

# Using XCS for Action Selection in RoboCup Rescue Simulation League

Ivette C. Martínez  
Grupo de Inteligencia Artificial  
Universidad Simón Bolívar  
Caracas 1080-A, Venezuela  
martinez@gia.usb.ve

David Ojeda  
Grupo de Inteligencia Artificial  
Universidad Simón Bolívar  
Caracas 1080-A, Venezuela  
david@gia.usb.ve

Ezequiel Zamora  
Grupo de Inteligencia Artificial  
Universidad Simón Bolívar  
Caracas 1080-A, Venezuela  
ezequiel@gia.usb.ve

## ABSTRACT

This paper presents a team of agents for the RoboCup Rescue Simulation League problem that uses an evolutionary reinforcement learning mechanism called XCS, a version of Holland's Genetic Classifiers Systems, to support the agents' decision process. In particular, we use this mechanism to decide the number of ambulances required to rescue a buried civilian and the number of Fire Brigades necessary to extinguish a fire. We also analyze the problems implied by the rescue simulation and briefly describe our solutions for every identified sub-problem using multi-agent cooperation and coordination built over a subsumption architecture. Our classifier systems were trained in different disaster situations. Trained agents outperformed untrained agents and most participants of the 2004 RoboCup Rescue Simulation League competition. This system managed to extract general rules that could be applied to new disaster situations.

## Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning—*knowledge acquisition*; I.6.3 [Simulation and Modeling]: Applications

## General Terms

Design, experimentation

## Keywords

RoboCup Rescue Simulation League, Action Selection, XCS, ERL

## 1. INTRODUCTION

RoboCup Rescue has become a standard problem for the artificial intelligence, intelligent robotics and multi-agents communities. In particular, the RoboCup Rescue Simulation League (RCRSL) problem, has proven to be an excellent environment for AI and Machine Learning software testing.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO '06, July 8–12, 2006, Seattle, Washington, USA.  
Copyright 2006 ACM 1-59593-186-4/06/0007 ...\$5.00.

This project started as a response to the Japanese rescue team's inability to deal with natural disasters observed in the Hanshin Awaji Earthquake in 1995 [8]. Its main goal was to create an heterogeneous multi-agent system for disaster prevention and mitigation and victim search and rescue in a simulated environment.

Tadokoro *et.al.* [8] define RoboCup Rescue Simulation League as a *semi-optimal behavior planning problem with extremely complex constraints having widely time-varying multiple objectives*, these constraints include limited time for decision making, constantly changing conditions and partial information. Additionally, communication is limited, and *there are sometimes incorrect rumors* [8]. In this work we try to solve the most important RCRSL's challenges: victim rescue and fire extinction.

Usually large state-space problems are managed through generalization techniques such as neural networks and other function approximators, *which allow compact storage of learned information and transfer of knowledge between "similar" states and actions* [2]. In order to manage this large state-space problem under RCRSL time restrictions, we use an accuracy-based evolutionary reinforcement learning mechanism called XCS [11]. This mechanism not only does an accelerated search, but also is able to derive general rules. In particular, the XCS decides how many Ambulance Teams are required to make an effective rescue of a victim buried in debris of a building and how many Fire Brigades are necessary to extinguish a fire. We decided to focus on these decisions because they require the analysis of a large amount of scattered information and are critical to the agents' performance.

Evolutionary reinforcement learning (ERL) is an approach to reinforcement learning inspired by Darwin's theory of evolution. Evolutionary algorithms can find satisfactory solutions in large state-spaces at a low computational cost. *Methods from genetic algorithms, evolutionary programming, genetic programming, and evolutionary strategies could all be used in this framework to form effective decision making agents* [4].

We compared our agents with other successful teams from the 2004 competition and obtained satisfactory results. We believe this technique is able to process the pertinent information from the environment and give the appropriate output to solve the victims rescue and fire extinction problems. Additionally we give a short description of our RoboCup Rescue decomposition into sub-problems and the approach we used for each one of them.

## 2. XCS

The XCS classifier systems [9], as well as Holland’s Learning Classifier Systems (LCS) [1], are domain independent adaptive learning systems. Their main distinguishing features are the base of classifier fitness on the accuracy of classifier reward prediction instead of the prediction itself, and the use of a niche genetic algorithm, i.e., a Genetic Algorithm (GA) that operates on a subset of the classifier population.

XCS is an example of a rule-based ERL that searches over a policy space. The advantage of XCS from the ERL point of view is its generalization capacity. For this reason, the XCS should be able to scale to more complex problems, in contrast with the RL traditional algorithms [11].

