The CSI Journal on
Computer Science and Engineering
Vol. 3, No. 2&4, 2005
Pages 1-7
Short Paper

A New Architecture for Multi Agent System in Rescue Simulation Environment

Jafar Habibi Hamid Reza Vaezi Joze

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.

Abstract

This paper presents a new architecture for implementing intelligent agents cooperating in a special Multi Agent environment, namely the RoboCup Rescue Simulation System. This is a three layered architecture which customized for cooperating three kinds of agents to minimize damages, caused by a big earthquake in rescue simulation environment. Structural decision making algorithms are combined with heuristic ones in this model. One of advantage of this architecture is supporting both Centralized and Distributed multi agent systems. At the end comparing this new architecture with those which be used till now could show the strength of new architecture.

Keywords: Multi-Agent System, RoboCup Rescue Simulation, Layered Architecture.

1. Introduction

The RoboCup Rescue Simulation League started with the intention of training the Disaster Response Team to minimize the financial and life costs in a disaster. A disaster in this system is defined as a town being hit by earthquakes which results in blocked roads and broken buildings and also some of the buildings catch fire. The RoboCup Rescue Simulation system makes a test-bed for implementation of various Multi-Agent algorithms. Its capabilities cover a wide range of possible styles of algorithms. It is also a standard environment for testing different techniques of making standard software agents with distributed architecture [5]. Rescue Simulation System also provides a standard framework for testing proposed algorithms and mathematical models of disaster events [10]. Designing an autonomous agent set like the one required for RoboCup Rescue Simulation and is a little bit more of a challenge. Planning effective collaboration for a Multi-Agent team in disastrous environments still remains a challenging area in AI. Efforts of Multi-Agent researchers have provided somewhat of a standard in modeling and designing software. A lot of effort has been made to coordinate different agents and making autonomous decisions that lead toward the team goal [11]. But practical results in complicated domains such as RoboCup Rescue Simulation indicates that heuristic criteria still remain as a major part of a successful system [6]. This may signal lack of satisfactory models for these complicated situations. These evidences encouraged us towards the implementation of new architecture for all kind of agents. Our structured model constitutes the core of the system which acts on advices generated by heuristic components. In fact heuristic components decrease the complexity of the domain and the structured core analyzes these reasonable incoming advices for making the required decision. Practical

results convinced us that this is an achievement over pure heuristic designs in this domain. The next section will introduce a brief problem definition. Then the architecture will be presented in Section 3. *Common Strategies* that is the second layer will be described in Section 4. Section 5 dedicated to the architecture when implemented for centers. In Section 6 we will provide some experimental results and finally in section 7 we will conclude the paper.

2. Problem Definition

The goal of the RoboCup Rescue simulation project aims to simulate rescue teams acting in large urban disasters [2]. Precisely, this project takes the form of an annual competition in which participants are designing rescue agents trying to minimize damages, caused by a big earthquake, such as civilians buried, buildings on fire and blocked roads. In the simulation, participants have approximately 30 to 40 agents of six different types to manage:

Fire Brigade
: There are 0 to 15 agents of this type. Their task is to extinguish fires. Each Fire Brigade agent is in contact by radio with all other Fire Brigade agents as well as with the Fire Station.

Police Force
: There are 0 to 15 agents of this type. Their task is to clear roads to enable agents to circulate. Each Police Force agent is in contact by radio with all other Police Force agents as well as with the Police Office.

Ambulance Team
: There are 0 to 8 agents of this type. Their task is to search in shattered buildings for buried civilians and to transport injured agents to hospitals. Each Ambulance Team agent is in contact by radio with all other Ambulance Team agents as well as with the Ambulance Center.

Center Agents
: There are three types of center agents: Fire Station, Police Office and Ambulance Center. These agents can only send and receive messages. They are in contact by radio with all their mobile agents as well as with the other center agents. A center agent can read more messages than a mobile agent, so center agents can serve as information centers and coordinators for their mobile agents.

