Using XCS to Build Adaptive Agents

Zahia Guessoum[⋆†]
[⋆]OASIS
Laboratoire d'Informatique de Paris 6 (LIP6)
Zahia.Guessoum@lip6.fr

Lilia Rejeb[†]
[†]LERI
Université de Reims
rejeb@poleia.lip6.fr

Olivier Sigaud[⋆]
[‡]OASIS
Laboratoire d'Informatique de Paris 6 (LIP6
Olivier.Sigaud@lip6.fr

Abstract

To deal with dynamic changes of their environment, agents need an adaptive mechanism. This paper proposes an integration of classifier-based framework (named XCS) and an agent-based framework (named DIMA). The result of this integration is an adaptive- agent framework. It has been applied to simulate economic models.

1 Introduction

Dynamic and complex systems, such as economic markets, are characterized by a large number of agents and a dynamic environment. To deal with the dynamic and unexpected variations of their environment, adaptive agents are very useful. Several learning-based multi-agent systems have therefore been realized (see Kazakov et al. (2001), Kudenko and Kazakov (2002) and Kazakov et al. (2003). The proposed solutions can be classified in two categories:

- Single agent learning: Agents can learn independently of other agents. Every agent is thus a simple learning algorithm and its environment is often static.
- Multi-agent learning: Each agent is endowed with a learning algorithm to build a model of its environment. The latter includes other agents.

Most realized works in multi-agent learning deal with real-life applications. The proposed solutions are thus often *ad hoc*, they cannot be easily reused to build other real-life applications. So, we still need works on generic adaptive agent models. The purpose of this project is to propose a generic adaptive agent framework (named XCS-Agent). This framework is the result of the integration of an agent-based framework (named DIMA (Guessoum and Briot (1999))) and a Learning Classifier System (LCS) (named XCS (Wilson (1995))). The application of XCS-Agent to simulate economic models has allowed to highlight the advantages of the proposed framework and to underline the open problems (coding complex environments, exploration/exploitation, ...).

This paper is organized as follows: Section 1 presents the example, Section 2 describes the framework XCS-Agent, Section 3 studies the exploration/exploitation problem, and Section 4 gives an overview of the realized experiments to validate XCS-based agents.

2 Example

The considered application is the simulation of an economic model. In this application, we consider a set of firms in competition with each other within a shared market. A firm is defined by the following main parameters:

- K, the amount of capital available,
- B, the R&D budget,
- the state variables (X vector) represent the different types of resources (funds, people, equipment, ...) owned by the firm,
- the Y variables represent the performances of the firm. They are directly influenced by the X vector,
- the strategy the firm follows to allocate its resources,
- the associated organizational form. An organizational form is an abstract entity that gathers a set of similar firms. Similar firms have similar behavior and similar structure (see Baum and Rao (1999) and Guessoum et al. (2003)).

Moreover, a firm is characterized by its decision process which aims to select the most suitable strategy in a given context. This context includes the internal parameters (K, B, X, ...) and the firm's perception of the other firms (their K, their B, their Y, ...). Several solutions may be used to represent this decision process such as inference engine, case-based reasoning and LCSs (see Holland et al. (2000)). However, the use of classifiers is more suitable to the dynamic and unexpected variations of economic markets.

