this *mechanical* changes will have an effect on the system.

Trying to adapt a multi-agent system in a *bottom-up* way, thus trying to exhibit emergent behavior through the design of the individuals, has been always desirable, as long as the the system as a whole gets adapted to the changes. We have to remind that by *adaptation* we refer to the opposite of *innovation*, according to the *KAI index* mentioned in section 2.5. To have an adaptive behavior in the society when a change —a significant change— has

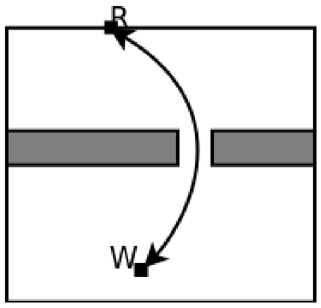


Figure 3.1: Adaptive robot in the surveillance - recharge task. Two fixed points are important: the Watch point and the Recharge point (denoted W and R respectively).

happened, first the society needs some innovation; after exploring the new opportunities that may have arisen —or realize that former opportunities do not exist now— the society can exploit this information.

We will address the problem of deciding when to be innovative or adaptive through the *Social Opinion* (R_Σ and κ functions, to be further explored later in sections 4.2 and 5.2).

If the agent is doing better each time, or approximately doing as good as before, the agent should *adapt* its behavior (according to the KAI index described before). If, on the contrary, is doing worse than before, the agent should innovate its behavior. Since *doing things better* is sensed locally, the senses of other agents should help to figure out if the society as a whole *is doing better* or not. In the section 5.3 a bit more about this advantage of having agents located in different states in the environment is discussed.

3.2 Innovative and Adaptive Agents in the Environment

The actions available in the environment can affect both the mechanical part or the social one. In RL the actions are usually seen as mechanical actions: actions that alter the mechanical state of the environment. But not every action has to be classified as mechanical actions; there can be some of them which affect only the social environment. An example is given: let's have an environment in which inside a room a robot has to perform some surveillance, always remaining in a fixed point (like a guard). But as time goes by, the battery of the robot decreases, so every once in a while the robot must find a charging point to recharge the battery, and then go back to its watch point.

Let's have a robot which is totally adaptive, which knows already where a charging point and its watch point are, and also knows the path between its watch point and the changing point (as in figure 3.1). We say this robot is totally adaptive because it will not try to explore different paths, nor find different charging points out of the knowledge it has. As long as the environment stays static from the point of view of the robot, everything will be performed as before, with no waste of time wandering around the room. There are some remarks here:

1. What does it mean that the environment stays static?

2. How did that adaptive robot gain the knowledge it has?

To answer the first question we position two different observers: one inside the environment, immersed in it, and one outside the environment, with omniscient capabilities (like the "All-Seeing Eye"). To the first observer, a change happens when the observer notices a different behavior which was not expecting. That means that the first observer, which can only perceive the environment from the state it is immersed, will not notice changes which only affect states the observer will not visit. If the observer visited every one of the possible states of the environment, all changes would be discovered. The assumption that the changes remain in the environment until the observer visits those states has to appear to claim than an immersed observer could notice the changes. So, for the first observer, the environment remains static as long as the changes are not visible from the states it visits, or there were no changes. To the second observer however all changes are visible at any given moment. This kind of *oracle* would say that the environment is static if and only if there were actually no changes.

The answer to the second question is that currently it does not matter, at least to define static environment or change detection. The robot could have been programmed with those behaviors a priori, or it could have learned them. That will not change the fact that the robot will not change its behavior in a static environment. If the robot learned that behavior it means that it had to explore the environment to figure out which behavior it should follow. From this exploration the robot gained the knowledge required for its task. After that, the robot had to cease its exploration behaviors and only "focus" on the surveillance-and-recharge task.

In the case we find ourselves with a totally innovative robot, it will wander around the room without trying to follow any behavior which would accomplish its main task, which was to do surveillance and recharge batteries when needed (figure 3.2). But this robot will be able to detect changes more easily

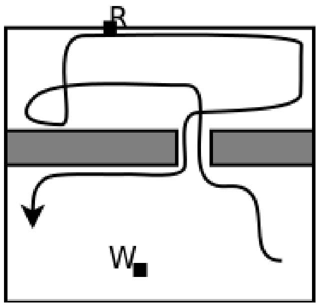
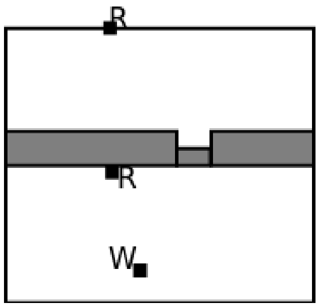


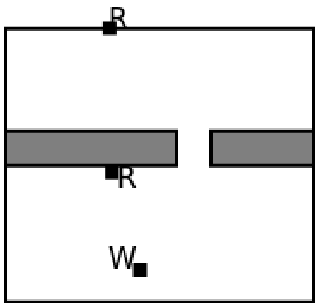
Figure 3.2: Innovative robot in the surveillance - recharge task.

than the former, adaptive one, because its goal is not to accomplish its task, but actually to explore as much as possible the environment.

Now, if a change is introduced, the adaptive robot will not be aware of it unless the change is actually modifying the path it is following in some way. An obstacle in the middle of the path would do exactly that. But if the change is to add a new point of recharge (figure 3.3(a)), maybe nearer the watch point, the adaptive robot will not notice it.



(a) Only a significant change is introduced (a new recharge point).



(b) Two significant changes are introduced (a blockage in the path to the other side of the room and a new recharge point).

Figure 3.3: Changes are introduced in the surveillance - recharge task.

On the contrary, since the innovative robot does not care much about its goal —surveillance and recharging—, the changes will be noticed faster than by its adaptive analogous robot. Not only faster, there will be some changes that the adaptive version will never see, such as in the figure 3.3(a). The innovative robot will see this change faster because detecting it depends only on the probability of the robot to be in a state where the change can be locally sensed, which —intuitively— will be higher for the one which is wandering around and visiting different states than for the robot only visiting the most rewarding states (the path and the watch and recharge points). The distribution of probability would affect this assumption, but since there is no information about where the changes will be produced, we cannot restrict this distribution to a known one. Given the particular case when this distribution places changes with higher probability in the "path" the adaptive robot follows, the innovative one will have less chances of finding changes. But again, that would require the probability distribution to "know" about the behavior of the adaptive robot, which is a very unusual case [1].

3.3 Composed Actions

The set of actions available to an agent is dependent of the agent itself and the state that agent is located in the environment. Some actions are available only in some states, while others are universally present along the environment's states. Usually the actions are seen as functions modifying the state of the environment. Some outcomes of these actions yield results which are positive for this agent, while some others may be neutral or counterproductive. These outcomes will be treated later when discussing the resemblance to MDP's.

An action can be seen as a function $f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$. $\mathcal{X}$ is the description of the environment —the set of variables that through a bijective mapping function represent the real universe—. The representation of $\mathcal{X}$ is also agent-dependent, but only those which are suitable are considered. From now on, $\mathcal{X}$ will simply denote the union of the representations of the environment for all the given agents —the complete finite descriptive set of states of the environment—.

Any action needs a time to be completed; that was not denoted thoroughly in the description above. The time needed by a given action f depends on the state of the environment $\mathcal{X}$ and the agent performing the action A_i.

The set of actions can be divided into *mechanical actions* and *social actions.*

Definition 3.3.1. *A Mechanical Action is a function $f_M : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$, which require not to communicate with other agents.*

Then for the mechanical actions communication is forbidden explicitly.

[1] In can happen, though, that the changes are introduced by an oracle outside the environment, which is observing the behavior of a given adaptive robot, e.g. a human testing the robot itself.

Definition 3.3.2. *A Social Action is a function $f_S : \mathcal{A} \times \mathcal{A} \to \mathcal{X}$ which require communication between agents.*

And for social actions the functions need to explicitly communicate between agents. This communication may be established by means of mechanical objects in the environment —such as pheromones or even the spoken natural language between humans which is carried out by physical means— but in any case both agents need to *understand* the symbols used in the communication channel, whichever it would be.

Using the example shown in figure 3.3, lets position two different robots —agents— in the environment. The mechanical actions would be the ones mentioned before, such as move in the environment, watch and recharge.

An example set of social actions would be composed by these four actions: (from the point of view of one of the agents)

- Order an agent to perform action k
- Order an agent to inform about its state inside $\mathcal{X}$
- Inform an agent about my state in $\mathcal{X}$
- Inform an agent about the actions available in my state $\in \mathcal{X}$

These social actions change the internal beliefs of an agent. Of course the beliefs of an agent are represented, as said before, in a suitable way, inside $\mathcal{X}$.

For the example given before a curiosity shows up: composing social actions is possible very easily: lets add a new agent and name the three of them as A_1, A_2, A_3. If A_1 issues the action of requesting A_2 to perform *watch*, A_2 can easily refer this “order” to other agent, say A_3. The outcome of this action will not be the same of the original one, since A_2 and A_3 are not in the same state in the environment, but nevertheless the decision of performing the action by the agent itself or referring the action to other agent can be taken. A clearer example of function composition can be given in the example: lets have A_1 issue an action to A_2 which reads: “order agent A_3 to order me to inform A_3 of my state in $\mathcal{X}$”. The composition is actually seen as: $\text{order}(A_1, A_2, \text{order}(A_2, A_3, \text{order}(A_3, A_1, \text{inform})))$.