In the simulation, each individual agent receives only the visual information of surrounded region. Thus, no agent has a complete knowledge of the global state of the environment. This uncertainty complicates the problem greatly because agents have to explore the environment and they also have to communicate to help each other for a better knowledge of the situation. There is a coordinating process that all other processes such as simulators interact through it, so we call this process *Kernel*. Communication within components of server is accomplished with an extended UDP protocol named *Long-UDP*. Moreover, there are certain buildings in the city that agents can heal themselves inside them (decreasing the amount of damage) or fill their water tanks (increasing the amount of available water for FireBrigade only) while they are inside them. These buildings are called **Refuge**s. Detailed information is available in [1].

3. Architecture

Figure 1 shows a schematic view of the agents' architecture. This structure is used in all agents. According to each agent's task, the higher layer (Decision Making) will be customized. So other layers are the same for all agents.

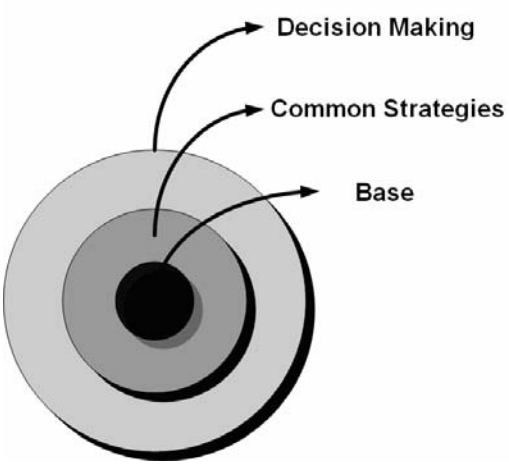

Figure 1. Three layered architecture

The system consists of three main layers. The lower layer is responsible for sending and receiving data from kernel. All decisions made by higher layers should be passed through this layer, so this layer act like an interpreter which translate high-level comments for kernel and also sensory data for higher layer. This layer is named *Base* in figure 1.

Beyond action that placed in *Base* layer there are some actions that are common in all agents. These are mainly modules used by agents to coordinate such communication, path planning, ordered based behavior and searching. Therefore they do not seem to be strategic procedure which could cause coordination within agents in the first glanced we placed them in *Common Strategies* layer. In the next section the aspects of this layer will be described more n details.

Decision Making layer contains heuristic module which customized for each agent. There are a wide range of algorithms that cover the topic of decision making. Most of these algorithms tend to make decisions upon a **restricted** number of well-defined parameters. The domain of the rescue simulation contains a large number of important parameters that should participate in decision making. Experiments for making direct decisions through passing all these parameters to such algorithms have shown unsatisfactory results. So, the role of heuristic components in this design is to provide an abstract view of the world model as an input for decision making routines. Although this world model is provided by sensory information and communication with other agents, the only information that is used as input is world model in this layer and filling world model as a common strategy in our multi-agent system has

carried out by *Common Strategies.* If we consider *Decision Making* layer as an input-output function, The World Model can be considered as the input for this layer and the output will be a high-level commend which is passed to the *based layer* through *Common Strategies.* The Concept of this paper does not cover specific algorithms for decision making of different kinds of agents, but this is important that our architecture has no limitation on algorithms in *Decision Making* layer so the algorithms could be centralized or distributed.

4. Common Strategies

Figure 2 shows the three layered architecture for agents. The *Base* layer has been descried in the previous section and *Decision Making* layer should be customized for each kind of agents. We are going to briefly describe main components of *Common Strategies* layer which are responsible for cooperation among agents, in the following of this section. Although in centralized systems center is the main part of cooperation, in distributed systems these strategies are the only components in system that make agents cooperate.

