# Ambulance Decision Support Using Evolutionary Reinforcement Learning in Robocup Rescue Simulation League

Ivette C. Martínez, David Ojeda, and Ezequiel A. Zamora

Grupo de Inteligencia Artificial
Universidad Simón Bolívar
Caracas 1080-A, Venezuela
{martinez,david,ezequiel}@gia.usb.ve

**Abstract.** We present a complete design of agents for the RoboCup Rescue Simulation problem that uses an evolutionary reinforcement learning mechanism called XCS, a version of Holland's Genetic Classifiers Systems, to decide the number of ambulances required to rescue a buried civilian. We also analyze the problems implied by the rescue simulation and present solutions for every identified sub-problem using multi-agent cooperation and coordination built over a subsumption architecture. Our agents' 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 on new disaster situations, with a computational cost of a reactive rule system.

## 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 problem (RCRSL) has proven to be a excellent environment for AI and Machine Learning software testing.

Tadokoro *et.al.* [9] define RCRSL as a *semi optimal behavior planning problem with extremely complex constraints having widely time-varying multiples objectives*, these constraints include limited time for decision making, limited communication, constantly changing conditions and incomplete and partial information.

In this work we decided to focus on one of the multiple challenges RCRSL offers, the victim rescue problem, which has the strongest impact on the team's performance. This problem depends on various simulation factors. We chose four of them: *world time* and victim *buriedness*, *damage* and *health points*. All this factors have a large feasible domain, which combined generate a very large state-space.

Usually large state-space problems are managed through generalization techniques such as neural networks and other function approximator; *which allow compact storage of learned information and transfer of knowledge between "similar" states and actions* [3].

G. Lakemeyer et al. (Eds.): RoboCup 2006, LNAI 4434, pp. 556–563, 2007.

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]. In particular the XCS decides how many Ambulance Teams are required to make an effective rescue of a victim buried in a building.

Evolutionary reinforcement learning (ERL) is an approach to reinforcement learning that takes advantage of 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* [5].

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 this problem. Additionally we give a short description of our RoboCup Rescue decomposition into sub-problems and the solutions we designed and implemented for each one of them.

## 2 XCS

The XCS classifier systems [10], as well as Holland's Learning Classifier Systems (LCS) [2], are domain independent adaptive learning systems. Its 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 GA that operates on a subset of the classifier population.

The structure of *XCS* rules' conditions are the translation of the conditional part of the logical rules. Rules' actions are binary strings that represent motion actions.

A classifier is a compact representation of a complex set of environment states. Rules have the form $\langle condition \rangle \rightarrow \langle action \rangle$. Conditions are strings of length $l$ in the alphabet $\{0, 1, *\}$. A classifier's condition satisfies a message if its condition matches the input message. A condition $c$ matches message $m$ if and only if: $\forall i, (1 \leq i \leq l) \rightarrow \Pi_i(c) = \Pi_i(m) \vee \Pi_i(c) = '*'$ [1]. Actions are fixed length strings in the alphabet $\{0, 1\}$.

XCS are composed by three subsystems: A performance system, a learning system and a rule discovery system.

The performance system takes an input from the environment, selects an action and transforms it into an output message.

The learning system takes feedback signals from the environment and updates the values of the four parameters that replaces the traditional fitness of LCS: prediction, prediction error, accuracy, and fitness. This change allows a more complete $State \times Actions \rightarrow Prediction$ mapping than traditional LCS.

The rule discovery system uses a GA in order to create new rules. XCS's rule discovery system has two operations: niche GA and covering. The niche GA acts over the Action Set $[A]$, choosing random parents in proportion to the rules'

[1] Where $\Pi_i$ represents the character located at the position $i$ of the string.

fitness. Offsprings are copies of the parents, modified by crossover and mutation. Covering is triggered when the matches set is empty or its mean prediction is a small fraction of the population [P] average prediction. Covering creates a new classifier whose condition matches the current input message and its action is generated randomly.

