# S.O.S. 2004: An Attempt Towards a Multi-Agent Rescue Team

Saman Amirpour Amraii, Babak Behsaz, Mohsen Izadi,
Hamed Janzadeh, Farid Molazem, Arash Rahimi,
Mohammad Tavakoli Ghinani, and Hamide Vosoughpour

Robotics Research Center, Department of Computer Engineering,
Amirkabir University of Technology
No. 424, Hafez Ave.,
Tehran, Iran
{amirpour, behsaz, izadi, janzadeh, molazem,
rahimi, tavakoli, vosoughpoor}@ce.aut.ac.ir
http://ce.aut.ac.ir/~sos

**Abstract.** This paper describes a brief technical specification of the methods we used in *S.O.S. 2004* Rescue Simulation Team. We participated for the first time at the *RoboCup 2003* Rescue Simulation League and ranked 3rd. In this paper, first a basic agent is introduced and then definitions of the problems concerning the tasks of each set of agents are presented and different solutions to each problem are proposed, characterized and evaluated.

## 1 Introduction

RoboCup Rescue Simulation project is an attempt to simulate large urban disasters in order to evaluate the performance of multi-agent rescue teams.

An acceptable performance of a multi-agent rescue team can be characterized by efficiency of the priority scheme for tasks, reaching high degrees of cooperation to accomplish goals, the ability to handle unpredicted situations, and efficiency of inter-agent communication protocol.

In the following sections we describe the methods we used for designing a high-performance rescue team. We first introduce SoSBasicAgent which is a suitable base for implementing our high-level algorithms and then try to produce a formal model for the problems facing each kind of agent and propose our solutions to these problems.

## 2 SoSBasicAgent

SoSBasicAgent is a general-purpose rescue agent used as the base for all our agents. SoSBasicAgent is characterized as a planning agent with a multi-level hierarchical state-based architecture and an explicit state-selection and state-switching policy.

For Platoon Agents, SoSBasicAgent provides two path-finding modules. One module is based on Focussed D* algorithm[6] and the other is a less flexible but extremely efficient method based on Dijkstra algorithm. Both modules have their own advantages and shortcomings and are used according to the situation and goal of Platoon Agents.

The communication system of SoSBasicAgent is consisted of tools to design, specify and use communication protocols based on an XML-based Protocol Definition Language that we designed and implemented.

From the behavior aspect, SoSBasicAgent has a set of predefined behaviors. SoSBasicAgent automatically escapes unsafe states. When it senses self-damage, it leaves its work, goes to the refuge and resumes its previous task again after being healed; and when buried, it broadcasts help requests until being rescued.

SoSBasicAgent is a suitable and complete base, appropriate for implementing high-level strategies and learning algorithms.

### 2.1 Search Strategies

Searching is an important task performed by agents in various situations. Early search for fire sources and search for buried civilians are examples of search operations performed by all the agents. In order to perform efficient search we should design appropriate search strategies.

Last year we used two search strategies. One was based on mapping the disaster space into a grid and assigning each agent to search one or more grid cells. The major shortcoming of this method is the possible unequal density of grid cells which results in unfair task assignment and significant decrease in search efficiency. The other approach is based on integrating search criteria into heuristc functions for Heuristic Search methods like Focussed D* and searching the whole disaster space using these methods. Here, the major shortcoming is that the whole disaster space is searched by each agent and as a result efforts of agents are wasted.

This year we designed a new search strategy. In this method, the search function space is clustered into regions of equal size using Simulated Annealing based on the idea of Cohenen Self-Organizing Map[2]. Then, Voronoi diagram is constructed from the cluster representatives and the regions extracted from the diagram are assigned to the agents for searching.

## 3 Fire Brigades

Fire Brigade agents have the major role of extinguishing fires and preventing fiery buildings from spreading fire to other buildings. Efficient cooperation and correct priority of actions is necessary to achieve this goal. In this section we describe the major problems towards accompishing this goal, and propose our solutions to these problems. First of all we present some definitions which help us characterize and model the problem space and then we identify problems and describe our solutions to each problem.

### 3.1 Basic Definitions

**Fiery Building:** A building with the "fieryness" value of 1 or more.

**Fire Expansion:** A fiery building, after some cycles, causes its neighbor buildings to start burning. We call this phenomenon Fire Expansion.