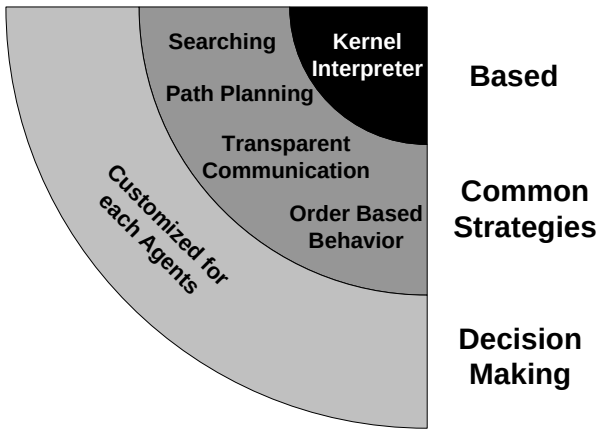

Figure 2. Three layered architecture for agents

4.1 Transparent Communication

As mentioned above the agent can decide about its action in *Decision Making* layer based on its world-model. Thus the information communicated among agents should be applied to world-model. We also know that agents get information from their surrounded environment. These are the main data that feed our multi agent system. Some agent developers tend to communicate processed information. Such of the agent telling other "Come with me" It is the same as raw information communication, because all such messages are derived from agents' sensed information. So the purpose of communication is to carry sensed information to complete agents' world-model. It is independent of the type of the agents. Moreover it is independent of each agent actions.

Implementing communication is well defined, reporting data that seems to be useful for others. The one thing that remains is to set priorities for information. The model we suggest is that every agent and center has a world model for every center or agent it is communicating with. Each of these world-models contains the information our agent thinks the corresponding agent has. When an agent wants to send a message to another it calculates the difference of its own world-model with the target agent's world model. Then the differences are sorted according to their priorities and best ones are sent.

We called this kind of strategy Transparent Communication which means that it is transparent for *Decision Making* layer and the process of getting and sending proper information by mentioned algorithm used autonomously in *Common Strategies* layer.

4.2 Path Planning

Path planning is necessary part of *Common Strategies* layer. Not only agents need to cooperate with determined suitable path to travel in order to avoid collision and blocking but also *Decision Making* layer needs estimated path length of each target to decide what to do (It could obtain from World Model). Path planning module contains three main components. These components help each other to find the appropriate paths to all targets when the World Model changes.

BFS: This is the main path planning module. Typically the agent needs to find physical paths on the simulated city to change its position. The input to this problem is a graph. Graph edges represent the open roads at the beginning of the disaster simulation. As the time goes by, some roads will be opened so this graph is a dynamic graph. The problem of handling such a dynamic graph is solved partially by deterministic approaches like continuous Dijkstra [8]; however since positions of blocked roads are unknown till late in the simulation, such a method will not give a satisfactory result. The base of our solution here is a simple BFS over a graph of all the potentially open roads. As the agents explore the graph, closed roads will be discovered and marked on the graph.

Collision Detection: Sometimes agents can't travel all over the roads of an open path. This occurs for example when a certain direction of the road is not accessible because some other agents have blocked the way. In this situation this heuristic strategy is used: Upon discovery of a blocked direction of a road, the certain edge will be removed from the graph, temporarily. After a fixed period of time, the edge will be tagged as an open road again.

Traffic Control: When agents are assigned to missions, in many cases the roads are blocked because of working agents. Since agents are working independently, it is very difficult for them to resolve this problem on their own. So appropriate cooperating needs to solve this problem. Since all agents use the same strategy and using transparent communication it could be supposed that after a constant time agent's World Model contain previous information of others, it is not difficult for agents to predict other agents' position or action. So using the same traffic control module in all agents and distribute them according their specification (such as assigned number) could overcome these kind of problems.

Figure 5 shows a sample of this event without using Traffic Control module (up) and after applying Traffic Control rules (down).