The composition of actions is not actually anything new: the mechanical actions are composed every time. E.g. when performing *watch* it is done on the environment modified by the actions taken before:

$$\text{watch}(\text{move}(\text{move}(\ldots, A_1), A_1), A_1)$$

Of course there is more than one agent in the environment, so the actions taken by the rest of the agents would modify as well the state of the environment, which is needed as the argument to the next action to be taken by the agent A_1. As said before, it is this fact the one that does not allow classic single agent RL algorithms to operate perfectly in a MAS: the composition of the actions are not only chained by our own actions, but also by the actions of other agents.

3.4 Composed Environment

Having two different types of actions defined, the environment can also be divided into two different not mutually exclusive environments: the mechanical and the social one.

The mechanical environment is defined as the set grouping together all the objects —variables— that the mechanical actions can modify. That is, the set of variables that may change once a mechanical action is performed. For a given mechanical action only part of the set of agents will notice a change in the mechanical environment. This set is a subset of the agents situated in the environment which are capable of observing the state of the *nearby* environment.

The social environment, analogously, is the set of variables that can be modified through the set of social actions. In this case, the variables may be only observable by the agent initiating the action or the one which is carrying it (or both). No other agent will be able to observe any change in these variables —which conform the social environment— after carrying out the social action.

Definition 3.4.1. *The mechanical environment is defined as the set containing the variables which are the union of the images of the mechanical actions:* $\mathcal{X}_M \subseteq \mathcal{X}$*.*

Definition 3.4.2. *The social environment is defined as the union of the images of every possible social action f that can be ever executed (F):* $\mathcal{X}_S = \forall a, a' \in \mathcal{A} \bigcup_{f \in F} f(a, a'),\ \mathcal{X}_S \subseteq \mathcal{X}$

Furthermore, the environment can only be built by the aggregation of the two sets described above.

Definition 3.4.3. *The environment is built from the union of the mechanical and social environment sets:* $\mathcal{X} = \mathcal{X}_M \cup \mathcal{X}_S$

Through these definitions the differentiation between social and mechanical environments can be established. It is out of the scope of this work, but worth to mention it, that in the social environment some *objects* can emerge. Concepts such as *organizations*, *roles* and *norms* can emerge through the use of social actions. These objects do not exist in the mechanical environment, nor have an analogous representation there.

For example, through communication agents can establish a hierarchy or orders: agent A_1 orders agents A_2 and A_3 to perform some actions. Agent A_2 usually delegates and orders other agents to perform these actions. Agent A_2 just orchestrates the results. Here, the concept of *role* has emerged through the repeated use of social actions. In a similar way the concepts of *society* and *norm* arise as well.

4 Agent Modeling

Contents

4.1 Psychological model

The psychological model is based in the OCC [Ortony et al., 1988] emotional model. In human emotional modeling we use emotions —plus some other factors— as the inputs which govern the opinion the individual will give to the society. The opinions of the people —which are in this individual's social network— will influence his KAI index. The KAI index will determine whether to behave innovatively or adaptively. The outcomes of the actions of the individual, again, will be evaluated through his emotional system to produce an emotional response, which is a set of emotions and their intensities. Figure 4.1 summarizes the relationships.

There are some remarks which are particularly interesting. The individual

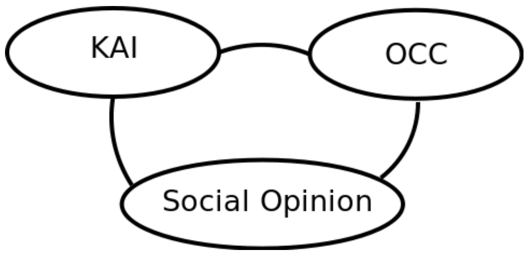


Figure 4.1: Human model based on the OCC emotional model transformation.

will take into account all the opinions of his social surrounding, or social network, but not every opinion has the same importance. People which are trusted by the individual (e.g. his family) will be more trustworthy than others (e.g. people that he encounters on the subway). The same principle applies to the reputation, i.e. people known to be experts in the surrounding environment will be taken into more consideration than those known to be novice (e.g. elder family may have a stronger impact than some other family members which are younger than himself).

Another important step is the way each individual interprets the outcomes of their actions. In Figure 4.2 a set of 11 functions are shown which transform the *emotional stimuli* to a set of 11 emotions with intensities, which just is the *emotional response*. This set of functions is peculiar for each person, so it is part of their *personality* —which is invariant. The functions which

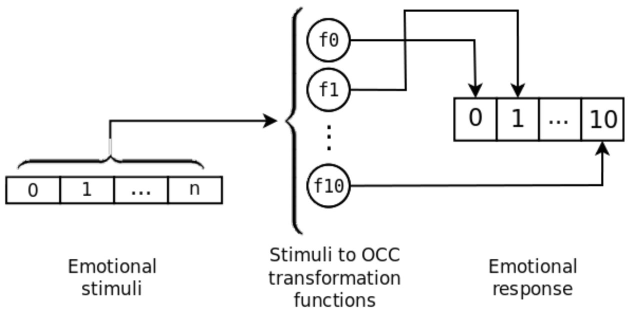


Figure 4.2: Emotional response computed for an individual. Each of the 11 emotions are *computed* by a different function.

conform the emotional response can take any number of observable events or objects in the environment, as well as the observable inner state of the individual, in a cognitive fashion. The functions cannot take any parameter which is not observable by the individual, e.g. the happiness of somebody else; just the expression of happiness could be a parameter. The functions must follow the guidelines described by the OCC model, such as some events should influence some emotions and not others [Ortony et al., 1988].

4.2 A KAI based emotional social agent model

One of the aims of this work was to propose a plausible model close to a human psychological model for a *social agent* that is an agent which is able to express its opinion and to hear the opinions of the agents in its social network. We will use the psychological model proposed in the former section to postulate a new social agent model that we called "KAI based emotional social agent model".

An agent is situated in an environment, but this environment is not only the physical world ruled by mechanical changes; it also comprehends the social world. The environment $\mathcal{E}$ is modeled as $< S, A, \delta, r >$, being S the set of states the environment has; A the analogous set of actions; $\delta : S \times A \rightarrow S$ the transition function, which changes the environment state to another when an action is performed; and $R : S \times A \rightarrow \mathbb{R}$ the observable reward function, that yields a reward when an action is performed in a state.

The theoretical interest of having *emotions* is double: On the first hand agents that are modeled with emotions resemble better human behavior —by not showing a pure rational behavior—, thus the model can be more accurate when simulating humans. On the second hand, there is psychological and neurological evidence [Martinez-Miranda and Aldea, 2005, Oatley et al., 2006, Damasio and Sutherland, 1995] that emotions may be *necessary* for a person to behave intelligently, which arises questions about their necessity in agents.

The transformation between emotional stimuli and emotional responses will be carried out by a set of 11 functions which output the intensity for each one of the 11 dimensions. Again, the functions' domains can be composed only by the set of observable and measurable objects and events of the environment and some actions of the agents located there. The emotional response is very important when emitting each agent's opinion. The Social Opinion will be the aggregation of every agent's opinion taking into account the trust for each of them, and the reputation which each agent has in this society.

An agent is modeled as $< \mathcal{P}, \mathcal{S}, R_\Sigma, \kappa >$. $\mathcal{P}$ is the set of 11 functions $\mathcal{P}_i : \mathbb{R}^n \rightarrow \mathbb{R}$, which map the emotional stimuli to the emotional response, and should —just as a reminder— follow the OCC model of emotions; $\mathcal{S} \in \mathbb{R}^n$, is the set of emotional stimuli, which can be any observable data in the environment and in the agent itself, suitable to be an input of some of the $\mathcal{P}$ functions; $R_\Sigma : \mathbb{R}^{11} \rightarrow \mathbb{R}$ is the social opinion function, used to expose a *public, observable opinion* about the agent's emotions; and $\kappa : {R_\Sigma}^n \to \mathbb{R}$ is the KAI index function calculator, which computes how innovative this agent is being given the emotional response.

The κ function is closely related to the KAI index: the KAI index measures how an individual is in between the two values of being innovative or adaptive, which can be reinterpreted as being exploratory or trying to exploit the information gathered. Or course the values will not correspond one to one between the two concepts, but the similarity of the concepts is there. A suitable transformation to compute κ is given in section 5.2.

Will the emotional response functions $\mathcal{P}_i$ be linear, they can be represented by a $11 \times n$ matrix P: all the intensities for the emotions of the OCC model are computed as $\mathcal{R} = P \cdot \mathcal{S}$, where $\mathcal{R}$ is the 11×1 vector containing the emotional response intensities and $\mathcal{S}$ is the $n \times 1$ vector for the emotional stimuli data.

In section 5.1 we give explicit functions for $\mathcal{P}$, R_Σ and κ. All the functions presented in $\mathcal{P}$ and the function R_Σ are linear (not the κ function). There exist other definitions for these functions which work well —or better— under this model; we do not claim that they are optimal in any sense, only convenient.