In the Reinforcement Learning (RL) research, two different approaches have been stated. These two approaches are known as: searching in value function space, and searching in policy space. In the first approach, RL algorithms try to find the optimum value function for the problem. Then, to find the optimal policy given the optimal values function, is immediate. The second approach is to search an optimal policy directly over the space of the policies. For this purpose, evolutionary algorithms are frequently used [5].

This RL approach based on evolutionary algorithms is called Evolutionary Reinforcement Learning (ERL). The ERL algorithms vary in terms of the policies representation method and the fitness evaluation for individual policies.

The two methods of policies' representation are: a chromosome representation and distributed rules-based representations. LCS [2] as well as XCS are examples of a rules-based ERL.

The advantage of XCS from the ERL point of view is its generalization capacity. For this reason, the XCS must be able to scale to more complex problems, in contrast with the RL traditional algorithms [11].

## 3 Design

We present the result of our analysis and decomposition of RoboCup Rescue 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 decision with slight relevance but central agents must decide the most important matters.

### 3.1 Problems Categories

Sub-problems were divided into four categories: Common Problems, Fire Extinction, Rubble Cleaning 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 central agents with the necessary information to make decisions.

   **Route planning:** This problem was solved by implementing the idea of LongRoads proposed by ResQ Freiburg [4].

**Communication:** In order to make information available for as many agents as possible, we designed and built a communication protocol which intends to use the messages as much as possible and gives preference to high-priority information.

2. **Victim Rescue**
We decided to use a centralized approach for this task. 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 takes the victim to the refuge.

   The *next victim selection* algorithm we are using is a based on the strategy proposed by *Damas Team* [6]. It goal is 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 3.3. Once the number of ambulances is fixed, the nearest ambulances are sent to the rescue. If we do not have enough ambulances, all free ambulances are sent and the rest will be sent when they report themselves as freed. The nearest ambulance of all takes the victim to the rescue.

3. **Fire Extinction**
The Fire Station Agent decides which fires to extinguish. It uses a set of fixed rules that chooses how many fire brigades are going to be sent to each fire and sends the nearest units. We consider a fire as a group of burning buildings relatively close to each other. Each fire is built using clustering techniques. Once the agent gets to the fire, it chooses which building is going to put out. Considering that all agents have a similar "world model", it is highly probable that the fire brigades assigned to this fire will choose the same building.

4. **Rubble Cleaning**
We implemented several techniques for choosing the next road to be cleaned, the Police Station sorts police agents to each one at the beginning of the simulation.

   - **Road selection techniques:**
     - Select the nearest LongRoad and clean all its roads.
     - Select the roads belonging to the most frequently used LongRoads.
     - Select the roads around a certain spot of the map.
     - Select the nearest road (only when all LongRoads are passable).
   - **Cleaning requests:** If a platoon agent 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.

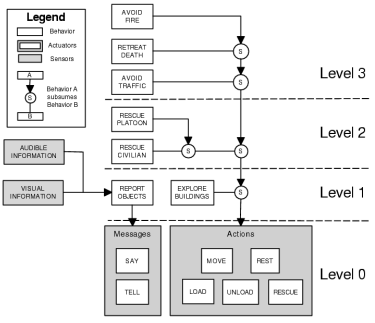



**Fig. 1.** Subsumption Architecture Diagram - Ambulance Teams

### 3.2 Subsumption Architecture

Our agents' conduct was modeled using *Brooks' Subsumption Architecture* [1]. Each kind of agent has different behaviors but they all have the same structure. We decided to sort our agent behaviors in four levels, which we describe next:

**Level 0.** This level contains the most basic behaviors, which are RoboCup Rescue commands and actions.
**Level 1.** This level contains the default behavior of agents such as building exploration and victim search.
**Level 2.** This defines those behaviors that are only activated by a central order such as victim rescue and extinction of fires.
**Level 3.** This is where the highest-priority behavior are located.