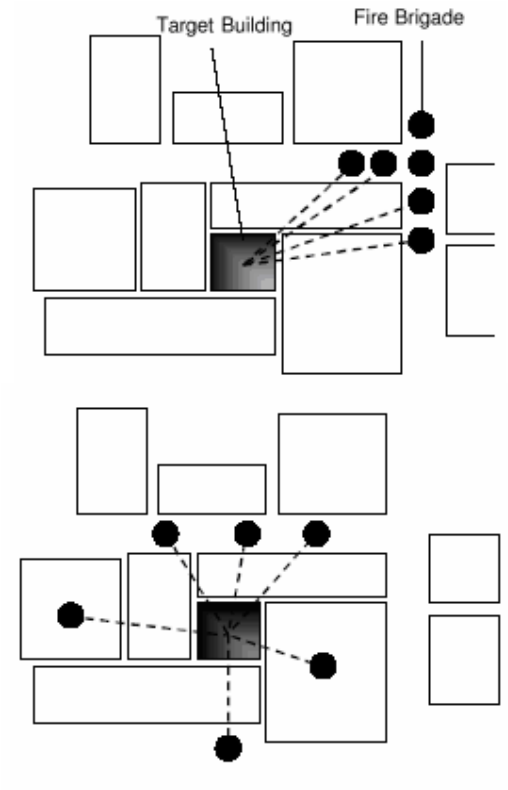

Figure 3. Traffic control

4.3 Searching

When an agent does all of his tasks or he has nothing to do, he starts to explore the city regularly. This is not only for finding new fired building but also for finding injured civilians that are not reported yet.

Our search task is not assigned to specific agents or specific types of agents. It is a task that is originally belong to all kind of agents and would be done automatically when agents do not have any other high priority task.

We have modeled the city's buildings into a simple graph G(V,E) where V are all buildings and building i is connected to building j if it can be seen from there i.e. in the current rules if the distance between building i and building j is less than 30 meter.

A dominating set for graph $G(V,E)$ *is* V' *where* V' *is a subset of* V *such that* $\forall u \in V - V'$ *there is* $v \in V'$ *for which* $(u,v) \in E$ *. And minimum dominating set is a dominating set with minimum size.*

Clearly, a dominating set of this graph is enough to be searched in order to be sure that all the buildings have been visited and according to our communication modules, we can suppose that all these data has been shared after a while. As Minimum Dominating Set problems is NP-Complete so there is no polynomial solution for this problem. So we need to find an approximate solution [12]. Also if map of a city is available so we can use offline process to find Minimum Dominating Set.

As searching task is a common task for agent cooperation and it is necessary to minimize the time that the whole city be explored. We divide buildings of dominating set to all agents so when an agent goes into searching mode, he moves to the nearest building of his dominating set, and the subset that has not been explored yet. When all of his subset are explored, he use another agent's subset.

4.4 Critical Situation

In critical situation the agents should ignore all plans and handle this critical situation. These situations act like the interrupt system in a computer which stops the normal routine and the exceptional routine are executed and after that control are returned to normal routines. It could occur in tow cases:

1. When an agent receives a certain minimum level of damage.
2. When a Fire Brigade agent detects that it has ran out of water.

In such cases all plans will be ignored except going to the nearest Refuge building. It means that in critical situation *Decision Making* layer is off.

4.5 Ordered Based Behavior

Architecture should be flexible for various methods in Multi Agent System. One of The main challenges in multi agent systems is "Centralized vs. Distributed", so our architecture should support both of them in order to be a popular architecture in multi agent system. In distributed system *Decision Making* layer decide the action considering World Model and this is totally distributed agent and agents just related to each other using transparent communication. In the case of centralized system obviously the corresponding center (Police Office, Ambulance Center or Fire Station) is responsible for providing decision for all its agents considering its World Model that is provided by transparent communication module of its agents. So this architecture needs a module to obey all comments received from a center, named Ordered Based Behavior. Despite this model supports Centralized decision making, in the lack of that, agents could decide themselves simply using their Decision Making layer. So it can also support hybrid models.

5. Center Architecture

The center architecture is exactly like agent architecture knowing that centers do not use Path Planning, Searching and Order Based Behavior. So the only active component in *Common Strategies* layer is Transparent Communication.

If the system is distributed, centers are not more than message passing so using transparent communication they get raw data from agents or other centers and send appropriate data to them. *Decision Making* layer is not used in this case. Figure 3 shows the architecture for center in distributed multi agent system.

In the case of centralized system, center should make decision for all of its agents in *Decision Making* layer. As an example of centralized decision making in the remaining of

this section a centralized method for Fire Station called FAIS [13] will be describe.