**Neighbor Buildings:** The set of buildings around a building, which are affected by its fire expansion, directly.

**Fiery Graph:** A simple, undirected graph with the set of vertices equal to the set of buildings and the set of edges equal to $E$ as:

$$E = \{(b_i, b_j) | b_i, b_j \in FieryBuildings, b_i \in Neighbors(b_j)\}$$

In this graph, every simple path between two vertices is called a "Fiery Path".

**Fiery Region:** Fiery Region is a subset of $V$ for which there exists at least one "Fiery Path" between each pair of members. In other words, set of fiery regions are equivalent to the set of connected components of the fiery graph.

**Boundary Vertices:** Boundary Vertices of a region $R$ is the set $B$ of buildings in $R$ defined as:

$$B = \{b_i | b_i \in R, \exists b_j \in neighbors(b_i) \ni b_j \notin FieryBuildings.\}$$

**Extinguished Buildings:** A building with "fieryness" value of more than 3.

**Terminated Region:** is a fiery region with the set of boundary vertices being a subset of extinguished buildings.

Now we are ready to define the goal of Fire Brigade agents. We first define a "Safe State" as a state of the disaster space having:

$$\forall r_i \in FieryRegions \ni r_i \in TerminatedRegions$$

or

$$BoundingVertices - ExtinguishedBuildings = \emptyset$$

In its simplest form, the goal of agents can be defined as reaching a "Safe State". But it is obvious that some "Safe States" have higher priority among others. For example we can say:

$$p(S_i) > p(S_j) \Leftrightarrow \sum_{b_i \in FieryBuildings(S_i)} area(b_i) < \sum_{b_j \in FieryBuildings(S_j)} area(b_j)$$

The above definition is not complete and having civilians died of burning in mind the definition becomes more complex. We should also note that not every "Safe State" is reachable due to the conditions of disaster situation. So an optimal state cannot be clearly defined and we define the goal as reaching the state $S$ with the highest possible value of $p(S)$.

### 3.2 Priority of Actions

As defined in previous section, the task of Fire Brigades is reaching a "Safe State". Decomposing this task to smaller ones, we have tasks such as "terminating each region" and in the lowest level, extinguishing a fiery building.

It is obvious that in most of situations the agents have more than one task to do and as a result they should choose an arbitrary permutation of these tasks based upon a randomized or a well-defined policy and certainly, execution of tasks based upon different policies result in different final states which may be safe or not. In other words, policy $p_i$ is considered better than $p_j$ in a situation if and only if:

$$p(resultingState(p_i)) > p(resultingState(p_j))$$

Our experience shows that having well-cooperating agents, using a suitable priority scheme for extinguishing fiery buildings results in a dramatic improvement over using a randomized priority on the final score in most disaster situations.

So designing a priority scheme for fiery buildings is an important task. Runtime efficiency is also of primary importance in this field since computation of a priority value for all buildings may not be possible in many situations.

In our team, a two level priority scheme is used, which first gives priority to fiery regions. Once assigned to a region, a building priority is used by the agents to assign priorities only to the buildings of that region. As can be seen, two-level priority scheme improves performance of agents by increasing the locality of successive tasks and subsequently, causing agents do more tasks with minimum move actions. Besides, there are much fewer problems regarding the runtime efficiency for this priority scheme.

Last year our priority scheme for fiery buildings was a linear function of various parameters such as position of target building, effort needed to extinguish it, effort needed to extinguish its neighbors if it expands fire, number of alive civilians in its unburned neighbors and near buildings, its reachability, etc. Weights for the parameters were tuned using various statistical methods.

This year we are trying to use a Neural Reinforcement Learning based on Sutton's model[4] for achieving an optimal priority scheme. A rule-based analyzer is used for associating external rewards or penalties with actions in certain conditions and time periods. The analyzer uses our last year policy(as one baseline) for calculating rewards and penalties.

### 3.3 Cooperation

Cooperation is of primary importance in case of Fire Brigades as none of them is able to perform even a small task alone. For example to extinguish a medium size building, there must be four or five Fire Brigades extinguishing that building with maximum power, all at the same time. As a conclusion we should prepare means and design methods to achieve high levels of cooperation.

