Poseidon Team Description Paper
Robocup 2011, Istanbul

Romina Abadi[1], Bahar Behazin[1], Mahya Eslami[1], Seyede Fatemeh Ghasemi Zavie Sadat[1], Bahere Moradi Dadkhah[1], Rozhina Pourmoghaddam[1], Mahta Ramezanian[1], Zohre Rezayat[1], Niloufar Seyed Majidi[1], Nasim Shirvani-Mahdavi[1], Anooshik Vartanian[1] and Pooria Kaviani[2],

[1] Farzanegan High School, Robotics Research Group, No. 56, Shahid Sarparast St., Taleghani Ave., Tehran, Iran
[2] Sharif University of Technology, Department of Computer Science, Azadi Ave., Tehran, Iran
{prof.rominaa, b.behazin, mahya285, ghasemi.zahra33, bahere.moradi, khonakaye.baroon, mahtaa, zo.rezayat, nilufar.majidi, nsmahdavi1995, kishoona94, Pooria.kaviani}@gmail.com

Abstract. This manuscript describes the algorithms and contribution of Poseidon 2011 to acquire the optimum and efficient solutions for rescue agent simulation problems. In this team, because of recent changes of the server, new WorldGraph structure and Communication system were implemented. Each section, explains new algorithms of this year in details. This version of Poseidon team is based on the sample code of the server with extensive changes in structure of agents and WorldModel. In Poseidon 2011, we have used computational geometry and artificial intelligence algorithms for solving the problems of rescue agent simulation.

1 Introduction

In real world, natural disasters, such as earthquakes, floods etc., cause a variety of problems, ranging from partial to complete destruction of communication lines, roads, bridges and airports, fuel shortage and fire. In order to come with the solution, a simulation is carried out, without casualties and at next to no costs. At first, World-model structure and World-Graph algorithms are described, followed by a description of radar system and related algorithms. Finally each agent's decision making and strategies are explained. All these tasks need a sagacious management [1].

2 World-Modeling and Communication

We should design a structure to store the data that server sends us about roads and buildings. This structure should be able to calculate the path between areas of the map, where there might be no path between some areas because of blockades. To do this, we specify an entrance object for each of entrances to areas. For example when two areas are neighbor in one edge, two entrances will be created in both areas for that edge. We check the entrances'

connectivity to find out if any path exists between areas. For knowing that if two entrances are connected together or not, we have implemented robot motion planning algorithms. So for each pair of entrances in an area we should determine that one agent can move from one entrance to another one without hitting blockades of this area or not. Size of each agent can be problematic at this point. To study this problem carefully, we treat the agent like a point [2]. So we expand blockades and area's frontiers, and then reduce agent's size to a point. Fig1 is an example of this expanding.

There are four logical methods to know where to go without hitting blockades in a map [2]: Roadmap approaches (such as Visibility Graph and Voronoi Diagram), Cell Decomposition, Potential Fields, Bug algorithms.

Server uses Potential Fields method. This method has some merits, for example it generates smooth paths to find a way. But it has some bad points too. For example an agent can be trapped in local minima areas. According to [2] our agent can use random-walk to escape from those areas. But random-walk is not perfect, so we decided to implement another method to decrease these difficulties. Visibility graph was a good method for this problem. In this method, each corner of area and blockades of that are nodes of a graph. Also we add start and goal point to these nodes. Two nodes of this graph are connected together, if the line between these two nodes does not have any intersection with blockades of that area. Now our geometric problem changed to a graph problem. There are many algorithms for checking reachability of two nodes in a graph. So we can now compute the connectivity of two entrances in world-graph. Fig1 shows the steps of path finding between start and goal points.