Fire Station module is responsible for centered decision making and coordination. It computes a profitable subset of fiery buildings and assigns a subset of idle agents to them for the extinguishing operation. The system sets a proper timeout for assigned mission called mission time, and after this deadline, Fire Station considers the assigned Fire Brigades as free agents. Figure 4 shows the details of the Fire Station architecture. The components of *Decision Making* layer are:

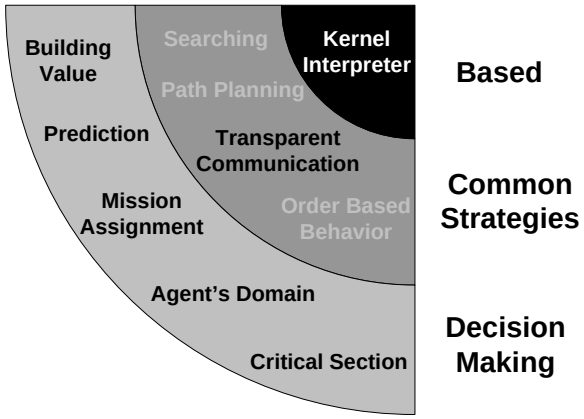

Figure 4. Three layered architecture for fire station

Building Value: This is a heuristic real value, computed for each building. Buildings with bigger values are candidates for extinguishing in the mission assignment. The value for building *b* is computed in the following way:

$$V(b) = \beta N(b) + \gamma HV(b) - \alpha \sum_{i \in J} d_i(b) \qquad (1)$$

di(*x*) is the distance between Fire Brigade agent number *i* and building *x*. *J* is the set of indices of currently free agents. *N*(*x*) is the number of unfired neighbors of the building *x*. *HV* (*x*) is a value associated with a building *x* that indicates how much *x* can be destructive if we let it spread the fire through the city. For example, a fiery building in the margin of the city is much less dangerous than a fiery building in the city center. Figure 4 shows the details. First all fiery neighbors are computed, and the convex hull of the set, *H*, is constructed through a Graham scan method [9]. Then a unit vector u is computed such that in position of building *x*, it points toward bisector of angle of *H* at vertex *x* (figure 4). All other fiery buildings that are not much further than a certain margin, from the semi-line in direction of u starting at *x*, are examined to find the nearest one to *x*, let *y* be that building. We define *HV* (*x*) = *distance*(*x; y*). α , β and γ are positive factors that are determined after fine tuning of implemented agents. Since a small difference in the score result could have been caused by the random functions involved during the simulation process we couldn't apply the fine tuning process by automatic means. Furthermore the final score doesn't show if each agent was acting in a more logical way during different parts of the simulation, so we changed the parameters by viewing the logs and deciding whether the agents had acted better or not. Because we changed these parameters by visually surveying the logical behavior of our agents they are independent of the city. The intuition behind (1) is that buildings with more unfired neighbors, having more dangerous position to spread fire and with much free Fire Brigades near them are more profitable to choose.

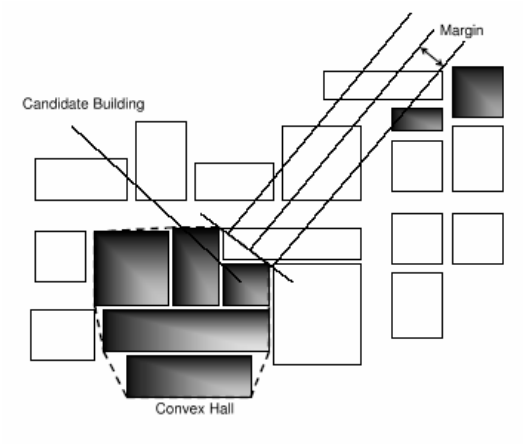

Figure 5. Finding the fire border

Agent's Domain: For the previous section, we should compute the set of free agents for each building. But this differs for each building, since the roads graph is dynamic and some idle agents can't reach some certain buildings (that is some certain graph vertices). So, in this component, the set of reachable buildings for each agent is computed through a BFS circumnavigation. Therefore, the set of free agents considered for each building can be a *proper subset* of all idle agents.