4.3 Differences between Innovative and Adaptive Agents

Even when an important change is made, if the agents in the system are behaving in an adaptive way, it could happen that they never realize there was a better solution available after the change: they are not innovating to explore some other possibilities. This problem is hard to solve because two different problems come together:

- Agents do not know when a change will happen.
- If there is a change, the agents will have to search for differences in the environment behavior.

As seen in the relationship between agents and environment in section 3.2, adaptive agents are prone to miss changes. Their behavior is almost "hard coded" once they discoverer a way to achieve their goals. Innovative agents, intuitively, will find these changes faster.

Lets start by defining the changes in strategies for both types of agents:

Definition 4.3.1. *An adaptive agent is one which tries to improve its performance by improving individually each action. It implies that the strategy the agent follows does not change once the agent has found one that works.*

Definition 4.3.2. *An innovative agent is one which tries to explore different strategies, thus usually discovering many ways to achieve a goal, but not intensively exploiting any of them.*

Then according to the definitions, an adaptive agent does not need to look for a different policy —once it has one that works— since it can reach its goals. But an innovative agent will not use the information it gained about the best policies to exploiting the environment and yield good results. An equilibrium between them is always desirable.

Lemma 1. *For an adaptive agent, any change in the policy will yield worse performance than no change at all.*

Proof. An adaptive agent A_i is adaptive because the sequence of actions is optimal. The proof is naive: for the given agent A_i it holds that $r(s_0, a_0) + \ldots + r(s_k, a_k) + \ldots + r(s_t, a_t) > r(s_0, a_0) + \ldots + r(s_k, a'_k) + \ldots + r(s_t, a_t)$, $\forall a'_k \neq a_k, k \in [0, t]$, otherwise A_i would not have an optimal sequence of actions, thus would not satisfy the definition for an adaptive agent. □

Lemma 2. *Adaptive agents tend with probability* $\rightarrow 0$ *to explore as time* $\rightarrow \infty$

Proof. Since adaptive agents are the ones with the lowest *KAI* index, their behavior is to stay in the policy they are following. They will not change policies (from Lemma 1). □

Theorem 1. *Innovative agents detect changes more probably than adaptive ones.*

Proof. A short intuitive proof by contradiction: if an agent is able, always, to detect new actions available it means the agent is always trying to move to different states in the environment; otherwise the agent would not *see* those actions. For an agent to move to all possible states in the environment it is needed a high innovative behavior, regardless of the performance of its actions. Thus they could never behave adaptively, not even for a short period of time, since in this short period of time some new actions could appear and disappear in some states of the environment.

More formally: a change —visible in a state s— can occur with a probability $p(s|C, E)$ given the a priori probability of a change C and the environment E. For an innovative agent the probability of visiting a state

s' is $p(s'|E)$, depending on the topology of the environment. For an adaptive agent the probability of visiting a state s'' is $p(s''|E) = 1$, $\forall s'' \in \pi$, 0 otherwise, according to the definition for an adaptive agent. Thus, unless $p(s|C, E) << p(s''|C, E), s \notin \pi, s'' \in \pi$ —which would mean that changes occur, somehow, more probably in the states that belong to the policy π— the net result is that $p(s = s'|E, C) > p(s = s''|E, C), \forall s, s', s'' \in X$, being X the whole set of states. $\square$

Of course, if the set of states X contain only states which are part of the policy of the adaptive agent, the adaptive agent has the same probability than the innovative agent.

And if the changes are distributed more probably in the states which the adaptive agent visits than in the other ones, and the probability of random visiting them is the same, the adaptive agent will have more chances to find a change than the innovative one. But again, that would mean that the changes occur more in the policy being used than in any other state.

5

Social-Welfare RL (SoWelL)

Contents

5.1 Relationship with MDP's

The *Social Welfare Reinforcement Learning* (SoWelL) algorithm proposed here allows an agent to learn an optimal policy in a MAS, as long as the environment stays stationary; but also allows agents to innovate their policies when there is a significant change. The algorithm changes the learning ratio accordingly to the agent model specified in section 4.2. In spirit it is similar to WoLF [Bowling and Veloso, 2002a], but instead of computing the average policy $\bar{\pi}$ to figure out if the agent must learn further, SoWelL bases the probability of exploration calculus —as described before— on the agent's emotional response and the social opinion. As a byproduct, that will allow computation of trust and reputation, since the agents know who gave an opinion different to their own, and at which grade that was.

The goal of the algorithm is keeping the agent adaptive while the environment stays static (with no significant changes) but innovative when a significant change happens, until the society reaches a (local) maximum.

Since the problem can be modeled as a MDP, it seems appropriate to use an algorithm rooted in the RL principles, such as Q–learning or a variant of it. A review of Q–learning was shown in section 2.7, but as a remainder of the challenges pending while using Q–learning it is to remember that algorithms in this family are guarantied to converge to the optimal policy when the agent has explored the environment enough.

By exploring enough it is intended to say to visit every pair <state,action> infinitely often. In practical approaches the latter is sometimes impossible to accomplish, since some states are very hard to reach, or even undesirable by any means (e.g. detection of failure of every actuator in a robot). The problem called *exploration-exploitation dilemma* —which has been addressed before ([Kearns and Singh, 2002, Bowling and Veloso, 2002a])— tries to solve when an agent should explore the environment, or exploit the actions it knows already that will yield the best cumulative reward. The algorithm proposed next does, in some manner, bypass the problem by using a function which is decreasing in time if the environment behaves statically, which, at the end, it is what was intended.

Some algorithms in the RL family allow finite representations of infinite state environments, by means of using the core ideas of a traditional method (e.g. Q–learning) with techniques such as Soft State Aggregation (SSA [Singh et al., 1995]), Case Based Reasoning (CBR) and so on. So from this perspective there would be no problem other than the actual (usually negligible) error induced by these *mappings* between $\mathbb{R}^d$ and $\mathcal{X}$, $d \in \mathbb{N}, \operatorname{card}(\mathcal{X}) < \infty$. For CBR's retrieval stage the same techniques can be applied, providing a more powerful solution if the reuse stage is implemented conveniently.

5.2 Mechanism for adaptability

Recalling what was said before, the agent is learning the functions which govern the behavior of the environment; it cannot distinguish between what it hasn't learned yet and what has changed. But what the agent can do is to compute a difference between the expected value for its action and the observed one. It will not help to detect a change in a *true or false* manner, but will give a degree of adaptiveness of the agent to the environment.

On the other hand, the KAI index (which is the probability of exploration

k[1]) has to be computed from the emotion *response* of each agent. Thus, since every agent has its own different personality, there will be a bias in the outcome of both the emotion response and, consequently, in the exploration probability. This is a convenient fact that will allow the MAS to niche their agents in a faster way than without the emotions in the agents.

Two of the emotions will be directly linked to the computation of the KAI index k: the joy/distress of the agent, noted by $\mathcal{J}$, and the hope/fear, noted by $\mathcal{H}$. The functions which compute both of them are shown below:

$$\mathcal{J} = \frac{Q_{t+1} - Q_t}{|\max(Q_{t+1}, Q_t)|} \tag{5.1}$$

Notation: Q_t means the discounted reward the agent has learned so far at time–step t for the given state and the taken action, such as $Q_t \equiv Q_t(s_t, a_t)$. The normalization is done by the maximum in the pair Q_{t+1}, Q_t; then $\mathcal{J} \in [-1, +1]$.

$$\mathcal{H} = -\frac{Q_{t+1} - Q_{t-1}}{2\max(|Q_{t+1}|, |Q_{t-1}|)} \tag{5.2}$$

The hope/fear emotion is computed as the *uncertainty* of the variability of the Q-values, based on the first discrete derivative of the Q-function.

The next step is to define the social opinion function, R_Σ:

$$R_\Sigma = \frac{\mathcal{J}\mathcal{H} + \mathcal{J}}{2} \tag{5.3}$$

This definition of R_Σ (figure 5.1) allow an agent to give a positive opinion when it *feels* joy ($\mathcal{J} \to 1$) and feels no fear ($\mathcal{H} \to 1$). The agent will give a negative opinion when it feels distress and has fear. The values of R_Σ are shown in table 5.1 when $\mathcal{J}$ and $\mathcal{H}$ take values in the limit cases $\{+1, -1\}$.

$\mathcal{J}$	$\mathcal{H}$	R_Σ
+1	+1	+1
+1	-1	0
-1	+1	-1
-1	-1	0

Table 5.1: Results of computation of R_Σ in the limit cases $\{+1, -1\}$ for $\mathcal{J}$ and $\mathcal{H}$

[1]Every parameter belongs only to one agent. Subscripts have been omitted for the sake of clarity; e.g. when it is said k it really means k_{A_i}, for the agent A_i. That holds for every parameter in the section.