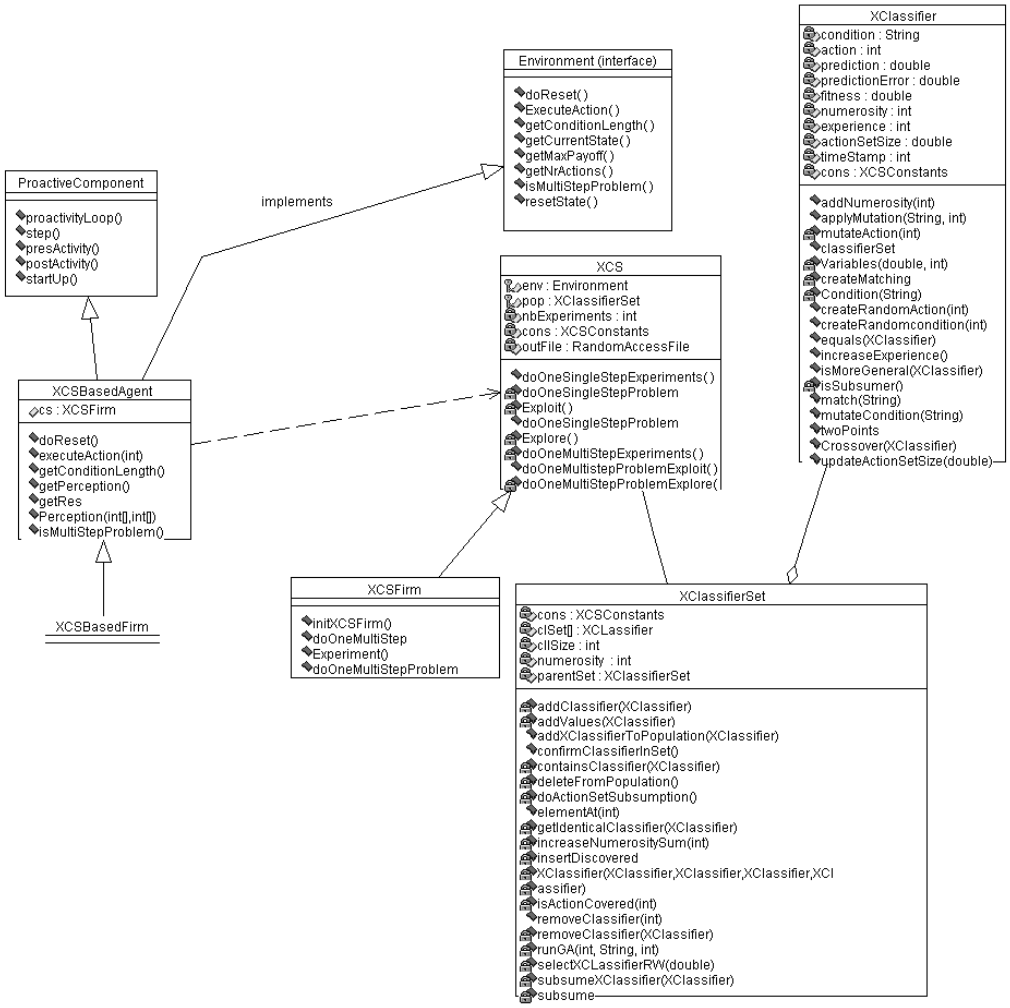

Figure 1: Overview of the framework XCS and XCS-Agent

3 XCS-Based Agent Model

This section presents first an overview of XCS, it then describes the XCS-Based agents.

3.1 Overview of XCS

XCS is a recent LCS (Wilson (1995)) which can solve complex learning problems. It is based on a standard classifier condition-action rules. Each classifier is characterized by three parameters: prediction p, error e, fitness f. A condition corresponds to a chain of 0, 1 and #. # represents 0 or 1, the associated attribute is not taken into account when checking the condition. The parameters p, e and f are automatically updated according to the reward obtained by the application of the chosen action.

In a context s, a step of XCS executes the following actions (see Table 1):

- scan the environment (define the state of the environment),
- execute a step by using the exploration or exploitation strategy.

XCS provides a set of generic classes which can be reused to implement LCSs (see Figure 1). To implement a LCS, one has to implement the Environment Interface.

XCS has been reused to build XCS-Agent. The agents context and their behavior are described in the following sections.