Our approach to solve this problem is a grouping strategy, i.e. agents are divided into groups. A group is a bunch of agents cooperating with each other to perform a predefined set of tasks. In every grouping strategy in Multi-agent systems, there are points and problems which must be addressed and solved. Some of these problems are formation, population management, coordination and task allocation for the group. Also, in some grouping strategies, a group may be deformed in some situations. In these situations, group deformation and reformation are also the problems which must be addressed. In the following sections we address each of these problems and describe the solutions used in our team.

**Group Formation** A group is formed when a free agent reaches a suitable condition to start a defined task. In the beginning of the simulation each agent searches for tasks and tries to put herself in a suitable condition to start a task as soon as possible. Reaching a suitable condition, the agent sends a message to the station requesting creation of a group. The station, receiving requests, accepts some of them and as a result, groups are formed and the requesting agent of each group is introduced as "Group Coordinator" for that group.

**Group Population Management** A group needs different number of members according to its allocated tasks in different situations. The number of needed members vary from 3 to 15 according to the characteristics of the allocated tasks. So a mechanism is needed for managing the popoulation of each group at the beginning of the simulation.

The first approach to be described is static population management. In this approach, a suitable population is estimated for groups and during the simulation the population for the group remains constant. The major shortcoming of this approach is the high possibility of wasting extinguish power and effort of fire brigades. But fortunately because its stability and ease of implementation, it will result acceptable performance, if a suitable population is estimated at the beginning of the simulation(An example of a team using this approach is S.O.S. 2003).

The other approach to solve this problem is dynamic group management. In this approach the group coordinator broadcasts the values of the function $f(WM, g_i, k, t)$ for all $1 \leq k \leq n$(number of fire brigades). $f(WM, g_i, k, t)$ is the benefit of the $k$th member for group $g_i$ in time step $t$ assuming the world model to be $WM$. It can be concluded that:

$$\forall k_1, k_2 \leq n \ni k_1 < k_2 \Rightarrow f(WM, g_i, k_1, t) \geq f(WM, g_i, k_2, t)$$

The group coordinator computes the values for $f$ based on estimation of the priority of its tasks and the number of needed agents in the next 10 to 15 cycles.

Values of $f$ is computed such that $f(WM, g_i, k1, t) > f(WM, g_j, k2, f)$ if the need of group $g_i$ for the $k_1$th member is more than that of $g_j$ for $k_2$th member in time $t$ of a specific simulation. Here if $g_i$ has $k_1 - 1$ members and $g_j$ has

$k_2 - 1$ members, then a free agent prefers to join $g_i$. Also if $g_j$ has $k_2$ members and there is no free agents and $f(WM, g_i, k_1, t) - f(WM, g_j, k_2, t) \geq \delta$ the $k_2$th member of $g_j$ leaves its group and joins $g_i$. Here $\delta$ is the coherency coefficient of groups. The bigger the value of $\delta$, the harder a group member leaves a group. $\delta$ can be viewed as $\delta(WM, g_i, t)$ which relates the coherency value to the state of the environment and group in cycle $t$.

While rule-based calculation of $f$ and $\delta$ may be suitable for simple situations, as the number of parameters grow and situations become more complex, computing a stable $f$ value, that satisfying the conditions described above, turns into an exhaustive and impossible task.

So we used a Reinforcement Learning method based on Q-Learning[3] to obtain an optimal $f$. Here awards and penalties are generated when events such as failure to extinguish a building, detection of effort over-estimation, detection of a chain of successful attempts and oscillation of values of $f$ in successive steps are detected.

**Group Coordination and Task Assignment** Once groups are formed, the most important goal of the groups is to cooperate efficiently in order to perform the assigned tasks of the group. To accomplish this goal, in its simplest form, the group coordinator broadcasts the current task of the group in each cycle and group members act regarding to this task.

In this approach, if the coordinator just broadcasts the current task in each cycle, upon finishing a task, there is at least a delay of 1 cycle between the time the coordinator decides on the next task and the time the group members receive and start performing it. This results in waste of time and resource.

A simple solution to this problem is broadcasting the task with the second priority in each cycle in addition to the current task. Group members receiving the pair of tasks hopefully consider the second task as the highest priority task if the current task is finished. So they immediately switch to the next task and therefore achieve high levels of cooperation in most of the situations.

