A Hybrid Neuro-Fuzzy Approach to Intelligent Behavior Acquisition in Complex Multi-Agent Systems

ALI AKHAVAN BITAGHSIR*, FATTANEH TAGHIYAREH**, AMIR-HOSSEIN SIMJOUR*, AMIN KHAJENEJAD*

Electrical and Computer Engineering Dep.
faculty of Engineering, University of Tehran
IRAN

Abstract: In recent years, multiagent systems have emerged as an active subfield of Artificial Intelligence (AI). Because of the inherent complexity of MAS, there is much interest in using Machine Learning (ML) techniques to help build multiagent systems. Besides, in these complex systems for which acquiring a mapping from the system's inputs to the appropriate outputs is not simple, the need for a good paradigm for converging the system's functionality to the appropriate goal is apparent. A layered paradigm which is inspired from incremental learning model is proposed. Our approach to using ML and fuzzy logic as tools for developing intelligent firefighter robots involves layering increasingly complex learned behaviors. In this article, we describe multiple levels of learned behaviors, ranging from low level environmental behaviors to more high level and complex behaviors. We also verify empirically that the learned behaviors perform well in disaster situations. Findings suggest that using a hybrid solution comprised from fuzzy logic and artificial neural networks provides us with both robustness and advanced learning ability.
Keywords: Artificial neural networks, fuzzy logic, layered learning, RoboCup Rescue Simulation System (RCRSS), multi-agent systems.

1 Introduction

In recent years, multiagent systems (MAS) have emerged as an active subfield of Artificial Intelligence (AI). Because of the inherent complexity of MAS, there is much interest in using Machine Learning (ML) techniques to help deal with this complexity [2, 3].

RoboCup Rescue is a particularly good domain for studying MAS [11]. The test-bed has enough complexity to be realistic; also good multiagent ML opportunities have brought this domain into a challenging area for MAS researchers.

Our approach to acquisition of intelligent behaviors for fire extinguishment by a team of fire-fighters, is to break down the complexity of decision making step by step, and solving simpler tasks first; going for acquiring higher level team behaviors (strategies), after learning the low level behaviors. This idea is mainly inspired from *Incremental Evolution* (discussed in [4]) which is a method for avoiding limitation of direct evolution in difficult problems where the percentage of the search space that constitutes a solution is very small, and the fitness landscape very rugged. In this case the probability of producing fruitful individuals in the initial random population will be low, and evolution will not make progress; thus the population gets trapped in suboptimal regions of the fitness landscape during the early stages of evolution. One way to scale ML algorithms to tasks that are too difficult to evolve directly, is to begin by viewing the task we want to solve, as a member of a family of tasks, ranging from simple tasks to complex tasks. As tasks get more difficult, the solution set becomes smaller, but because successive tasks are somehow related (depending on the chosen abstraction level for tasks' decomposition), each task positions the population in a good region of the space to solve the next task. Eventually, if the tasks are generated properly, the goal task can be achieved.

A layered paradigm is inspired from Incremental learning model discussed above. Our research focuses on acquiring behaviors for tasks in which a direct mapping from inputs to outputs is intractable. Previously, hierarchical reinforcement learning has been studied and motivated by the well-known "curse of dimensionality" in reinforcement learning (RL). As surveyed in [5], most hierarchical RL approaches use gated behaviors; meaning that there are a collection of behaviors mapping the environment states into low level actions and a gating function decides upon which behavior must be executed [6, 7]. Also MAXQ algorithm [8] and feudal Q-learning [9] learn at all levels of the hierarchy, simultaneously. A constant among these approaches is that the behaviors and the gating function are all control tasks with similar inputs and actions, however in this research the input representation of different layers may be learned previously in lower levels. Moreover none of the above methods has been implemented in a large scale, complex domain. More inline with this type of learning is the work presented by Stone [1]. In their approach a layered model has been tested on RoboCup Soccer server which is a complex domain; three abstraction levels were observed for learning a soccer player robot's behavior. However, in this paper, introducing the abstraction level for learning robots' behavior is done in a different manner. Namely, in our domain of discourse the layers may not necessarily represent robot behaviors. Instead we may learn the *environment*'s behavior (model) in a layer in order to provide more robust decision making in higher levels.