Fig. 1 shows the subsumption diagram for ambulance platoon agents. Level 3 behaviors are shared by all platoon agents, Level 2 has the behaviors that entail those agents main tasks, while Level 1 behaviors encode cooperation mechanisms.

### 3.3 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 using this kind of system.

**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 Fig. 2.

| HP/Damage | | | | | | | | Buriedness | | | | | | | World-Time | | | | | | Output | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

**Fig. 2.** Ambulance Center's Classifier Structure

Bits 0 to 7 contain the ratio of the victim's health points to its damage, bits 7 to 14, its 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 used to rescue the victim. The translation 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.

## 4 Experiments and Results

Two different classifier sets were defined in order to train the XCS system using the parameters shown in Table 1. The rules of the first set, called *Foligno-Rules*, were generated using the classifier system on a learning phase over two maps of the same city: FolignoEasy and Foligno. The second set of rules, *Kobe-Rules*, was generated with the same procedure, only changing the maps to Kobe and KobeEasy.

**Table 1.** XCS and GA Parameters

| **XCS Parameter** | **Value** |
|---|---|
| Genetic algorithm probability | 0.2 |
| Reinforcement update rate ($\alpha$) | 0.1 |
| Min error ($\varepsilon_0$) | 0.5 |
| Error Penalty ($n$) | 5 |
| **Covering Parameter** | **Value** |
| *Don't care* bit probability | 0.2 |
| Initial prediction | 0 |
| Initial error | 100 |
| Initial fitness | 0 |
| Max population size($\|[P]\|$) | 100 |

| **GA Parameter** | **Value** |
|---|---|
| Replacement algorithm | Elitist |
| Selection algorithm | 8-tournament |
| Crossover algorithm | One point |
| Mutation algorithm | One point |
| Mutation probability | 0.02 |

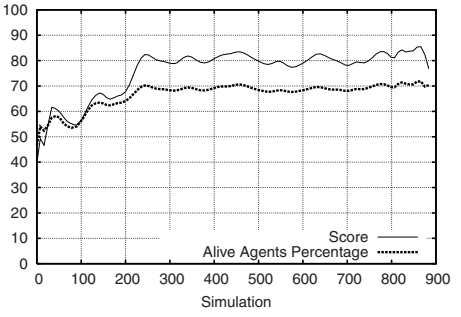
Each classifier set was initially empty. All new rules generated by covering or by the evolutive steps of the XCS. After each simulation resulting rules were stored and used in the next simulation.

We used the percentage of alive agents and the score at the end of each simulation to measure the performance of the decision system for the rescue operations. This procedure was repeated 900 times. The analyzed data are shown in Fig. 3.1.

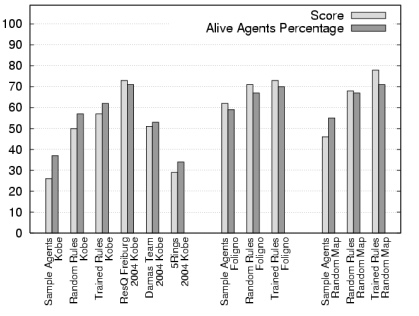
We selected the trained classifier that showed the highest score and alive agents at the end of the simulation. The rules used around the $250^{th}$ simulation for the *Foligno-Rules* training were selected for our experiments.

In order to observe the performance of the classifier system, we selected three maps from different cities and compared the rescue task of our agents using trained rules and random rules. Each group of agents' results are a mean of 20 simulations, except for the results of ResQ Freiburg [4], DAMAS Team [6] and 5Rings [8], which were extracted from the 2004 competition logs.

Fig. 3.2 shows the results of the experiments on all maps. It can be noticed that on all test cases our agents show better results when using the trained set of rules. These agents achieve higher scores and percentage of alive agents than the ones using randomly generated rules. This demonstrates that the evolutionary reinforcement learning system tends to refine and keep better rules for the XCS.