## 3. DESIGN

In this section we present the results of our analysis and the decomposition of RCRSL into sub-problems. We use a hybrid approach for decision making, i.e., some decisions are centralized while others are taken by platoon agents. Therefore, platoon agents can take decisions with slight relevance but center agents must decide the most important matters.

### 3.1 Problem Classification

We divided RCRSL sub-problems into four categories: common problems, fire extinction, rubble cleansing and victim rescue. Now we describe some of the identified problems and their solutions.

#### 1. Common problems:

**Civilian search.** Our agents look for civilians in all the buildings in the city. Each platoon must do this job when there is no other higher-priority task to do. All agents have a “world model” which they share in every turn to avoid visiting an already explored site and to provide center agents with the necessary information to make decisions.

**Route planning.** The problem of finding optimal routes between two points in a city was solved by reducing the state space, specifically using *LongRoads* as proposed by ResQ Freiburg [3].

**Communication.** We designed and built a communication protocol based in tokens which uses the message space as much as possible and gives preference to high-priority information.

#### 2. Victim rescue:

The Ambulance Center must decide which victim is going to be rescued next, the number of ambulances that will be sent to the rescue site, which ambulances must go and which one will take the victim to the refuge.

The *next victim selection* algorithm we are using is based on the strategy proposed by the *Damas Team* [6], in order to minimize future casualties considering the next rescue.

The number of ambulances for each victim is determined using an XCS classifier system whose structure and parameters are explained in Section 3.2. Once the number of ambulances is fixed, the nearest ambulances are sent to the rescue. The nearest ambulance of all takes the victim to the shelter.

3. **Fire extinction:** The Fire Station Agent decides which fires to extinguish. At first it selects the 3 most centered fires and uses an XCS to decide how many fire brigades are going to be sent to each fire and sends the nearest units (see section 3.2.2 for Fire Extinction XCS details). We consider a fire as a group of burning buildings relatively close to each other and build them using clustering techniques. When an agent gets to the fire, it chooses which building is going to put out by itself. Each fire brigade decides on its own when to refill its water tank.

4. **Rubble cleaning:** The Police Office prioritizes the roads to be cleaned taking into account their location and traffic.

**Cleaning requests:** If a platoon agent (other than Police Forces) needs to go through a blocked area, it sends a cleaning request to the Police Station, who assigns a police force agent to clean it.

### 3.2 Description of Genetic Classifiers

As we mentioned in Section 2, we use XCS genetic classifiers to support our decision making. In particular we decide how many ambulances are required to rescue a victim and how many Fire Brigades are going to extinguish a fire using this kind of system. The center agents gather information from the platoons and other center agents and decides what platoons should do.

#### 3.2.1 XCS Design for Victim Rescue

Our classifier system takes into account the following attributes: *health points*, *damage*, *buriedness* and *world time*. Each classifier contains 24 bits as shown in Table 1.

Bits 0 to 7 contain the ratio of the victim’s health points to his damage, bits 7 to 14, his degree of buriedness, and the other 6 bits from the input represent the simulation time in order to inform the system about how much time can be spent to rescue the victim. The conversion between integer and binary representation is accomplished by creating predefined ranges.

Each classifier has 3 output bits that represent the number of ambulances that will be sent to rescue the victim.

The reward assigned to each decision taken by the XCS was positive if the victim was rescued and negative if the victim died or wasn’t rescued by the end of the simulation.

#### 3.2.2 XCS Design for Fire Extinction

The fire extinction XCS takes input from three different fires at a time and make a decision on the number of Fire Brigades necessary to extinguish each fire in a single query. The pieces of information taken into consideration are: fire areas, number of unburned buildings around the fire, distance from center of the map, and the cost of the best route to the nearest refuge.

Table 1 shows how information for each fire is given to the XCS, using a 20-bit string composed of 4 fields of 5 bits each. The output for each input fire is a 4-bit string representing the number of ambulances that are going to be sent to extinguish the fire.

Finally, each decision is rewarded positively if fire was extinguished and negatively if the fire could not be controlled by the end of the simulation.