This paper contributes the concrete representation of layered learning in a complex multi-agent domain, namely RoboCup Rescue Simulation System. In section 2 the formalism of our approach is given, discussing about the layered paradigm, formally. A brief specification of simulated RoboCup rescue robots is given in section 3. Our observation of different layers in addition to the implementation phase is demonstrated in section 4. In section 5, the result of our proposed method is discussed; and finally in section 6, we arrive at conclusion and discuss directions for future work.

2 The Layered Learning Paradigm

The layered learning paradigm is designed for domains in which a direct mapping from input representation to output representation is not tractably acquired. Our research involves layering increasingly complex behaviors of both the *rescue robot controller* and the *environment* itself. In this section a formalism much like the one addressed in [1], but with necessary modifications due to the complex dependency of the robot controller to the environment's behavior is presented. The major characteristic of the paradigm is that the output of each layer can have direct effect on **at least** one of the subsequent layers by: (I) supplying the features used for learning; and (II) forming the training example set. Besides, the output of each layer can give us more knowledge about the contribution of previous layers in the final goal; for example we may realize that previous layers are polluted by noisy, irrelevant,..., features which lead us to revise the feature selection process and repeat the layered approach again.

2.1 Formalism

Consider the learning task of acquiring a function f from among a class of functions F which map a set of input features (world state sensory information) I to a set of outputs O, such that based on a set of training examples, f is most likely (of the functions in F) to represent unseen examples and provide the appropriate output. In order to accomplish the task, several layers

$\{L_1, L_2, ..., L_n\}$ are introduced based on the previous knowledge of the designer in the domain of context, complying with the following form:

$$L_i = (I_i, O_i, f_i, (M_i, T_{M_i})) \quad (1)$$

In which:

$\boldsymbol{I_i}$ **:** is the set of inputs (features) selected by the designer; I_i^j for indicates the j^{th} feature acquired directly from the environment or previous layers outputs. Each member of I_1 (I_1^j) is a member of I ; $\boldsymbol{O_i}$: is the set of outputs which may indicate the *environment'*s behavior (model) or the robots appropriate action for the corresponding subtask of this layer (if any). $O_n = O$; $\boldsymbol{f_i}$: is the approximated function which maps I_i into O_i ; $\boldsymbol{M_i}$: f_i is acquired whether by means of a machine learning algorithm or any manual method which may exploit the inherent knowledge of the domain. The method used in this layer is called M_i ; $\boldsymbol{T_{M_i}}$: In case a machine learning algorithm is used as M_i , a set of training examples T_{M_i} is fed into M_i . It is noteworthy that f_i provides one or more inputs for some of the subsequent layers: I_{i+k}^j where $i + k \le n$ and j is an arbitrary value. In the following sections, each layer is described in detail.

3 Robocup Rescue Simulation Environment

Robo**C**up **R**escue **S**imulation **S**ystem (**RCRSS**) is designed to simulate the rescue mission problem in real world [10]. In this simulation system a communication center and a number of simulators are existent to simulate the traffic after earthquake, fire accidents as a result of gas leakage, road blockages, etc [10].

RCRSS environment is a heterogeneous multi-agent system in which the agents correspond to the agents involved in a real rescue mission. Types of agents in this domain are as follows: **1-** ***Fire brigade:*** This agent is responsible for extinguishing burning buildings. **2-** ***Fire station:*** It organizes the function of fire brigades. **3-** ***Rescue, Police and their Center Agents:*** These agents are other platoon agents each responsible for their specific task in rescue simulation environment.

Our research goal is to arrive at effective fire extinguishment behavior for fire brigade agents. The system simulates Kobe city for 300 cycles (each cycle corresponding to one minute in real world) after the earthquake [10]. In each cycle, the fire simulator simulates fire propagation in the city by means of pre-computed statistical information gathered from the real Kobe earthquake in 1995. The final performance of the agents' work is assigned in proportion to the unburned buildings at the end of simulation. At each cycle each fire brigade agent can send one of the following actions to the system's kernel : **(I) Extinguish (B):** for which the simulator extinguishes (decreases the burn of) building **B** in proportion to the maximum amount of water a fire brigade can supply ; **(II) Move (R):** in which **R** is a route plan; the traffic simulator moves the fire brigade through **R** in the next cycle. By regulation, an unburned building can be ignited by one of its neighboring burned buildings. Let's call a group of neighboring burned buildings a **fire site**. In order to stop the spread of existent fire to other unburned buildings (and thus achieving a higher final performance), the fire fighters must try to extinguish the boundary buildings of each fire site at first (see figure 1).