## 4 Police Forces

Police Force agents are responsible for opening the blocked roads caused by earthquake in the beginning of the simulation. While their operation does not affect the score directly, they help other agents to move through the disaster space and reach their targets fast and easily and as a result they play a key role towards improving the quality of rescue tasks.

In this section we describe our approaches and solutions to the problems concerning Police Force agents. Here our focus is on the decision-making process and efficient handling of traffic jams.

### 4.1 Decision-Making Process

The behavior of our Police Force agents is formed by a mixture of centralized and decentralized decision-making systems.

In the centralized system, Police Office assigns a set of tasks to each Police Force agent based on the state of the disaster space and other agents. These tasks are divided into two major sets:

1. The set of tasks based on the available information about the state of the disaster space. For example in the beginning of the simulation, as information about the initial fire sources is received, Police Office assigns some Police Forces to clear the area around the sources in order to help Fire Brigades reach and extinguish them as soon as possible. It also assigns some Police Forces to clear some routes to the refuges having more strategic locations.
2. The set of tasks generated upon receiving requests to open a route or an area, from other agents. These tasks are prioritized based on parameters such as the priority of requesting agent, time of the request and some static rules(e.g. releasing an agent from a block is more important than processing a request for opening a route between two points).

Centrally-assigned tasks from Police Office to Police Forces is of major importance. Each Police Force agent receiving a task, dedicates some time(based on importance and size of the task) to do it and then switches back to its previous job.

In case of not being assigned to any task by Police Office, Police Forces make decisions in a decentralized manner. Each Police Force behaves in such a way to perform efficiently, according to its major goals.

The major goal of Police Forces is to clear a suitable permutation of roads while at the same time searching for fiery buildings and buried civilians, collecting information of world model updates and sharing this information with other agents via communication lines. A suitable permutation of roads can be defined as a permutation of roads that, if cleared in order, minimizes the number of requests of the agents.

In order to achieve the best team performance to accomplish these goals, agents model the disaster space as a grid. Then a "temperature" is calculated for each cell, based on the status of the agent and disaster space. The temperature of a cell is a estimate of the importance of the cell for the agent.

Once a temperature is calculated for each cell, the temperature distribution is smoothed over the grid. In the smoothing process, the smoothed temperature of a cell is defined as:

$$st(c_i) = t(c_i) + \frac{1}{16}\sum_{c_j \in neighbors(c_i)} t(c_j) + \frac{1}{48}\sum_{c_j \in level2Neighbors(c_i)} t(c_j)$$

During the simulation, when there is no task from Police Office, The Police Forces, in successive time intervals, choose the cell with the highest temperature and go to that cell, trying to decrease the temperature of the cell to an acceptable degree.

Semi-smooth distribution of temperature over the grid causes the Police Force going to a high temperature cell, to become close to other high temperature cells as well.

### 4.2 Traffic Jam

One important problem affecting the traffic flow in the disaster space is "Traffic Jam". Traffic jam occurs when the number of agents which decide to go to a road in a cycle exceeds the capacity of that road.

The strategy of Police Force agents to deal with traffic jam consists of traffic jam prevention and traffic jam elimination in the case it occurs.

To prevent the occurrence of traffic jam, Police Forces try to use different paths from each other, while moving, in order to minimize the probability of two Police Forces to choose the same roads, while trying to pass a region.

To do this, each agent at the beginning of the simulation, after making the grid, constructs a multigraph $G(V, E)$ with the set of Vertices $V$ equivalent to the set of grid cells and:

$$E = \{(a, b, r) | a, b \in V, r \in roads, head(r) \in cell_a, tail(r) \in cell_b\}$$

Then a set of spanning trees is found on $G$ such that their union contains all *strategic roads*(for a definition of strategic road refer to [5]). Then agents are assigned to trees such that each agent is assigned to only one spanning tree(the reverse may not be true). The agents trying to find a path, give high priority to the roads in their own spanning tree. This results in a remarkable decrease in occurrence of traffic jam while at the same time causes each strategic road to be visited at least by one police force agent.

On the other hand, for traffic jam elimination, each agent, after execution of a move command, calculates the distance traversed by that command and tries to estimate the degree of crowdedness of the route on which it is moving and determines a traffic jam coefficient for that route. The traffic jam coefficient is passed to the path finding module which considers it as an effective factor for the path-finding process.