Table 1: Classifier Structures	
<b>Ambulance Center Classifier Structure</b>	
<b>Input</b>	21 bits
Victim's hitpoints/damage	8 bits
Victim's buriedness	7 bits
World time	6 bits
<b>Output</b>	3 bits
<b>Fire Station Classifier Structure</b>	
<b>Input</b>	60 bits (20 per fire)
Fire area	5 bits
Unburned buildings	5 bits
Distance to center	5 bits
Cost to refuge	5 bits
<b>Output</b>	12 bits (4 per fire)

Table 2: XCS and Genetic Algorithm Parameters

XCS Parameter	Value
Genetic algorithm probability	0.2
Reinforcement update rate ( $\alpha$ )	0.1
Min error ( $\epsilon_0$ )	0.5
Error Penalty ( $n$ )	5
<b>Covering Parameter</b>	
<i>Don't care</i> bit probability	0.2
Initial prediction	0
Initial error	100
Initial fitness	0
Max population size ( $ [P] $ )	100
<b>GA Parameter</b>	
Replacement algorithm	Elitist
Selection algorithm	8-tournament
Crossover algorithm	One point
Mutation algorithm	One point
Mutation probability	0.02

## 4. EXPERIMENTS AND RESULTS

We implemented our agents in Java over the standard agent development kit for RoboCup Rescue Simulation System (RCRSS), YabAPI [5]. We also developed our own XCS libraries, following Holland's basic framework presented by Wilson [10].

The XCS system was trained using the parameters shown in Table 2. Two sets of rules were generated using the system during its learning phase over two different instances of the city of Foligno<sup>1</sup>. These sets correspond to the victim rescue and fire extinction systems and will be called *Foligno-Rules* on further results. Another couple of sets, called *Kobe-Rules*, were also derived from a similar procedure, but using instances of the city of Kobe.

Each classifier set was initially empty. All new rules were generated by covering or by the evolutive steps of the XCS. After each simulation, rules were stored and used in the next simulation. We analyzed all simulations in order to obtain the percentage of alive agents and destroyed buildings. These data measured the performance of the rescue and extinguish decision systems. We simulated over 900 disaster situations, alternating maps according to the rule set. The analyzed data are shown in Fig. 1.

We selected the trained classifier set that showed the high-

<sup>1</sup>Foligno and Kobe are standard maps included on the RCRSS.

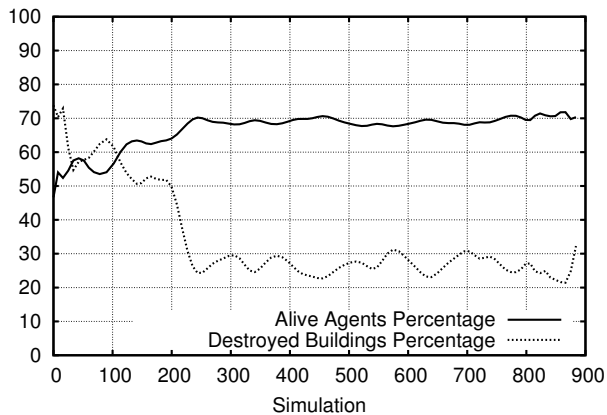


Figure 1: Alive Agents and Destroyed Buildings Evolution over *Foligno-Rules* Training

Table 3: Maps and Rule-Sets for experiments

Map	Trained Rule-Set
FolignoEasy 2005	<i>Foligno-Rules</i>
Kobe (1/10) 2004	<i>Kobe-Rules</i>
RandomEasy 2005	<i>Kobe-Rules</i>

est percentage of alive agents at the end of the simulation. The rules used around the 250<sup>th</sup> simulation for the *Foligno-Rules* training were selected for our experiments.

When the training phase was over, we observed the performance of the classifier systems on three different city maps. The maps and rules selected for the experiments are shown in Table 3.

We also compared our agents using the same decision system with random classifiers. Each group of agents' results is a mean of 20 simulations, except for the results of ResQ Freiburg [3], Damas Team [6] and 5Rings [7], which were extracted from the 2004 competition logs. Fig. 2 shows the results of the experiments on all maps.

In all test cases our agents show better results for rescue operations when using the trained set of rules. These agents achieve higher percentage of alive agents than the ones us-