Recalling equation (5.3) and table 5.1 it is observable that this agent is somewhat pessimistic. It will only give positive feedback when it feels joy and hope. As the joy tends to distress, it will start giving negative feedback, accordingly to the hope —or certainty— that it has about the future. It is to notice that R_Σ resembles one of the emotional pairs described in the OCC model: the satisfaction/fears-confirmed pair. The resemblance goes into the semantic level —how satisfied is this agent with the current actions in the current environment— as well as into the variables needed to the emotional response —the events and objects needed to compute them are the same—. This resemblance may induce a change in a future model to re-write R_Σ as one of the $\mathcal{P}$ functions: the one that computes satisfaction/fears-confirmed. This way the personality, which is invariant to an agent, will include how the agent treats its emotional stimuli to expose an opinion, without the need of a different function, which is more intuitive: we usually say —for a person— "she is optimistic. That goes with her personality", hence the cause resides *in* the personality.

The social opinion, weighted by the reputation and trust of each of the agents, will be used by κ function to compute the KAI index. The total social welfare —noted as σ for short— is computed as:

$$\sigma = \sum_{A_i} \rho_{A_i} \tau_{A_i} R_{\Sigma A_i} \tag{5.4}$$

for every agent A_i, including itself. Of course different agents will have

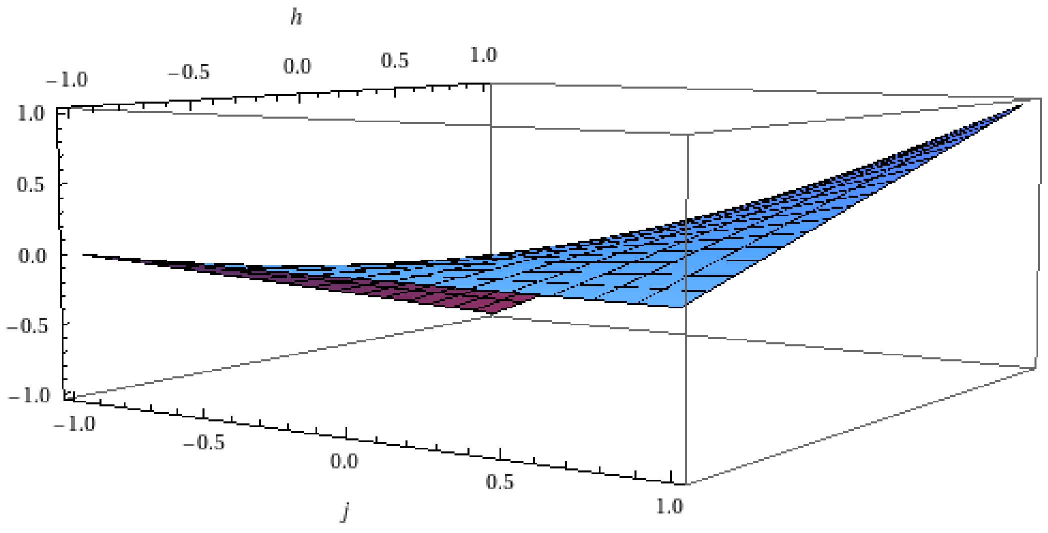


Figure 5.1: Function R_Σ plotted. Its behavior in the limits is more clearly shown in table 5.1

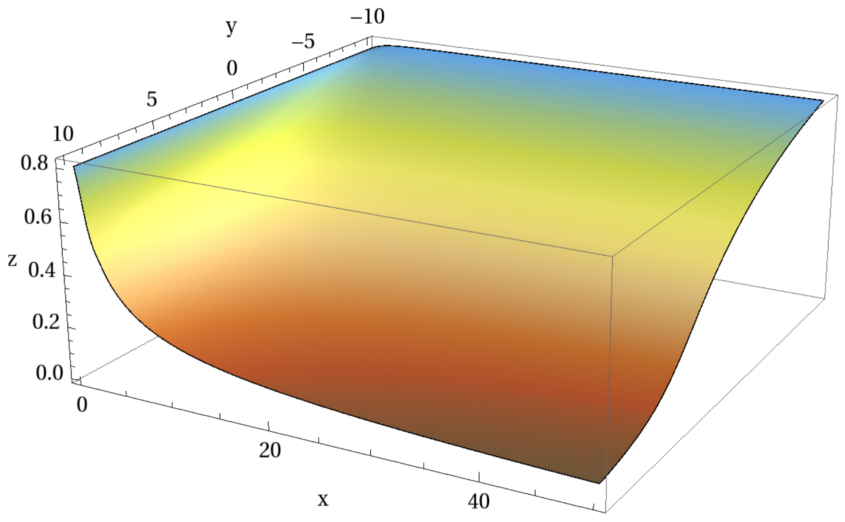


Figure 5.2: Function κ as a time series k_n. X-axis: n; Y-axis: σ; Z-axis: k_n

different values for ρ (reputation) and τ (trust). It must hold that $0 \leq \rho, \tau \leq 1$, $\forall A_i$.

For the KAI index calculation —which we can see as the exploration probability—, carried out by the κ function, we propose a time series in which both the last value of κ and the total social welfare σ from equation (5.4) are used:

$$\kappa \equiv k_{t+1} = \frac{k_t}{k_t + 2^{k_t(\sigma-2)}} \tag{5.5}$$

The series is similar to a sigmoid when centered around $\sigma = 0$ (see figure 5.2), but converges differently. The κ series is not very sensible to the initial value k_0, as long as $k_0 \in]0, 1]$ (with $k_0 = 0$ it holds that $k_i = 0$, $\forall i$, regardless of the values of σ).

5.3 Convergence of Learning

Convergence of the learning process is proven under some assumptions:

- The agent can represent the universe without taking into account other agents. Al least the agent has function of representation which can represent the universe with no ambiguity with a high probability, probability that increases with time.

- The agent knows the set of actions that are available at any time.
- The agent can observe the response of the system to its actions.

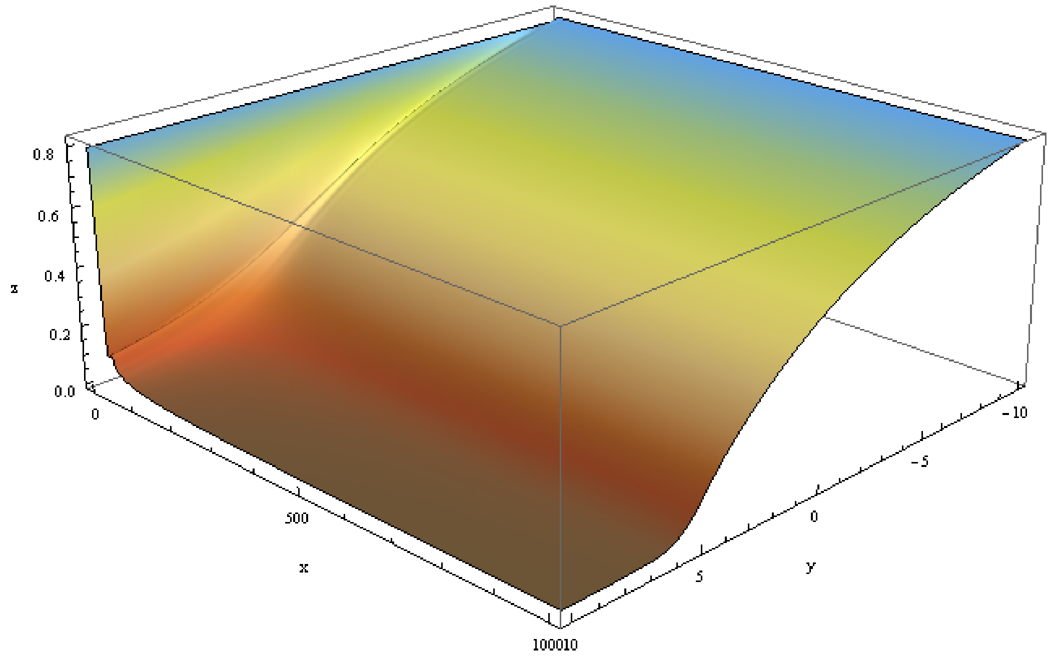


Figure 5.3: Function κ. X-axis: n; Y-axis: σ; Z-axis: κ. The iterations are now 1000, and it starts to take the shape of the function it converges to.