## 5 Ambulance Teams

Ambulance Teams are responsible for rescuing citizens that are buried under the debris caused by collapse of buildings and saving their lives. Efficiency of their operation is of significant importance as it directly affects the number of alive citizens of the city in disaster which is the most important goal of a rescue team.

In this section we first present a formal definition of the problem facing ambulance teams and then solve a reduced version of this problem based on some assumptions. At last we present our efforts towards transforming the reduced version of the problem into the original one.

### 5.1 Goal Specification

Taking a closer look at the official evaluation formula for RoboCup Rescue Simulation League[7]we can realize that Ambulance Teams must first make use of a policy $P_i$ that maximizes the number of alive humanoids. Between two different policies $P_i$ and $P_j$ that result in equal number of alive humanoids one is considered better that results a greater value of $S_{HP}$ where:

$$S_{HP} = \sum_{h_i \in aliveHumanoids} HP(b_i)$$

Furthermore, we should consider the situations where some members of the rescue team are also buried. Then it is mostly obvious that ambulance teams should give a higher priority to rescuing these agents than that of other humanoids. The proplem becomes even more complex if we note that the policy chosen by Ambulance Teams to rescue them must be, in such a way, to maximize the overall performance of the whole rescue team.

## 5.2 Reduced Version Of The Problem

Now, we want to take some assumptions that result in a problem which can be solved efficiently. Then we use the solution to this reduced problem to attack the original one. These assumptions are:

1. None of the members of the rescue team are buried.
2. The information about the status(place, HP, damage, buriedness) of each civilian is available to Ambulance Teams.
3. Having the status of a specific civlian, the exact time when it would die if not rescued can be computed exactly.
4. The time that it takes for an agent to transfer between two known positions is equal to zero.
5. Once a civilian is rescued, there is no need to transfer it to a refuge.

Now the problem is:

*Given these assumptions and having n buried civilians and k Ambulance Teams, a policy is needed which maximizes the number of alive civilians at the end of the simulation.*

*Solution* : First of all, we ignore all civilians that will stay alive until the end of simulation without the need to be rescued. The remaining ones are sorted by *deadline* and indexed from 1 to $n$(*deadline* is defined as the maximum time before which should Ambulance Teams reach a civilian in order to be able to rescue it). Now a dynamic programming approach is used to choose the set of civilians to be rescued:

$$p(i,t) = \max \left\{ \begin{array}{c} 1 + p(i+1, t + rescueTime(i)) \\ p(i+1,t) \end{array} \right\}$$

Where $p(i,t)$ is maximum number of civilians starting from the $ith$ one that can be rescued from time $t$, and $rescueTime(i)$ is the duration of rescue operation for the $ith$ civilian.

Note that $p(n+1,t)$ is zero for all $t$'s. Also $p(i,t)$ is equal to $p(i+1,t)$ when $t > deadline(i)$. Our target is to compute $p(1, InitialTimeStep)$.

It can be proved that having all Ambulance Teams work in a single group( $rescueTime(i) = \lceil buriedness(i)/k \rceil$) maximizes the number of alive civilians at the end of the simulation.

### 5.3 Towards the Original Problem

Now, that the reduced problem is solved, a deeper look at each of the assumptions is required in order to reach a generalized solution.

**Buried members of Rescue Team:** As stated before, in most of the situations, serving members of the rescue team is of higher priority than that of civilians. In our team, Ambulance Teams, using a predefined policy, rescue all buried members of rescue team and then start their expected rescue task. Here the goal of our dynamic function becomes $p(1, InitialTimeStep + t_r)$ where $t_r$ is the duration of rescue process for buried members of rescue team. Here, we should note that this would not always result the best score. Sometimes, ignoring a member of the rescue team or assuming it a simple civilian would result a better score but unfortunately, this case is very complex to detect and handle.

**Status of buried Civilians:** In a real situation, our second assumption is not true and agents should search the disaster space in order to find buried civilians. To accomplish this, Ambulance Teams divide the disaster space into regions and try to find buried civilians when having none to rescue. In addition, Fire Brigades and Police Forces also report the buried civilians found while doing their jobs and after finishing their job, they perform a full search of the entire disaster space for the civilians and report the results to Ambulance Teams.
Ambulance Teams, each time, using the method we described, try to find the best civilian to rescue and start rescuing it. As the information about the status of civilians is updated, the sequence of civilians to be rescued is also updated. In the special case, where the sequence is updated while Ambulance Teams are rescuing a civilian, the priority of that civilian is compared between the last sequence and the current one and if it is of a lower priority in the current one, and if the difference between its priorities in the last sequence and the current one is more than a special threshold, Ambulance Teams leave it and switch to the most important civilian according to the current sequence.