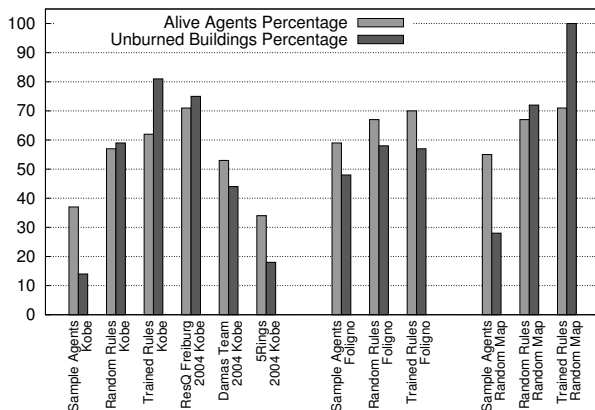
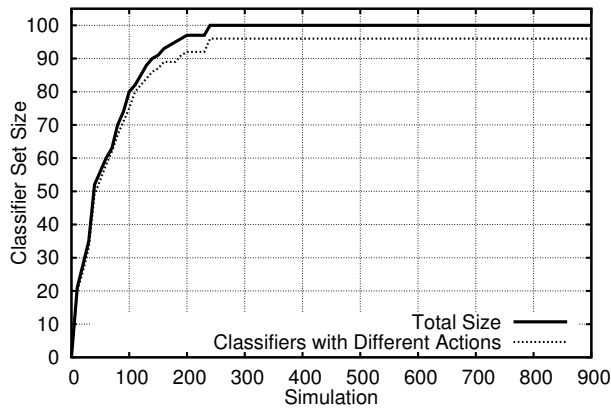


Figure 2: Teams result over Kobe, Foligno and Random maps.



**Figure 3: Size of the classifier set of the fire extinction system over *Foligno-Rules* training**

ing randomly generated rules. This demonstrates that the evolutionary reinforcement learning system tends to refine and keep better rules for the XCS.

Our trained agents also managed to rescue more agents than Damas Team and outperformed the 5rings agents. However, ResQ Freiburg agents can solve the victim rescue problem with better results.

Fire extinction decisions were also notably better than other agents except for the results on Foligno, where randomly generated rules performed better than trained rules. This result suggests that the learning in the training system stalled.

After a detailed analysis of our XCS parameters we realized that the Action Sets generated in each iteration in most cases had only one element. This anomaly is present because the classifier set is significantly smaller than the set of possible actions and this originates a low repetition rate of actions in the classifier set as shown in Fig. 3. These singleton Action Sets suppress the execution of the genetic algorithm of the XCS (since it is impossible to select two classifiers for the GA) and therefore halts the evolution.

## 5. CONCLUDING REMARKS AND FUTURE WORK

This paper presents an approximation to the RCRSL problem that uses an evolutionary reinforcement learning technique, particularly XCS classifier systems. This technique is used to support the decision making of a center agent that coordinates several platoon agents on two complex tasks: civilian rescue and fire extinction.

The design proposed in this paper proved to be an effective solution to the problem and is competitive with other agent teams. Reinforcement learning techniques proved to be a feasible method of extracting general rules that can support decision making in RCRSL. This study found that a trained classifier system provides a good approximation. We obtained sets of rules with a satisfactory degree of generalization: agents trained in one map perform successfully in new environments. Therefore, evolutionary reinforcement learning approaches are appropriate for the RoboCup Rescue domain.

The experiments of this study also pointed out a problem

in parameter selection of the XCS. A maximum population size comparatively smaller than the number of all possible actions, considerably crippled the evolution of the learning system.

The design addressed in this paper showed satisfactory results even considering that the current design of the XCS classifier system for the rescue task is simple. The design of the classifier system for fire extinction decisions must be revised in order to assure that the learning process doesn't stop. An extension of the elements considered by the rules could outperform the current system with a bigger training procedure trade-off.

## 6. REFERENCES

- [1] J. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, University of Michigan Press, 1975.
- [2] L. P. L. Kaelbling and Moore. A. P. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1998.
- [3] A. Kleiner, M. Brenner, et al. ResQ Freiburg: Team description paper and evaluation. *RoboCupRescue simulation league*, 2004.
- [4] D. Moriarty, A. Schultz, and J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research.*, 1999.
- [5] T. Morimoto. YabAPI: API to develop a RoboCupRescue Agent in Java. <http://ne.cs.uec.ac.jp/~morimoto/rescue/yabapi>, 2004.
- [6] S. Paquet, N. Bernier, and B. Chaib-draa. Damas-Rescue description paper. *RoboCupRescue simulation league*, 2004.
- [7] P. T. Silva and H. Coelho. The 5Rings team report. *RoboCupRescue simulation league*, 2004.
- [8] S. Tadokoro, H. Kitano, T. Takahashi, et al. The RoboCup-Rescue project: A robotic approach to the disaster mitigation problem. In *ICRA*, page 4089, 2000.
- [9] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
- [10] S. W. Wilson. Structure and function of the XCS classifier system. <http://web.tiscali.it/LCS/Papers/TUT2.pdf.zip>, 1997.
- [11] S. W. Wilson. Generalization in the XCS classifier system. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, 1998.