3.2 Agent Context

In a classifier, the condition represents the context of the agent. It is defined by its local parameters and its perception of the environment. For instance, the firm's context includes:

- the capital, the resources, the budget,
- a representation of the competition (information on the other firms),

Table 1: An example of method of the XCS step

```
private void doOneSingleStepProblemExplore (String state, int counter){
XClassifierSet matchSet= new XClassifierSet(state,pop,counter,env.getNrActions());
PredictionArray predictionArray= new PredictionArray(matchSet, env.getNrActions());
int actionWinner= predictionArray.randomActionWinner();
XClassifierSet actionSet = new XClassifierSet(matchSet, actionWinner);
double reward = env.executeAction(actionWinner);
actionSet.updateSet(0, reward);
actionSet.runGA(counter,state,env.getNrActions());}
```

- a representation of the organizational forms which is defined by the resource variations and the performances of the associated firms. Each variation of a resource is described by a symbolic value (small, medium, large). In our experiments, we use the same fuzzy granulation (Zadeh (2001)) for the various resources (see Guessoum et al. (2003)).

The various attributes of a firm are not binary. We have thus decomposed the definition domain of each attribute `i` in `n` intervals. An attribute can be thus coded by a binary string of `n` bits which indicate the corresponding interval. However, the decomposition of the definition domain into intervals is not easy and the performances of a firm rely on this decomposition. Indeed, if the interval boundaries do not fit with the natural boundaries of an optimal strategy, the adaptive agent cannot perform optimally (see Section 5).

An action corresponds to a strategy of the firm (see Section 2). An example of classifier is given in Table 2.

Table 2: Example of classifier

condition
K ∈ [-300,100],
B ∈ [0,100],
X[1] ∈ [2,5], ..., X[8] ∈ [1,3]
AverY[1] ∈ $[3, 20]$, ..., $AverY[3]$∈ [0,3]
action
strategy1,
parameters
P = 0.5 ,
e = 0.01,
F=100.

The capital intervals are: $[-300, 100]$, $[101, 300]$, $[301, 500]$, $[501, 600]$, $[601, 800]$, $[801, 1000000]$. In the condition of the given example (Table 2), it is represented by 7bits: 1000000. Each bit indicates if the value belongs to the corresponding interval.

The reward corresponds in our model to the aggregation of the variation of performances which result from the application of the chosen strategy. It is calculated by the following formula:

$$r = agreg(\frac{Y_t[1] - Y_{t-1}[1]}{Y_t[1]}, \frac{Y_t[2] - Y_{t-1}[2]}{Y_t[2]}) \quad (1)$$

where `agreg` is an aggregation operator.

3.3 Agent Behavior

The used agent framework is DIMA (described in Guessoum and Briot (1999)). DIMA is a framework of proactive components representing autonomous and proactive entities. It is illustrated by a minimal set of classes and methods defining the main functionality of a proactive component. This functionality may be extended in the subclasses. This framework is mainly composed of the class ProactiveComponent (see Table 1) which describes:

- The goal of the proactive component, it is implicitly or explicitly described by the method `isAlive()`.
- the basic behaviors of the proactive component, a behavior is a sequence of actions that allow to change the internal state, to perform a message or to send a message to other proactive components. Each behavior is implemented as a java method of this class.
- the meta-behavior defines how the behaviors are selected, sequenced and activated.

The step of a firm is defined in Table 4:

Table 4: Step of an agent

```
public void step() {
updateCompetitionRepresentation();
getProfitVariation();
updateBudget();
%% begin decision process
budgetRest=applyStrategy(chooseStrategy());
%% end decision process
updateCapital();
caculatePerformances();
updateMarket(); }
```

Table 3: Main methods of ProactiveComponent

Methods	Description
public abstract boolean isAlive()	Tests if the proactive component has not yet reached his goal.
public abstract void step()	represents a cycle of the meta-behavior of the proactive component.
void proactivityLoop()	Represents the meta-behavior of the proactive component. public void proactivityLoop() {while (this.isAlive()) { this.preActivity(); this.step(); this.postActivity();}}

public void startUp()	Initialize and activate the meta-behavior.
	
	public void startUp() {
	this.proactivityInitialize();
	this.proactivityLoop();
	this.proactivityTerminate();}

XCSBasedAgent (see Figure 1) is defined as subclass of ProactiveComponent and implements Environment. An XCS is associated to each XCSBasedAgent and its meta-behavior uses the XCS step (methods doOneMultiStep*). The step of an adaptive firm is defined in Table 5.