To prove that the algorithm converges, the k_t series has to converge as well —necessary condition—. In figure 5.3 κ is depicted using a bigger number of iterations (1000), which *indicates* the possibility of the series to converge to a function. Being the k_t series convergent, the Q-learning algorithm will behave as usual, thus converging to the optimal policies (under the former assumptions) if all the states are visited enough.

$$k_{t+1} = \frac{k_t}{k_t + 2^{k_t(\sigma-2)}} \tag{5.6}$$

Lemma 3. *k_{t+1} converges for $t \to \infty$*

Proof. In the limit when $t \to \infty$ we expect $k_{t+1} = k_t$, thus

$$\begin{aligned}
k_t &= \frac{k_t}{k_t + 2^{k_t(\sigma-2)}} \\
1 &= k_t + 2^{k_t(\sigma-2)} \\
1 - k_t &= 2^{k_t(\sigma-2)} \\
1 &= (1-k_t)2^{-k_t(\sigma-2)} \\
2^{\sigma-2}(\sigma-2) &= 2^{\sigma-2}(\sigma-2)2^{-k_t(\sigma-2)}(1-k_t) \\
2^{\sigma-2}(\sigma-2) &= 2^{\sigma-2-k_t\sigma+2k_t}(1-k_t)(\sigma-2) \\
2^{\sigma-2}(\sigma-2) &= 2^{(1-k_t)(\sigma-2)}(1-k_t)(\sigma-2) \\
ln(2)2^{\sigma-2}(\sigma-2) &= e^{(1-k_t)(\sigma-2)ln(2)}(1-k_t)(\sigma-2)ln(2) \\
(1-k_t)(\sigma-2)ln(2) &= W(ln(2)2^{\sigma-2}(\sigma-2)) \\
1-k_t &= \frac{W(ln(2)2^{\sigma-2}(\sigma-2))}{(\sigma-2)ln(2)} \\
k_t &= 1 - \frac{W(ln(2)2^{\sigma-2}(\sigma-2))}{(\sigma-2)ln(2)} \qquad (5.7)
\end{aligned}$$

□

In equation (5.7) the $W(x)$ stands for the Lambert-W function such that for every number $x \in \mathbb{R}$, $x = W(x)e^{W(x)}$. Equation (5.7) shows that the proposed κ as a time series converges to the function depicted in figure 5.4.

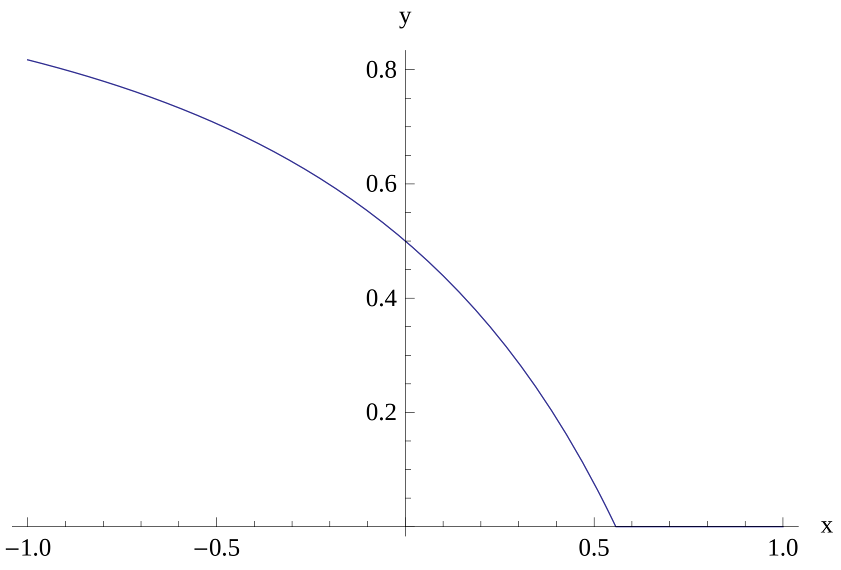


Figure 5.4: κ converges to this function. In the X-axis the social opinion σ

This function is not symmetric with respect to the X–axis. The behavior of the convergence function is desirable from the psychological point of view: it is easier for the agent to adapt than to innovate. The first derivative of that function would show that it is steeper when $\sigma > 0$ (σ is in the X–axis) than in the case of $\sigma < 0$. That means that the agent tends to adapt more easily than innovate. Of course, the main principle that started the design of the κ function is still preserved: negative values will move the agent towards the innovative side, and positive ones towards the adaptive side.

Theorem 2. *The SoWelL algorithm proposed converges when $t \to \infty$*

Proof. The proof of the convergence of the learning algorithm follows intuitively from the proof of Q-learning convergence [Watkins and Dayan, 1992]: The agents will learn a policy converging to Q^* while exploring. With enough time, the Q values will converge; thus their opinion will be positive, making them explore less and less. It is possible, though, that the values obtained are optimal only locally, when the exploration time wasn't enough to make them visit other states very often. In short, given that the Q–values converge, and the κ function, which is the exploration probability, converges as well when the Q–values do, the whole algorithm does. $\square$

The *personality* $\mathcal{P}$ —or transformation of the observable— and the KAI calculus function κ can make the algorithm non-convergent. I.e. if an agent has a transformation which result is to expose an opinion that is not related to the Q values, convergence is not probable. These two functions —particularly κ— must be designed taking into account convergence.

6 Tests and Validation

Contents

6.1 Study Case

To study the effects of social learning on a practical level we have proposed a simple collaborative game: a market economy in which the production line has different types of units, and the agents should build some or consume certain products for their survival. The agent enters the system with certain amount of money and life points; these values hold for every agent entering the system. In this scenario the agent competes with other for survival, or by obtaining as many points as possible. But it can collaborate to build some of the intermediate products, and earn money to use it later. Agents with life below 0 disappear: it is considered by the system that the agent is unable to produce more than what it consumes. This is called a *failure* of an agent

—it failed to be productive—. In the event of a player disappearance, the market automatically removes all products which were unsold and belong to that player. If the agent reenters the market, it spawns with the initial values of *life* and *money*. The market hosts the products that the agents build. Any product may have n dependencies of n_i, $i \in [0, p]$ units of m different products, and need some specific time to be built.

Finally, the price of the products is fixed by each one of the agents.

To get a product from the market the player must pay in advance. Negative balances are not allowed, but agents are able to modify the price of the products they built even when they are already on the market.

In this environment the actions of other players are not observable, but its results may be: they cannot tell if an agent is making a product, but they notice —if they are willing to— that the agent built something when it puts the product on sale on the market.

The system —the market— may change the production rules of the system at any time. The variations allowed are the creation and deletion of product types, changes on the dependencies —both the amount and the type— and the modification of the base time needed for production. Those will be what we called *mechanical changes* in section 3.1.

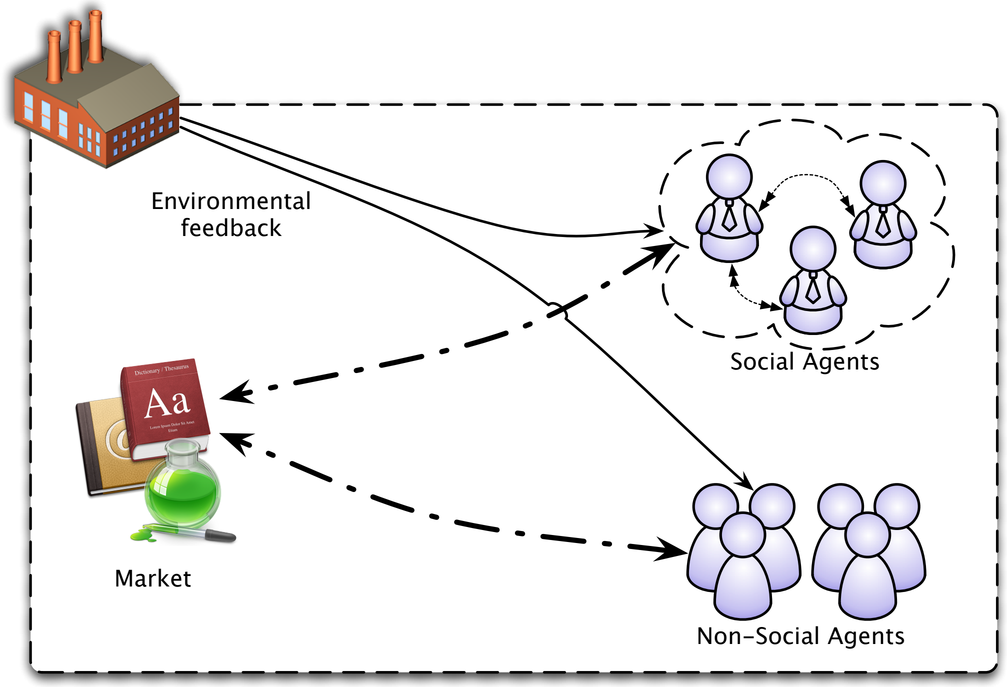


Figure 6.1: Overview of the Economic Market Simulator mini–Framework

The *social changes* encloses the possibility of agents entering and leaving the market at any time.

This framework is illustrated in figure 6.1. The environment can simulate supply chain processes, where demand can vary and rules can be altered. This way, a good agent would innovate and adapt to the variations automatically, and —with some time— come back with a good behavior competing and collaborating with other agents inside the environment.

6.2 Simulation

Reinforcement learning in a continuous domain environment cannot be applied unless some conditions are given. For the simulation the agents use a well-known technique for grouping states (Soft State Aggregation, or SSA) which allow them to represent an infinite space of continuous values, with D dimensions, in a D-dimensional finite space (clusters) [Singh et al., 1995]. For the study case the environment state space is seen as a D-dimensional matrix, grouping the states ($\in \mathbb{R}^D$) in an exponential way. The discrete position (or index of the cluster) i for the continuous variable x is computed as:

$$i = \min(\lfloor log_2(x+1) \rfloor, \omega) \tag{6.1}$$

Bounding i on some number (ω) which is to be considered near to the maximum that is to be seen for that variable, for D dimensions that are taken into account for the environment representation. Therefore the total set of states which every agent must represent is $\omega^D \times |A|$. The number of actions $|A|$ is known by every agent (although it may change with time).