4 Implementation and Experiment

In this section, we illustrate our layered approach via a full-fledged implementation in RCRSS [11]. Here, the high-level goal is for a team of fire brigade agents to achieve complex collaborative behavior.

4.1 The Fire Spread Speed

First, the agents learn a basic environmental behavior: the fire spread speed. As mentioned before, the potential buildings for burning are the buildings which are neighbor to at least one of the **border** buildings of a fire site. We chose to have our agents learn this behavior by means of a ML algorithm, because the fire simulator behavior in

Figure 1. Border buildings are marked with "*" symbol. Four fire brigade agents are extinguishing a building in border.

in this case is so complex and thus, fine-tuning an approximative function by hand is difficult.

We provided our agents with a large number of training examples and used a supervised learning technique: neural networks (M_1). A fully connected neural network (f_1) with 13 inputs and 16 hidden sigmoid units and a learning rate of 0.7 was trained. I_1 consists of the following parameters gathered from the environment at regular time intervals: **the potential building B's Total Area,** $\{1 \le \forall i \le 3$ **: Fieryness of B_i, Distance Between B and B_i, Burning Time of B_i** $\}$ where B_i s for $1 \le i \le 3$ are the three nearest buildings to B, respectively; and **Fieryness** is the state that specifies how much the building is burning [11]. Also $O_1 = \{$ ***EF(B)*** $\}$ where is ***EF(B)*** the expected time for building B to be ignited. T_{M_1} was constructed by sampling the environment's parameters in regular time intervals for 2000 times. The neural network was trained for 15000 epochs. The network was trained by *Joone* [12] (a java package for training and using neural networks), giving us the opportunity to *serialize* the trained neural network weights and biases into a file for real time usage during the simulation. At the first cycles of the simulation, each agent loads this file into its memory. Thus, our learning method is off-line in this case, providing the same knowledge to all the agents in their mission domain.

The RMSE (error) of the NN at the end of training was approximately 0.04[1]. Also, for unseen data, the trained network performs well by an average error of 8 cycles (for $EF(B)$), where the range of $EF(B)$ is 150 cycles and the average is taken over 500 patterns.

4.2 The Effect of Collaboration

Collaboration is a key idea for successful teamwork in RCRSS, as well as other Multi-agent systems. Although the effect of extinguishing is not defined explicitly, it may not be difficult for even a few fire brigades to extinguish an early fire. On the contrary, it is difficult for even many to extinguish a late and big fire. Consequently, the agents should be aware of the effect of collaboration in fire extinguishment for further decision making. In this layer (L_2), we aim to understand the environment's feedback to the joint effort of k fire brigade agents for extinguishing a specific building B. Similar to the previous layer, complexity leads us to using a ML algorithm, which is again a Neural Network (M_2). I_2 is comprised both from the *target building*'s characteristics and the *number* of collaborating agents : I_2 = **{B's Total Area, B's Fieryness, B's Burning Time, NP, MinDistance, NCol}** where **NP** is the number of B's neighboring buildings which are potential for igniting B, **MinDistance** is the minimum distance of such buildings to B and **NCol**} is the number of collaborating agents. Also O_2 = **{*EX(B)*}**, where ***EX(B)*** is equal to the expected time for the collaborating agents to extinguish building B. A fully connected neural

[1] Inputs and outputs are normalized to [0,1].

network ($\boldsymbol{f_2}$) with 6 input and 15 hidden sigmoid units was trained. The learning rate was equal to 0.7. The neural network was provided with 2500 input patterns ($\boldsymbol{T_{M_2}}$) gathered from several simulation runs. In each run, the agents try to extinguish a specific building as the target, taking log data at regular time intervals from environment for constructing $\boldsymbol{T_{M_2}}$. The NN was trained for 5000 epochs. After nearly 4500 epochs the network's RMSE will not be decreased. So we finished learning the NN at this epoch. The NN may learn the input pattern noises in case of further learning. We've provided the NN with training examples in which the **NCol** parameter ranges from 1 to 6. Our NN has the strength of generalizability (when $\mathbf{NCol} > 6$), However, if the maximum number N of collaborating agents in a simulation is known in advance, one may train N different neural networks separately (providing the i^{th} NN only training examples with $\mathbf{NCOL} = i$) in order to have more specialized NNs.