Table 5: Step of an adaptive firm

```
public void step() {
updateCompetitionRepresentation();
getProfitVariation();
updateBudget();
%% start decision process
cs.doOneMultiStepExperiment(3);
%% end decision process
updateCapital();
caculatePerformances();
updateMarket(); }
```

4 Exploration/Exploitation

LCSs must find a good compromise between two complementary strategies: exploration and exploitation. When the uncertainty in the current prediction is high, the system should better explore than exploit (see Wilson (1996)). An adaptive agent should be able to observe its behavior and choose the most suitable strategy according to its experiences. To deal with that problem, we introduce meta-rules which allow to adapt the exploration/exploitation rules to the evolution of the context and the state of the classifier set according to the variations of the firm performances. These meta-rules are mainly based on two parameters:

- m: the number of steps during which an agent uses exploration,
- n: the number of steps during which an agent uses exploitation,

After each m exploration steps, the system executes n exploitation steps. It executes then these meta-rules:

- if the Perf(t+n) $=<$ Perf(t) then the system must still learn, the number of exploitation steps is then decreased (n =n/2)
- if the Perf(t+n) $>$ Perf(t) then the system has learned enough, the number of exploitation steps is then increased (n =n*2).

They are simple and adapt the behavior of the LCSs to the evolution of the agent environment. They provide thus a good solution to the Exploration/Exploitation dilemma.

5 Experiments

The proposed framework was tested on the simulation of economic models (see Section 2). We first compared the XCS-Based firms and firms that use *a priori* defined rule-based systems. We considered two populations of firms: rule-based firms and classifier-based firms. We injected in each population one XCS-based firm and we observed the performances and the number of classifier of this firm. The considered parameters are:

- A population size =800
- A #_probability = 0.5
- a learning rate (b)=0.2
- a crossover Rate=0.8
- mutation rate = 0.02
- qGA =25
- minimum error = 0.01

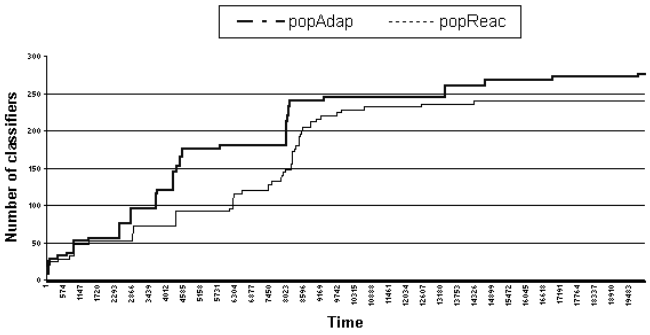

Figure 2: Convergence of the number of classifiers

The experiments show that the convergence of the classifier number within a population of non adaptive firms is easier (see Figure 2). In fact, in adaptive-firm population, the firms need a lot of time to learn and construct their classifier populations. These results are, nevertheless, sensitive to some initial values of the parameters of the XCS such as the learning rate. The learning coefficient beta is important in LCSs. Its default value, in XCS, is 0.2. To show the influence of this parameter, we realized experiments with different learning rates. Figure 3 shows that the reduction of this value improves the convergence.