Using SSA the *retrieval* phase of the CBR system allows to choose cases "similar" because they share a small distance between them in terms of the grouping defined by the clustering function (6.1). This function is convenient for its simplicity, although much better representation could be achieved by using some other metric which would allow a probabilistic outcome —the way usually SSA is defined is by this kind of metrics—.

The *reuse* and *revise* stages are carried out by using the algorithm shown in section 5.2. The algorithm, grounding in Q–learning, is always *retaining* the outcome of the solution, so the solution applied to the situation that the agent is facing is always recorded —implicitly— as well as its outcome —explicitly by recording it to estimate the Q–function—.

Two different types of agents will be differentiated; those who can receive and understand social opinions —executing Social-Welfare RL— and those which not —using standard Q–learning—. The social network between the social agents is established as a fully connected graph. Scalability may not be a problem, since there are many different ways to communicate in this fully connected graph by using communication artifacts: similarly as the jabber protocol can do, the communication can be done by using an amount of messages in the order of $O(n)$ messages, being n the number of participants. Issues of how convenient is to have *messaging services* such as the latter proposed are beyond the scope of this work —trust issues, delays and so on—.

Learning of the Q-table was done individually for both the social-aware agents and the non-social. Parameter ω was fixed at 5, which means that the last state for each dimension will represent values of $x \in [15, +\infty]$. The dimensions (or variables) taken into account to represent the environment are the number of products of each kind in the market plus agent's *life* and *balance*.

Three types of products where loaded: *Wheat*, with no dependencies and needing 1 cycle. *Flour*, requiring 2 units of *flour* and 2 cycles. And *Bread*, eatable, providing 10 units of life, requiring 2 units of *flour* and 2 cycles.

The actions available to the agents are the creation of any kind of the products, plus another one called *eating*. The total space of representation needed by each agent is only of $(5)^{(3+2)} \times 4 = 12500$ states. Learning stage has taken $1,000,000$ cycles, with a probability of exploration $k = 1$, hopping to explore as many states as possible, as many times they could. Every agent starts with exactly the same Q-matrix at the beginning of the simulation. The experiment includes three different scenarios, thus there are two significant changes. The scenarios are:

1. No changes in the environment.

2. Relaxing the production rules: *Flour* needs 1 unit of time and 1 unit of *Wheat*; *Bread* needs 1 unit of time and 1 unit of *Flour*.

3. Hardening the production rules: *Wheat* needs 2 units of time; *Flour* needs 2 units of time and 3 units of *Wheat*; *Bread* stays as originally, needing 2 units of time and 2 units of *Flour*.

The scenarios data is summarized in table 6.1. The column "Total Time" refers to the needed time to manufacture that product; e.g. in scenario 1, for the product *Flour*, it is needed 4 units of time because the dependencies demand 2 units of wheat —2 units of time for making that wheat— plus the required time to *assemble* the flour: in total 4 units of time.

As a remark, it is important to control the value of κ —to avoid machine precision problems— such that never reaches 0. For that matter, the

Case	Product	Time	Dependencies	Total Time
Scenario 1	Wheat	1	0	1
	Flour	2	2–Wheat	4
	Bread	2	2–Flour	10
Scenario 2	Wheat	1	0	1
	Flour	1	1–Wheat	2
	Bread	1	1–Flour	3
Scenario 3	Wheat	2	0	2
	Flour	2	3–Wheat	8
	Bread	2	2–Flour	18

Table 6.1: Overview of the three scenarios

computation is modified as:

$$\kappa = \begin{cases} \epsilon & \text{if } \kappa < \epsilon \\ 1-\epsilon & \text{if } \kappa > (1-\epsilon) \\ \kappa & \text{otherwise} \end{cases}$$

for a given $\epsilon \rightarrow 0$ which is representable by a machine.

It is expected to observe a better adaptation of the social-aware agents compared to the non-social ones, which use traditional reinforcement learning (SSA Q-learning).

6.3 Results

Two different experiments were carried out: the first one with 4 agents in the system, and the second one with 32. Two rounds per experiment were done: one with classic Q-learning agents (using SSA to represent the environment), and another round with social aware RL agents (also using SSA). In both experiments we measured the number of failures and the welfare of the society, for the two types of agents. Both values are normalized per number of agents. In each figure the two vertical lines represent the moment in time when a change in the environment was made.

The *failures* represent agents which were in the system for a time yielding loses instead of producing welfare. It is the equivalent of business having loses in the economy: they start with an initial amount of resources but they have a limited time until they start yielding benefits. Otherwise they would confront bankruptcy. Notice that the lower the number of failures, the better the system as a whole —it means the system has coped with many different agents and few of them had problems with their policies—.

The *welfare* represent the wealth of the society. The higher, the better. The final aim of the system is, actually, to produce as much wealth as possible. The agents, which do not cooperate explicitly, face the problem of concurrency in the environment, hence the fact that the double the number of agents does not mean the double of wealth.

From the data in figure 6.2 we see that social aware agents are able to cope with the changes better than the others. The algorithm keeps agents innovative until they reach a (local) maximum; after this, the agents start adapting their behavior to the new conditions. Nevertheless, we see in figure 6.3 that the welfare per agent in the social-welfare society is not much higher than the one in the non-social. The agents have not find a good policy. Despite the fact that the number of failures is low, their achievements are not too good. We will talk about this in a moment after examining the other experiment.

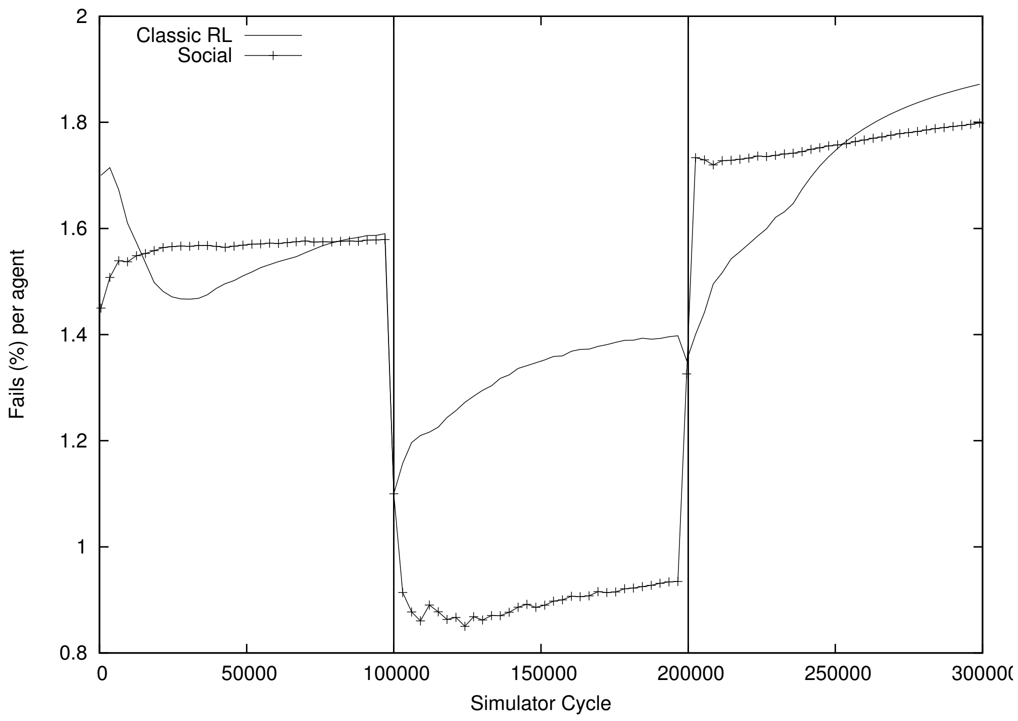


Figure 6.2: System with two significant changes, failures per agent comparison. 4 agents in the system.

In figure 6.4[1] it is shown that social agents have fewer failures than non-social ones. As expected in the theory, the more agents the system has, the