**DieTime Estimation:** As can be seen our approach is based on calculation of dieTime($i$). To estimate the function, we designed a 3-layer Back-Propagation Neural Network[2] with normalized values of HP, damage and $\frac{\Delta d}{\Delta t}$ where $\Delta d$ is the difference between the initial damage value of the civilian and the most updated value and $\Delta t$ is the difference between these two times. For better performance of the Neural Network, we also added another parameter which was the estimated value of deathTime($i$) based on regression methods. After training the network with data of about 1000 buried civilians in 300 or more time steps, the Neural Network was able to estimate the value of dieTime($i$) with at most 5% error.

An important thing worth noting about esitmation of $dieTime$ is that in the beginning of the simulation, having initial status of a civilian, a good estimationof $dieTime$ is not possible. So, as the simulation continues, agents should check the civilians that were found and update their status in order to have better estimations.

**Transfer Time:** the assumption that causes most changes to the reduced solution, if not considered, is the fourth one. Taking into account, the transfer time between two places, the formula of the dynamic programming approach must be changed into:

$$p(i,j,t) = \max \left\{ \begin{array}{c} 1 + p(i+1, i, t + transferTime(i,j) + rescueTime(i)) \\ p(i+1, j, t) \end{array} \right\}$$

Where $p(i,j,t)$ is the maximum number of civilians, from the $ith$ one, that can be rescued from time $t$ given the fact that the last rescued civilian is the $jth$ one. $transferTime(i,j)$ is the time it takes for an ambulance team to move from the position of the $ith$ civilian to the $jth$ one. $transferTime(-1,i)$ for all $i$'s is the time it takes for an ambulance team to move from its current position to the position of the $ith$ civilian. Now, our target becomes $p(0, -1, InitialTimeStep + t_r)$.

Here, we have two new problems. First, a reasonably good estimation of $transferTime(i,j)$ is needed which is not easy to compute. But second, and certainly the more important one, is that this new policy assumes that all the Ambulance Teams act as a single team and gives the best rescue sequence with this assumption. It should be noted that if Ambulance Teams sometimes divide into 2 or more teams they can save more civilians.

Our efforts towards designing a good strategy of when, where, and how to divide into more than one teams, has resulted in some general solutions which are still costly to use, with respect to time and memory but also resulted some solutions which would not be characterized as the best policy but in most situations work better than the original single-team solution.

## 6 Conclusion and Future Works

This year, having enough experience and appropriate set of tools, our focus was on making our agents more robust, adaptive and cooperative by designing and implementing learning schemes and algorithms and utilizing major concepts of uncertainty.

Our future works are attempts to make our agents' cooperation schemes more decentralized and communication-independent. We are also working on some tools and APIs to help agent developers in the process of designing, implementing and validating agent strategies and also dedicating some effort to implement new simulators for RoboCup Rescue Simulation System.

We hope that someday we would be able to see real autonomous rescue robots cooperating optimally to minimize the consequences of disasters.

## References

1. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer-Verlag (2001)
2. Haykin, S.: Neural Networks: A Comprehensive Foundation. 2nd edn. Prentice Hall (1998)
3. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
4. Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuron-Like Adaptive Elements That Can Solve Difficult Learning Control Problems. IEEE Transaction on Systems, Man and Cybernetics (1983)
5. Javanmardy S. et al: Team Description of S.O.S. `http://ce.aut.ac.ir/~sos/downloads.htm` (2003)
6. Stentz, A.: The Focussed D* Algorithm for Real-Time Replanning. Proc. of Int'l Joint Conf. on Aritificial Intelligence (1995) %
7. RoboCup 2003 Rescue Simulation League Technical Committee: Rules of RoboCup 2003 Rescue Simulation League. http://sakura.meijo-u.ac.jp/ttakaHP/kiyosu/robocup/Rescue/2003/Rule/2003-rule-3.html (2003)