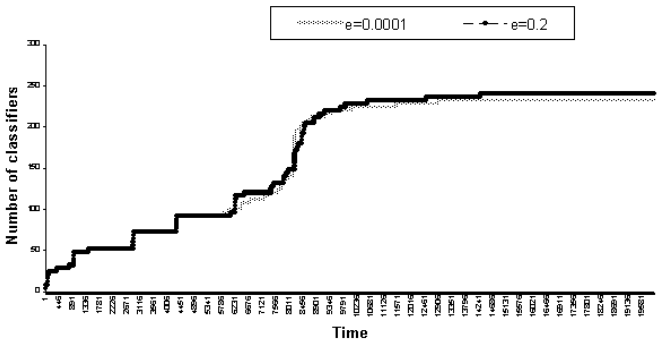

Figure 3: Comparison of the convergence of adaptive firms using different learning coefficients

We set then this learning rate to 0.0001 for the rest of the experiments. We study also the effect of the representation on the convergence of the firms. We use for this two populations: 1) in the first population, the classifier representation is based on a representation on 8 intervals and 2) in the second one, the classifier representation is based on 16 intervals. Figure 4 shows that the more precise representation allows easier convergence. So in the rest of these experiments, we use 16 intervals.

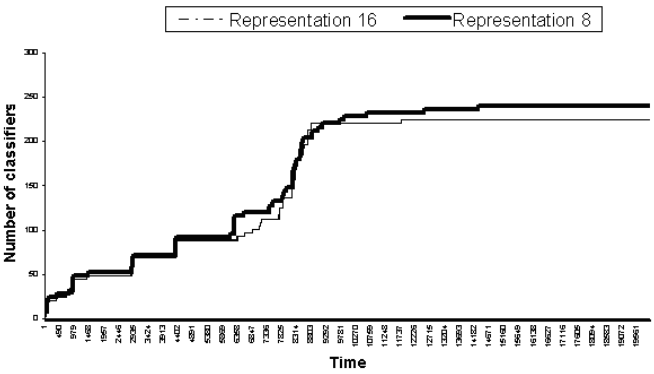

Figure 4: Comparison of the convergence of populations with different representations

In the second series of experiments, we studied the Exploration/Exploitation problem. We considered three populations of 500 XCS-based firms with different strategies: exploration, exploitation, and meta-rules (Explore-Exploit).

The results (see Figure 5) show that the population with meta-rules (the black one in the Figure) has slightly higher performances. We note that the difference is not important (5 %). These meta-rules are then a good technique but more experiments are needed to find the adequate parameters such as m.

6 Conclusion

This paper presented a new XCS-based agent framework and its application to simulate economic models. The latter are dynamic and complex systems. This application showed the advantages of using LCSs in dynamic multi-agent environments. Large-scale multi-agent systems provide thus very good applications to validate LCSs, but also very challenging ones, given the continuous and non-stationary character of these applications. The first experiments are interesting but more experiments are needed to choose the most suitable parameters and intervals to improve the performances. On that point, using adaptive interval techniques such as the one suggesting comparison of the profit of populations with different strategies (see Wilson (2000)) is a major area for future work. A second perspective of this work is the definition of a methodology to facilitate the development of adaptive-agent systems.