[1]The Y-axis is in logarithmic scale

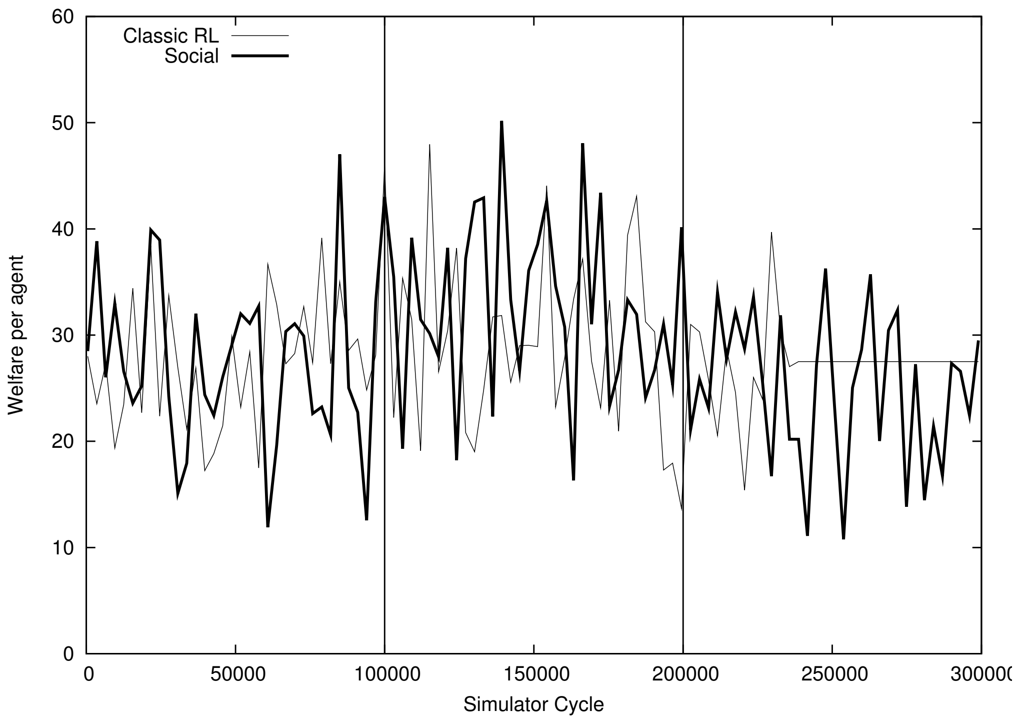


Figure 6.3: System with two significant changes, welfare per agent comparison. 4 agents in the system.

easier is to notice changes; but also when there are many agents, it is more probable that they interfere with each other, which would make harder to find a policy in a short period of time —policies get more complicated as agents are added in the society—. That explains also the data shown in figure 6.5, where a big difference between the two types of agents appear. When having more agents in the system it is easier to have them spread out of the different states, so when the majority agree that they should behave adaptively, it is more probable that the different states of the system have been already explored.

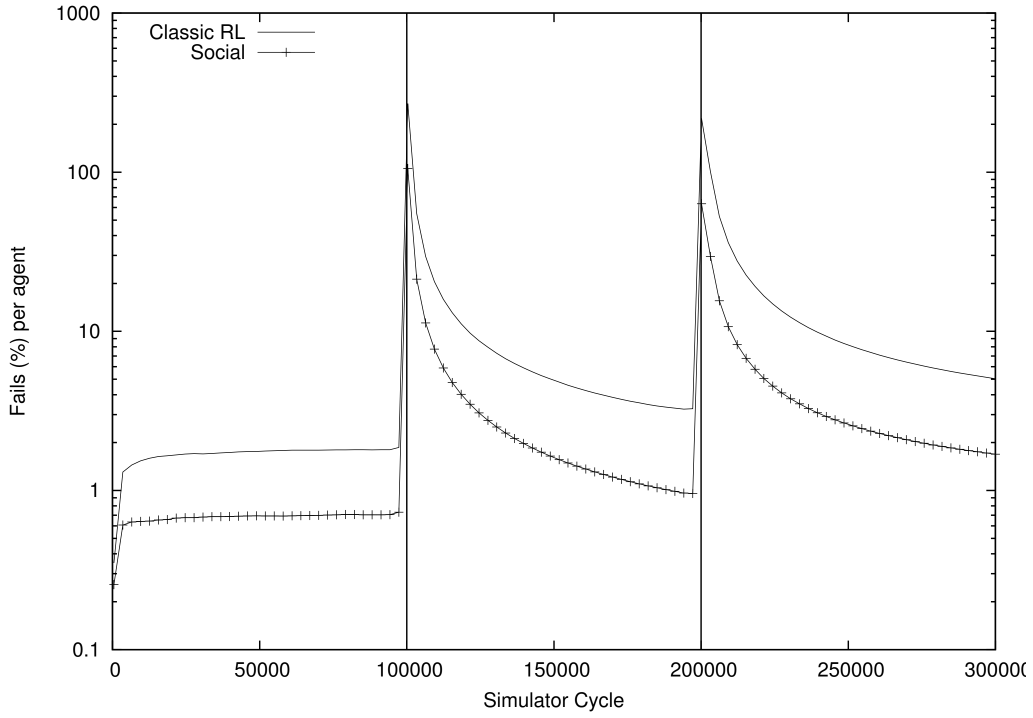


Figure 6.4: System with two significant changes, failures per agent comparison. 32 agents in the system.

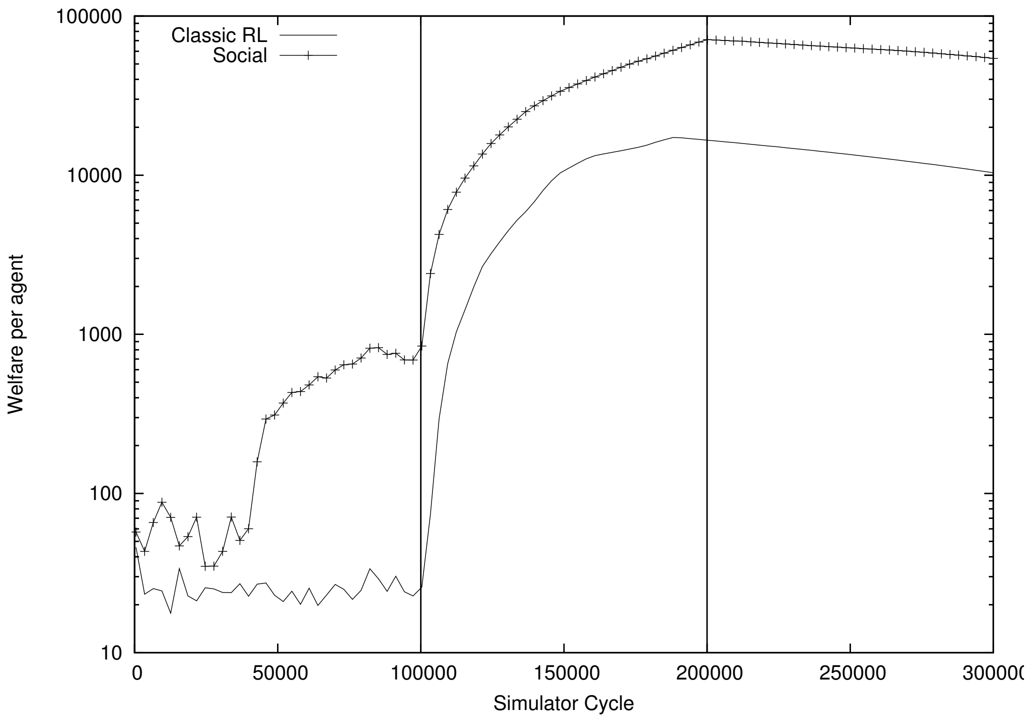


Figure 6.5: System with two significant changes, welfare per agent comparison. 32 agents in the system.

7

Conclusions and Future Work

Contents

7.1 Conclusions

This work has presented a new agent model based on three main topics: the KAI index, emotions and the opinion of the society in which it is immersed. The communication of the opinion inside the society is a good mechanism to trigger innovation when a change requires it. The mechanism works better when the number of agents increases, which is very promising in terms of scalability when the MAS is using SoWelL as the learning algorithm.

The convergence of SoWelL has been proven, but the agents can converge towards a non-global optimum. Where this may be a problem for some environments, it can be alleviated with agents which are even more pessimistic than the one showed in this work —being more pessimistic would move the KAI index towards innovation faster, thus exploring the environment longer—.

The decision of whether the society should innovate or adapt is carried out by the Social Opinion. The agents cope with the problem by locally sensing the opinion of their social network and making a local decision. The aggregated opinion is a global indicator of the decision that the society —in average— will do: either adapt or innovate.

7.2 Future Work

Some further work should be carried out to integrate the social welfare mechanism in algorithms such as WoLF or one of its variants.

Also tests in zero-sum games would clarify whether this mechanism works well in pure competitive games. Some scenarios —rock, paper, scissors; matching pennies; etc.— are a good benchmark in zero-sum games.

Related to this line of research is the possibility of computing the similarity of goals of two agents given their social opinions. Intuitively we can say that when two agents have similar opinions, their goals are similar, with increasing probability as time increases. This similarity would be semantically similar to a composition between trust and reputation between both agents, but will allow what we just sketched in section 5.1: having a degree of similarity. In order to have agents with different goals sharing the same environment it is crucial to carry out this computation. Some ideas are to use graph spectral analysis, using an opinion distance function as the weight of the arc between two agents.

A thorough study of which types of functions are suitable for the 11 dimensions in the personality could be interesting. Of course there are some which would not work at all, and some which should work very well, depending on both the problem and the functions that other agents are using. It would be interesting to see when and why incompatibilities —if there will be any— can arise.

Further work could be developed studying the dynamics of the system between changes. Although converge is guarantied, the dynamics of how the agents would see their policies affected are not clarified in this work. Also a mechanism to extend this social opinion to temporal difference learning would be desirable.

Probably the most ambitious work would derive from the fact that agents could start communication by using environmental mechanical objects. In the model presented there was a distinction between mechanical and social actions, as well as their environments associated, but could the communication be achieved in a higher level by using only mechanical actions? That question, where very appealing, needs to be refined and cautiously examined.

Of course, methods for faster convergence of Q–learning or other RL algorithms —such as temporal differences— would come in handy to this approach. Also distance functions between cases which could adapt the importance of variables could lead to a better model and algorithm.