Now that the agents are provided with useful primary knowledge, they must be able to plan an intelligent strategy for extinguishing a whole fire site.

4.3 Extinguishing a Fire Site

In order to extinguish a whole fire site, the agents use their learned functions (**EF, EX**) to decide upon which building is more urgent (prior) to extinguish in each situation. In this layer ($\boldsymbol{L_3}$) each building $\boldsymbol{B}$ will be assigned a priority value for extinguishment $P(B)$ ($\boldsymbol{O_3}$), based on its influence on the unignited buildings. Regarding this priority, all the agents rush to the building with the maximum priority value by sending the **Move** commands to the system's kernel, sequentially. After all agents were located in a certain distance to the target building, they collaboratively extinguish this building by sending Extinguish commands[2]. After extinguishing this building, the agents will evaluate other buildings' priority value again, choosing their next target building for extinguishment. The agents will repeat this process until no burning building remains in the fire site. As the parameters used for decision making in this layer can be noisy and inaccurate, a fuzzy rule-base system was used to evaluate the priority of a building. The developed fuzzy system uses singleton fuzzifier, the Larsen inference engine and the function left maximum defuzzifier [13]. At first, the fuzzy system assigns to each unburned building $\boldsymbol{B}$, a danger value ($D(B)$), which indicates the potential damage that $\boldsymbol{B}$ can impose to the system's performance when ignited. Due to our observations, the much area a building has, the later the fire brigades can extinguish it (causing lower performance). Consequently, $D(B)$ is in direct proportion to $\boldsymbol{B}$'s total area. On the other hand, dangerousness (potential imposing damage) of an unburned building depends on its expected time for ignition; namely the sooner a building is ignited, the more damage it will impose to the system's performance, in long run. The system uses these two linguistic variables ($\boldsymbol{I_3}$) for determining $D(B)$ as a crisp value between 0 and 100. The membership function of **EF(B),** $\boldsymbol{B}$'s area and $D(B)$ in High, Average and Low sets are depicted in figure 2; also the corresponding values for the labels are given in figure 3. The knowledge base of the fuzzy system consists of 9 fuzzy rules. Regarding figure 3, for each of the 9 membership status of the linguistic variables in the sets, a rule is generated. For example, the entry in the first row and third column of dangerous table in figure 3 corresponds to the following rule:

> if **EF(B)** is **LOW** and $\boldsymbol{B}$ **'s Total Area** is **HIGH** then $\boldsymbol{D(B)}$ is **HIGH**

[2] In our simulation, the agents work together at all the times.

Now that $D(B)$ is evaluated for each unburned building, the agents should determine the most urgent (burning) building for extinguishment.

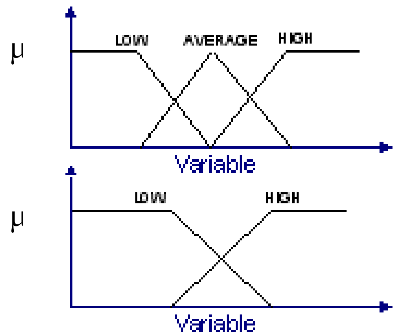

Figure 2. The general Member function diagrams for all linguistic variables' fuzzifiers.

For this purpose, another fuzzy system is used for evaluating the priority of each burning building ***B*** ($P(B)$) for extinguishment. $P(B)$ is evaluated based on three parameters: **(1) EX(B)**: The more time extinguishing building ***B*** takes, the less prior is ***B*** for extinguishment, due to the agents' time loss for extinguishing this building. **(2)** The maximum value of $D(B)$ among ***B***'s neighbors ($\max(D(B))$) **(3)** The distance between ***B*** and its most dangerous neighbor building ($Dis(B)$).The second fuzzy system's configuration is much like the first one. The membership functions of $\max(D(B))$ and $Dis(B)$ are the same as previous functions (Figure 2). The corresponding values for labels are given in figure 3. The only difference is $EX(B)$, for which only two labels (HIGH, LOW) are used. The knowledge base of the fuzzy system is constructed like the previous system regarding figure 3 (based

	LOW	AVERAGE	HIGH
B's Area	25000	50000	100000
EF(B)	17	50	80
D(B)	25	50	75
Dis(B)	10000	20000	30000
EX(B)	10	-	25
P(B)	25	50	75