**3.1.** Score and Alive Agents Evolution over *Foligno-Rules* Training




**3.2.** Teams result over Kobe, Foligno and Random maps

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 by using all available Ambulance Teams to rescue a civilian and choosing the rescue order with a GA [4].

## 5 Concluding Remarks and Future Work

This paper presents an approximation to the RoboCup Rescue simulation problem that uses an evolutionary reinforcement learning technique, particularly XCS classifier systems, to support the decision making of a central agent that coordinates several platoon agents on the complicated victim rescue task.

The agents can solve numerous coordination problems presented by RoboCup Rescue using a distributive coordinated search for civilians, as well as road cleaning, and a centralized coordination for victim rescue and fire extinction.

Many ideas and approaches used by our agents are based on previous studies and agent teams. These included informed search using LongRoads, victim selection minimizing future casualties, building clustering, token-based communications, distributed civilian search and road unblocking petitions.

The design proposed by 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 to extract general rules that can support decision making on RoboCup Rescue. In particular, the number of ambulances needed to save a victim depends on several non-predictable factors; this study found that a trained classifier system provides a good approximation at a low

computational cost. We conclude that evolutionary reinforcement approaches are appropriate for the RoboCup Rescue domain.

Even though this paper presents a successful decision support system and a design for a RoboCup Rescue agents team, further development is needed to present these agents on a competition. The agents must be revised in order to assure compatibility with the current competition rules, since our agents were developed and tested with the rules published in 2004.

Parameters of the GA should be examined in future studies. Determination of which mutation and crossover strategies work best with this problem should be considered.

The current design of the XCS classifier system for rescue task is currently very simple. An extension of the elements taken into account by the rules shall outperform the current system with a longer training procedure tradeoff.

We are currently meticulously studying an appropriate design for a XCS that can determine the number of fire brigades needed to control and extinguish a fire.

## References

1. Brooks, R.: How to build complete creatures rather than isolated cognitive simulators. Architectures for Intelligence, 225–240 (1991)
2. Holland, J.H.: Escaping Brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In: Mitchell, Michalski, Carbonell (eds.) Machine Learning: an artificial intelligence approach, Morgan Kaufman, San Francisco (1986)
3. Littman, K.L.P., Moore, A.P: Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 237–285 (1996)
4. Kleiner, A., Brenner, M., Bräuer, T., Dornhege, C., Göbelbecker, M., Luber, M., Prediger, J., Stückler, J.: ResQ Freiburg: Team Description Paper and Evaluation. RoboCupRescue simulation league (2004)
5. Moriarty, D.E., Schultz, A.C., Grefenstette, J.J.: Evolutionary Algorithms for Reinforcement Learning. Journal of Artificial Intelligence Research. 11, 199–229 (1999)
6. Paquet, S., Bernier, N., Chaib-draa, B.: DAMAS-Rescue Description Paper. RoboCupRescue simulation league (2004)
7. RoboCup Rescue Commitee: Robocup 2004 Rescue Simulation League Official Home Page (2004), `http://robot.cmpe.boun.edu.tr/rescue2004`
8. Silva, P., Coelho, H.: The 5Rings Team Report. RoboCupRescue simulation league (2004)
9. Tadokoro, S., Kitano, H., Takahashi, T., Noda, I., Matsubara, H., Shinjoh, A., Koto, T., Takeuchi, I., Takahashi, H., Matsuno, F., Hatayama, M., Nobe, J., Shimada, S.: The RoboCup-Rescue Project: A Robotic Approach to the Disaster Mitigation Problem. ICRA (2000), `http://www.rescuesystem.org/robocuprescue/icra00resc.pdf`
10. Wilson, S.: Classifier Fitness Based on Accuracy. Evolutionary Computation Journal, 149–175 (1995)
11. Wilson, S.: Generalization in the XCS Classifier System Genetic Programming 1998. In: Proceedings of the Third Annual Conference, pp. 665–674. Morgan Kaufmann, San Francisco (1998)