Functions for the reuse phase which could map back the learned policy for the observed input have not been used in this work. A full study of how to do this mapping —which in some sense is the *inverse* mapping observed in the retrieval phase— would be constructive. Finding a good set of mapping functions for the retrieval and reuse phase, although is usually problem dependent, would make the system a much more powerful one.

A

Related Publications

The related publications can be divided in two parts: the first one dealing with CBR's modeling and systems based on CBR models; and the second one dealing with multi–agent reinforcement learning algorithms for adaptiveness.

J. A. García-Pardo, S. Heras, R. Ramos-Garijo, A. Palomares, V. Julian, M. Rebollo and V. Botti. *CBR-TM: A New Case-Based Reasoning System for Help-Desk Environments.* In *17th European Conference on Artificial Intelligence (ECAI-06) Vol. 141 pp. 833-834. (2006).* (Core A)

S. Heras, J. A. García-Pardo, R. Ramos-Garijo, A. Palomares, V. Julian, M. Rebollo and V. Botti. *CBR Model for the Intelligent Management of Customer Support Centers.* In *7th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL-06) Vol. 4224 pp. 663-670. (2006).* (Core C)

S. Heras, J. A. García-Pardo, M. Rebollo, V. Julian and V. Botti. *Intelligent Customer Support for Help-Desk Environments.* In *Hybrid Artificial Intelligence Systems pp. 73-80. (2007).* (Core C)

S. Heras, J. A. García-Pardo, R. Ramos-Garijo, A. Palomares, V. Botti, M. Rebollo and V. Julian. *Multi-domain case-based module for customer support.* In

Expert Systems with Applications Vol. 36 N. 3 pp. 6866-6873. (2009). (JCR 2.908, ranking 15 of 103 in *Computer Science*)

V. Sánchez-Anguix, J. A. García-Pardo, A. García-Fornes and V. Julian. *Towards soccer simulation as a testbed for adaptive systems and agreement technologies.* In *8th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2010) Vol. 71 pp. pp19-27. (2010).*(Core C)

J. A. García-Pardo, J. Soler and C. Carrascosa. *Social Reinforcement Learning for Changing Environments.* In *2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-10) pp. pp269-272. (2010).* (Core C)

J. A. García-Pardo, J. Soler and C. Carrascosa. *Social Conformity and Its Convergence for Reinforcement Learning.* In *Multiagent System Technologies Vol. 6251 pp. 150-161. (2010).* (Core B)

Bibliography

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59.

[Akchurina, 2009] Akchurina, N. (2009). Multiagent reinforcement learning: algorithm converging to nash equilibrium in general-sum discounted stochastic games. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 725–732.

[Allingham, 2002] Allingham, M. (2002). *Choice theory: A very short introduction.* Oxford University Press.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming.* Princeton University Press.

[Botti et al., 1999] Botti, V., Carrascosa, C., Julián, V., and Soler, J. (1999). Modelling agents in hard real-time environments. In *MAAMAW'99 Proceedings*, volume 1647 of *LNAI*, pages 63–76. Springer-Verlag.

[Bowling, 2000] Bowling, M. (2000). Convergence Problems of General-Sum Multiagent Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000), June 29-July 2, 2000, Stanford University*, page 89. Morgan Kaufmann.

[Bowling and Veloso, 2002a] Bowling, M. and Veloso, M. (2002a). Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250.

[Bowling and Veloso, 2002b] Bowling, M. and Veloso, M. (2002b). Scalable learning in stochastic games. In *AAAI Workshop on Game Theoretic and Decision Theoretic Agents*, pages 11–18.

[Castelfranchi, 1995] Castelfranchi, C. (1995). Guarantees for autonomy in cognitive agent architecture. *Intelligent Agents: Theories, Architectures, and Languages*, 890:56–70.

[Chamberlain, 2010] Chamberlain, G. P. (2010). *Understanding Strategy.* Createspace.

[Clark, 2001] Clark, A. (2001). Unsupervised Language Acquisition: Theory and Practice.

[Damasio and Sutherland, 1995] Damasio, A. and Sutherland, S. (1995). *Descartes' error: Emotion, reason, and the human brain.* Picador.

[Fabregat et al., 2008] Fabregat, J., Carrascosa, C., and Botti, V. (2008). A social reinforcement teaching approach to social rules. *Communications of SIWN*, 4:153–157.

[Franklin and Graesser, 1996] Franklin, S. and Graesser, A. (1996). Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages.* Springer-Verlag.

[Gold et al., 1967] Gold, E. et al. (1967). Language identification in the limit. *Information and control*, 10(5):447–474.

[Helleboogh et al., 2007] Helleboogh, A., Vizzari, G., Uhrmacher, A., and Michel, F. (2007). Modeling dynamic environments in multi-agent simulation. *Auton. Agents Multi-Agent Syst.*, 14(1):87–116.

[Holland, 1999] Holland, J. (1999). *Emergence: From chaos to order.* Basic Books.

[Hu and Wellman, 2003] Hu, J. and Wellman, M. (2003). Nash Q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069.

[Huhns and Singh, 1998] Huhns, M. and Singh, M. P. (1998). *Readings in Agents. Chapter 1*, pages 1–24. Morgan Kaufman.

[Jennings and Wooldridge, 1998] Jennings, N. R. and Wooldridge, M., editors (1998). *Agent Technology. Foundations, Applications and Markets.* Springer-Verlang.

[Kearns and Koller, 1999] Kearns, M. and Koller, D. (1999). Efficient reinforcement learning in factored MDPs. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 740–747. Citeseer.

[Kearns and Singh, 2002] Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232.

[Kirton, 1976] Kirton, M. (1976). Adaptors and innovators: A description and measure. *Journal of Applied Psychology*, 61(5):622–629.

[Kirton, 1989] Kirton, M. (1989). *Adaptors and innovators: Styles of creativity and problem-solving.* Routledge.

[Lopez De Mantaras et al., 2005] Lopez De Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., et al. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(03):215–240.

[Martinez-Miranda and Aldea, 2005] Martinez-Miranda, J. and Aldea, A. (2005). Emotions in human and artificial intelligence. *Computers in human behavior*, 21(2):323–341.

[Mataric, 1994] Mataric, M. (1994). Learning to behave socially. In *From Animals to Animats: International Conference on Simulation of Adaptive Behavior*, pages 453–462. MIT Press.

[Melo and Ribeiro,] Melo, F. and Ribeiro, M. Coordinated learning in multiagent MDPs with infinite state-space. *Autonomous Agents and Multi-Agent Systems*, pages 1–47.

[Mitchell, 1997] Mitchell, T. (1997). Machine learning. WCB. *Mac Graw Hill*, page 368.

[Nwana, 1996] Nwana, H. S. (1996). Software agents: An overview. Technical report, Intelligent Systems Research. AAT, BT Laboratories, Ipswich, United Kingdom.

[Oatley et al., 2006] Oatley, K., Keltner, D., and Jenkins, J. (2006). *Understanding emotions.* Wiley-Blackwell.

[Ortony et al., 1988] Ortony, A., Clore, G. L., and Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge University Press.

[Ovum, 1994] Ovum (1994). Intelligent agents: the new revolution in software. Technical report.

[Ralston et al., 1993] Ralston, A., Reilly, E., Hemmendinger, D., and (Firm), X. (1993). Encyclopedia of computer science.

[Russell and Norvig, 1995] Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall International Editions.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial intelligence: a modern approach*. Prentice hall.

[Sargent, 1992] Sargent, P. (1992). Back to school for a brand new abc. *The Guardian*, (12 March, p. 28).

[Shoham and Leyton-Brown, 2008] Shoham, Y. and Leyton-Brown, K. (2008). *Multiagent systems: algorithmic, game-theoretic, and logical foundations*. Cambridge Univ Pr.

[Singh et al., 1995] Singh, S. P., Jaakkola, T., and Jordan, M. I. (1995). Reinforcement learning with soft state aggregation. In *Advances in Neural Information Processing Systems 7*, pages 361–368. MIT Press.

[Steunebrink et al., 2007] Steunebrink, B. R., Dastani, M., and Meyer, J.-J. C. (2007). A logic of emotions for intelligent agents. In *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pages 142–147. AAAI Press.

[Sutton and Barto, 1990] Sutton, R. and Barto, A. (1990). Time-derivative models of Pavlovian reinforcement. *Learning and computational neuroscience: Foundations of adaptive networks*, pages 497–537.

[Van Parunak and Odell, 1999] Van Parunak, H. and Odell, J. (1999). Engineering artifacts for multi-agent systems. Invited Speech at the MAAMAW'99 Workshop.

[Von Neumann and Morgenstern, 1967] Von Neumann, J. and Morgenstern, O. (1967). *Theory of games and economic behavior*. John Wiley & Sons Inc.

[Waldrop, 1992] Waldrop, M. (1992). Complexity: The emerging science at the edge of order and chaos. *New York*.

[Watkins and Dayan, 1992] Watkins, C. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

[Wilkinson, 2006] Wilkinson, D. (2006). *The ambiguity advantage: what great leaders are great at.* Palgrave Macmillan.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152.