Prediction: The Fire Station needs to have proper approximation of future situation to schedule effective missions. Since phenomena like fire spreading in deterministic environments obey some certain rules, it is possible to provide some approximation. This module is responsible for maintaining a compact set of previously known scenarios and matching the current scenario with them to predict some future parameters like which buildings will be ignited in the next cycles and how much water is sufficient for extinguishing a certain building. A feed forward neural network provides the compact representation in addition to proper response time and accuracy [4].

Mission Assignment: This module does the real task of mission assignment. At first prediction module approximates future state of the scene and then potential missions are identified. Building Value module then selects some profitable buildings between these. This comes up with a limited number of proper buildings. Free agents should be assigned to these buildings in an optimum manner, i.e. sum of the distances for all agents to reach their missions be minimum. This problem is a restricted type of the graph matching problem with limited degrees at vertices [3]. An LP solver does these final assignments [7].

Critical Section: The way our agents behave in the first cycles of the simulation is of utmost importance. Extinguishing more buildings in the first cycles will reduce the amount of fire spreading quite significantly. This will make handling the disaster far easier. Furthermore, not only the prediction module hasn't enough information to make satisfactory predictions in the first cycles, but also if the prediction results are acceptable, many roads are blocked at the start of simulation, so many assigned missions will not be completed successfully.

Critical Section component overrides mission assignment in the certain interval of initial cycles. In this interval, missions are not assigned optimally, but simply all agents are assigned to the best building for extinguishing. This has shown a better performance than using prediction and assignment from the beginning.

6. Experimental Results

A set of Agents and Centers based on this architecture is fully implemented and participated as *Impossibles* team in the 2005 Rescue Simulation RoboCup Competitions. There was a wide range of strategies available in the competition and our agents showed the best performance between them and placed first. Table1 and Table2 are the result of Semifinal and Final of the competition which contain running systems in 5 and 9 different environments. In this section we will discuss how this architecture helped us in achieving this result.

Table 1. RoboCup 2005, rescue simulation result, semifinal

Rank	Team	Score	Point
1	Impossibles	409.06	36
2	Caspian	394.136	32
3	MRL	352.216	27
4	Kshitij	349.975	25
5	S.O.S	338.375	20
6	RoboAKUT	334.986	20
7	ASAP	307.153	12
8	Persia	284.011	8

Table 2. RoboCup 2005, rescue simulation result, final

Rank	Team	Score	Point
1	Impossibles	574.329	32
2	Caspian	564.046	28
3	Kshitij	519.559	21
4	MRL	428.014	9

6.1 High-level Programming

Since in this architecture, three layers have separate functionally, the higher level that decides final decision, namely *Decision Making* is not so complicated by low-level action and complexity of strategies. It helps developer to do high-level programming in Decision Making layer so manipulation this part is easy. We change our algorithms simply according to visual comparison during firs rounds. And this improves our performance in the semifinal and final of the competition.

6.2 Critical Section

Since in some sections of the competitions there were maps with lots of initial fiery buildings, correct decision on task assignment in the first few cycles of the simulation greatly affected the overall result.

6.3 Prediction

In cities with lots of initial fire points teams had problems with their agents losing control of the fire sites. In such cases our predictor module proved to be quite useful since it provided our Fire Station with a good overall view of the situation. This helped our agents to work in a logical manner even during such circumstances.

6.4 Communication

Cooperation of agents that are using distributed system such as Police Force and Ambulance Team convince us that they use same data to decide and it means that Transparent Communication could share raw data as soon as possible. The strength of this method appears well when injured civilians were rescued by Ambulance Teams as a Police Force find that injured civilian.

6.5 Traffic Control

As mentioned before traffic control is of utmost importance during this simulation process. In the logs we saw many teams had trouble with traffic jams which caused major problems for their agents. The path planning process which our agents used, help them in choosing optimum paths with lower traffic. This process took advantage of our fire station to inform the agents of any temporary road blocks which occurred during the process.