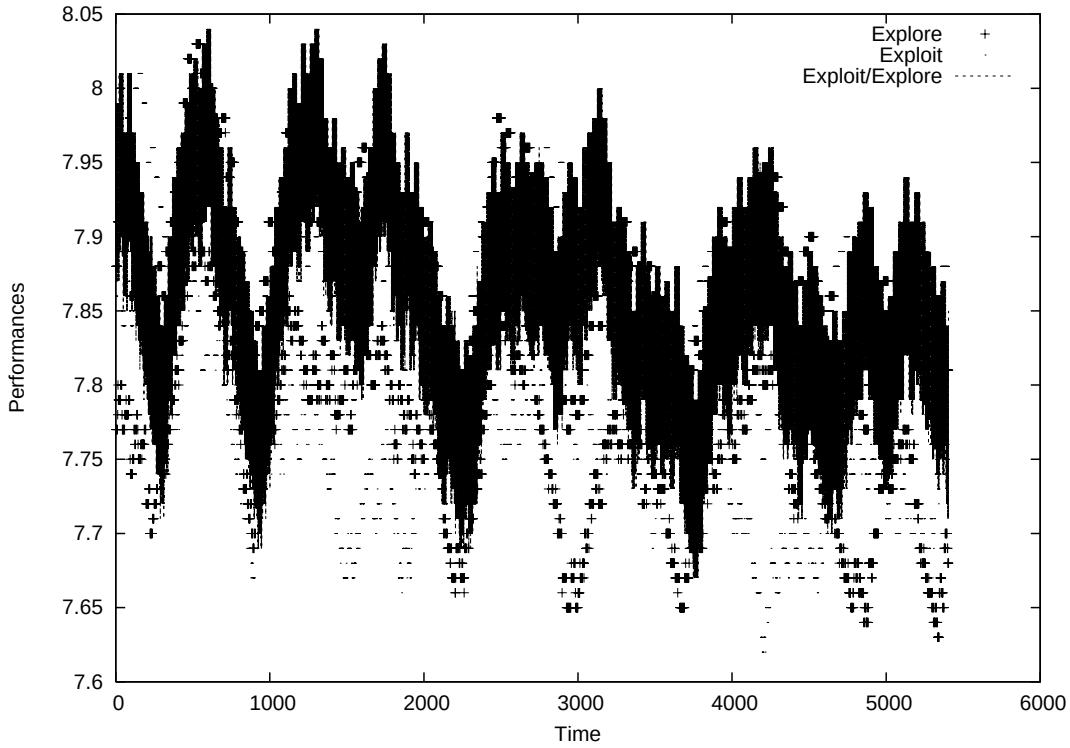

Figure 5: Comparison of the performances of the different populations with different strategies

References

A. Joel A.C. Baum and Hayagreeva Rao. *Handbook of Organizational Change and Development*, chapter Evolutionary Dynamics of Organizational Populations and Communities. Oxford University Press, 1999.

Z. Guessoum and J.-P. Briot. From active objects to autonomous agents. *IEEE Concurrency*, 7(3):68–76, 1999.

Z. Guessoum, L. Rejeb, and R. Durand. Emergence of organizational forms. In *AAMAS'03*, Aberystwyth, UK, April 2003. AISB.

J. Holland, L. B. Booker, M. Colombetti, M. Dorigo, D. E. Godberg, S. Forrest, R. Riolo, R. E. Smith, P.-L. Lanzi, W. Soltzmann, and S. W. Wilson. What is a learning classifier system? in learning classifier systems: from foundations to applications. In P.-L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning Classifier Systems: from Foundations to Applications*, pages 3–32. Springer-Verlag, Heidelberg, 2000.

Dimitar Kazakov, Eduardo Alonso, and Daniel Kudenko, editors. *Adaptive Agents and Multi-Agent Systems*, York, UK, 2001. AISB.

Dimitar Kazakov, Eduardo Alonso, and Daniel Kudenko, editors. *Adaptive Agents and Multi-Agent Systems*, Aberystwyth, 2003. AISB.

Daniel Kudenko and Dimitar Kazakov, editors. *Adaptive Agents and Multi-Agent Systems*, London, 2002. AISB.

S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

S. W. Wilson. Explore/exploit strategies in autonomy. In J. Pollac J.-A. Meyer P. Maes, M. Mataric and S. Wilson, editors, *From Animals to Animats 4, Proc. of the 4th International Conference of Adaptive Behavior*. MA,, Cambridge, 1996.

S. W. Wilson. Get real! XCS with continuous valued inputs. in learning classifier systems: from foundations to applications. In P.-L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning Classifier Systems: from Foundations to Applications*, pages 209–220. Springer-Verlag, Heidelberg, 2000.

L. A. Zadeh. A new direction in ai: Toward a computational theory of perceptions. *AI Magazine*, 22(1):73–84, 2001.