Dangerousness

EF(B) \ Area	Small	Average	High
Small	AD	HD	HD
Average	SD	AD	HD
High	SD	SD	AD

Priority

EX(B) \ max(D(B))	Small	Average	High
Small	AP --	AP	HP
Ave.	SP	AP	HP --
High	SP	AP --	AP

Figure 3. The labels' corresponding value used in fuzzy inference (left), corresponding table for evaluating dangerousness (middle), corresponding table for evaluating priority (right).

on $\max(D(B))$, $Dis(B)$), except for the dashed entries. For these entries the following rules were used:

(i) if **Dis(B)** is **LOW** and **D(B)** is **LOW** and **EX(B)** is **LOW** then **P(B)** is **AVERAGE**
(ii) if **Dis(B)** is **LOW** and **D(B)** is **LOW** and **EX(B)** is **HIGH** then **P(B)** is **LOW**

These rules distinguish the problem features at entry (1, 1) for the **EX(B)** variable. The same pattern is used for the other two dashed entries[3]. Finally a crisp value for $P(B)$ is driven from the fuzzy system. Now the fire brigade agents choose the building with maximum priority as their next target.

5 Results

In order to evaluate the proposed method, the developed team of fire brigade agents was tested several times with different configurations of the city. In the previous work [14], another "state of the art" approach was presented, in which the fire brigade agents extinguish the fire site by dividing it into several sectors (assuming the fire site as a circle) and

[3] Entries (3, 2) and (2, 3).

select the sectors to extinguish, based on a cost function implemented in "Eternity" rescue simulation team. The agents will extinguish all of the buildings in one sector before going to the next sector; also, the most prior buildings for extinguishment are found with "state of the art" algorithms not including machine learned components. The team has won the 4th place in the 2003 International RoboCup rescue simulation league and won the championship later in CIS2003 competitions. In figure 4, our proposed method's performance is compared with Eternity's performance based on the initial size of the fire site. The comparison was done for two configurations. In the first configuration the city does not have any road blockage; but in the second, the city is simulated with road blockages generated by RCRSS GIS [11]. The results show the *average* performance of these two methods on the specified initial city map. The performance is evaluated regarding

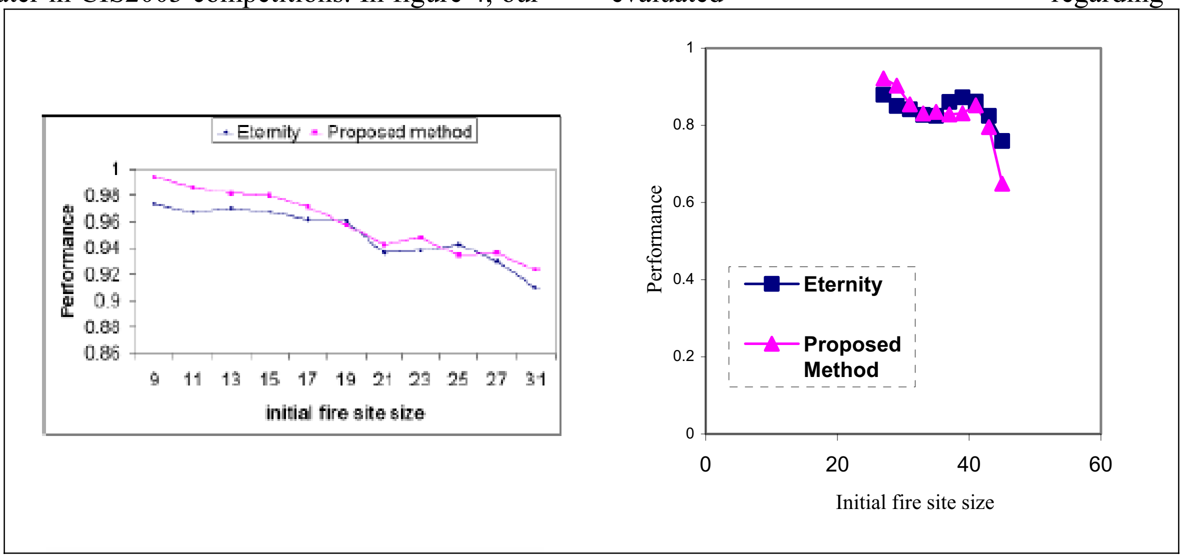

Figure 4. Performance comparison between the proposed method and Eternity's approach. In the first configuration the city is simulated with no road blockage (left), however, the second configuration considers road blockages (right).