6.6 Flexibility

When protocols of communicating with Rescue Simulation Kernel are changed in 2006 the only thing that needs to be changed in our architecture was *Base* layer while some architecture faced critical problems. This is a good evidence for proving flexibility of this layered architecture.

7. Conclusion

In this paper we discussed a new architecture, which has been used to develop multi agent system for the rescue simulation environment. Multi agent environments are typically complicated and require a great degree of cooperation between the agents and a potent architecture in order to accomplish their duties. The proposed architecture by benefiting from its three layered design, overpowered any previously known and implemented architecture in the rescue simulation field. Although this architecture designed to categorized actions in low-level, strategy and high-level actions, it is a flexible architecture for multi agent system

which supports centralized model, distributed model and hybrid model. In the other hand separating strategies from decision make it possible to test different decisions with different strategies that cause it good test-bed for comparison of various methods.

Future works will include testing this architecture on more practical multi agent environment.

Acknowledgement

This research was conducted in the AI and Robotics Laboratory of Sharif University of Technology and we would like to thank all the members of the *Impossibles* team who worked on the rescue simulation project.

References

[1] The RoboCup Federation, RoboCup Regulations and Rules, http://www.robocup.org, July 2001.

[2] H. Kitano, "RoboCup rescue: A grand challenge for multi-agent systems," *Proc, 4th International Conference on Multi-Agent Systems(ICMAS),* pp. 5-12, 2000.

[3] C. Berge, *Graphs and Hypergraphs*, 2nd ed., North-Holland, 1976.

[4] L. Fausett, *Fundamentals of Neural Networks*, Prentice Hall International, 1994.

[5] J. Habibi, M. Ahmadi, A. Nouri, and M. Sayadian, "Implementing heterogeneous agents in dynamic environment, a case study in robocup rescue," *The First German Conference on Multiagent System Technologies*, Springer Verlag, Vol. 2831, pp. 95-104, 2004.

[6] J. Habibi, M. Ahmadi, A. Nouri, M. Sayyadian, and M. M. Nevisi, "Utilizing different multiagent methods in robocuprescue simulation," Proc, RoboCup-2002 Symposium, 2002.

[7] B. Korte and J. Vygen, *Theory and Algorithms*, Springer, 2000.

[8] J. S. B. Mitchell, "Shortest paths among obstacles in the plane," *Proc, the ninth annual symposium on Computational geometry,* pp. 308-317, 1993.

[9] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction,* Springer-Verlag, 1985.

[10] T. Takahashi, I. Takeuchi, F. Matsuno and S. Tadokoro, "Rescue Simulation Project and Comprehensive Disaster Simulator Architecture," *Proc, IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1894-1899, 2000.

[11] C. H. Yong and R. Miikkulainen, "Cooperative coevolution of multi-agent systems," Technical report, the university of Texas at Austin, 2001.

[12] F. Kuhn and R. Wattenhofer, "Constant-Time Distributed Dominating Set Approximation," *Proc, the 22nd Annual ACM Symp. on Principles of Distributed Computing (PODC),* pp. 25–32, 2003.

[13] A. Geramifard, P. Nayeri, R. Zamani-Nasab and J. Habibi, "A Hybrid Three Layer Architecture for Fire Agent Management in Rescue Simulation Environment," *International Journal of Advanced Robotic Systems,* Vol. 2, No. 2, pp. 111-116, 2005.

Jafar Habibi received his B. S. degree in computer engineering from the Supreme School of Computer, his M. S. degree in industrial engineering from Tarbiat Modares University and his Ph.D. degree in Computer Engineering from Manchester University. At present, he is faculty member at the Computer Engineering department at Sharif University of Technology. His research interests are mainly in the area of computer engineering, simulation systems, MIS, DSS and evaluation of computer systems performance.
Email: habibi@sharif.edu

Hamid Reza Vaezi Joze received his B. S. degree in Software/Hardware Engineering from Sharif University of Technology in 2006. Currently He is M. S. student in Computer Engineering department at Sharif University of Technology in the field of Artificial Intelligence. His research interests include Artificial Intelligence, Robotics and Vision.
Email: hrvaezi@gmail.com