RoboCup 2003 regulations:

$$Performance = \sqrt{\frac{B}{B_0}}$$

where B_0 is the total area of buildings and B is the area of the unburned buildings at the end of the experiment.

The results show that in the first configuration, our method works better than Eternity's approach for nearly all of the tested values as the initial fire site size. As the initial size of the fire site increases, Eternity's performance gets closer to our method's performance. This is due to the fact that for larger fire sites, the agents' movement cost between the site's buildings increases, which causes lower performance in the proposed method. However, in Eternity's approach the agents may not move to the next sector of the fire site, unless they extinguish the current sector's buildings; thus theagents will not pay much for movement costs.

In the second configuration, however there exist road blockages in the city which will cause severe movement cost for agents. As it is clear in the diagram, our proposed method's performance is better for fire sites with initial size less than 37, however, as the fire site initial size increases, Eternity's performance becomes better. This is because the road blockages in the second configuration cause more movement cost in comparison to the first one; thus Eternity has the opportunity to achieve better performance in large fire sites. As the fire site initial size decreases, our proposed method performs better due to less movement cost.

6 Conclusion and Future Work

This paper has presented a layered paradigm for acquiring a function which tries to maximize the fire brigade agents' performance, and illustrated it with a fully-implemented application in RoboCup Rescue Simulation System (RCRSS) as a challenging and complex multi-agent control problem. Moreover, the results showed that using the layered paradigm for solving such complex task lead us to significant improvement in the simulated robots performance. It is noteworthy that using this paradigm gives us the opportunity of directly combining different ML algorithms within a hierarchically decomposed task representation; thus one may exploit both the robustness of Fuzzy logic algorithms beside the strong learning capability of neural networks. In order to find out whether the proposed model is suitable for other domains, applying it to other control problems is necessary. Based on the acquired results, there's a strong possibility that applying the proposed method to other domains may bring about good performance improvements. Important directions for future work are to: **(I)** Design, revise and develop further layers in this domain; and **(II)** applying this paradigm to other complex domains.

For instance, it would be beneficial to introduce another layer above the previous learned layers, which is responsible for dividing the fire brigade agents among different sites (when several sites exist). This will introduce higher level of cooperation for the agents. The findings lend support to the conclusion that layered learning paradigm's power is derived from the concept of simultaneously using different ML algorithms in a hierarchically task representation.

References:

[1] Peter Stone. Layered Learning in Multi-Agent Systems. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1998. Available as technical report CMU-CS-98-187.

[2] AAAI. Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS9601. Sandip Sen–Chair.

[3] Gerhard Weiβ and Sandip Sen, editors. Adaptation and Learning in Multiagent Systems. Springer Verlag, Berlin, 1996.

[4] Faustino John Gomez : Robust Non-linear Control through Neuroevolution. Report AI-TR-03-303, August 2003.

[5] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4:237{285, May 1996.

[6] Long-Ji Lin. Reinforcement Learning for Robots Using Neural Networks. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1993.

[7] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviors. In Proceed- ings of the Eighth National Conference on Artificial Intelligence, pages 796{802. Morgan Kaufmann, 1990.

[8] Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In Proceedings of the Fifteenth International Conference on Machine Learning. Morgan Kaufmann, 1998.

[9] Peter Dayan and Geokrey E. Hinton. Feudal reinforcement learning. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, Advances in Neural Information Processing Systems 5. Morgan Kaufmann, San Mateo, CA, 1993.

[10] Hiroaki Kitano, Satoshi Tadokoro, Itsuki Noda, Hitoshi Matsubara, Tomoichi Takahashi, Atsushi Shinjou, and Susumu Shimada. Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In Proc. Of IEEE Conf. on System, Man and Cybernetics, Dec. 1999.

[11] The RoboCup Rescue Technical Committee: RoboCup Rescue Simulation Manual, version 0, July 2000.

[12] Joone's Technical Reference, see *http://www.jooneworld.com* for more information.

[13] Li-Xin Wang : A Course in Fuzzy Systems and Control. Prentice-Hall International Inc., 1997.

[14]E.Mahmoudi,S. Mirarab,N.Hakimipour, A.Akhavan Bitaghsir : Eternity's Team De scription, Springer Verlag, Robocup 2003 Issue, 2004 (To appear).