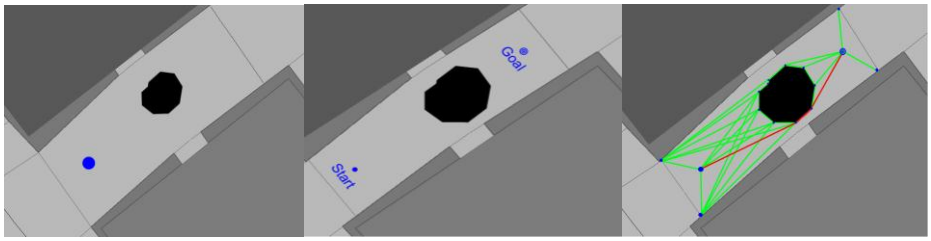

Fig1.Expanding blockades and visibility graph of area

Communication is an important factor in each team. Sharing the information with other agents is necessary. The buildings that are ignited, the civilians, the agent's help requests, the decision of police and ambulance centers, the accessibility of roads will be shared between agents. To send this information, we encode them to byte array and then send them. When we hear it, we decode the message with same algorithm and update that information.

To demonstrate the information about connectivity of entrances of a road, we make a table by the number of entrances and fill it with 0 and 1. 0 means there is no way between them and 1 means that there is a way between them. For example we have got a table like this for a road with 7 entrances:

	0	1	2	3	4	5	6
0	1	1	0	1	1	0	0
1	1	0	0	1	1	0	0
2	0	0	1	0	0	1	0
3	1	1	0	1	1	0	0
4	1	1	0	1	1	0	0
5	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1

In this table, the number in 0-0 cell shows the entrance 0 connectivity to its neighbor entrance (Like all the yellow cells). Also we do not need to say green cells because they are repetitive. For example if entrance number 0 has way to entrance number 1, certainly entrance number 1 has way to entrance number 0. There is no need to send all

another half too. For example if we know that entrance 0 has a way to entrance 3 and it has a way to entrance 4 too, we understand that entrance 3 has a way to entrance 4. So we do not need to send purple cells. We just need to send blue and yellow ones. They are 15 bits in this example. That means we send 15 bits instead of 49 bits which means sending 2 bytes instead of 7 bytes.
Each agent sends its around buildings information. So finally we have an enormous message including all the around buildings' ids and entrances. To truncate this message, in pre-computation, we determine roads which we can see from each area. Then we give a number to each of them. So when we want to send blockades' information, instead of sending an id with six digits, we just send the number of it that maximally has 2 digits.

2 Ambulance Team Agent

Due to the importance of the civilians' life in real world, in simulation, minimizing human casualties by use of ambulance and emergency services is vital. This must be achieved by developing ambulance performance and improving decision making. There are two situations for maps, maps which have communication and maps without communication. In maps without communication each ambulance agent searches to find the position of civilians. When an ambulance agent finds a civilian, it goes to the position of that civilian for rescue. In maps with communication, one agent (such as center building if exists or an ambulance agent) decides for all ambulances and informs them about their tasks via radar. With this type of decision making, tasks of two agents cannot be the same, because one specific agent, who called center, decides instead of all agents. For this, center needs reliable information from civilians' conditions and ambulances' locations in the map. So, center gathers this information from agents searching the map, by radar. After some computation on these data, center can provide a task for each agent. Then center should inform agents from their jobs by radar. These tasks should be updated because map information is subject to change during a simulation.

When we studied the ambulance issue we realized that for solving this problem we should have an efficient model of job-scheduling to apply the Artificial Intelligence algorithms on it. If we have a simple and complete model of job-scheduling, AI algorithms become more optimized. An easy way to represent a model is one 2D array. The dimension length of this array is the number of ambulance agents in the map and each row of this array has 300 cells. Each row in the map represents the tasks of one agent, and each cell of a row, stores the civilian that should be rescued in a specific cycle. We implemented this model and applied the ACO algorithm on it in Poseidon2009 [5]. But this representation model is not memory-efficient. This great size makes the search harder and takes a long time to search and apply AI algorithms.

To make search easier we have defined a more efficient model. There is an array in this model with civilians' number length. Each cell of this array corresponds to one civilian and represents the agent which has been selected for that civilian. So there may be some cells with the same agent. Now that agent, who is in more than one cell, should sort its cells and rescue them in order. This is more memory-efficient. So we can search easier and it takes less time than before. For example a 1D array filled with these numbers: {31258, 22579, 31258, -1, 14587, 14587} represents the tasks of 3 agents for a map with 6 civilians. Agent 31258 will rescue first and third civilian and forth civilian cannot be rescued.

Before describing our strategies for assigning jobs to each agent, we should describe some definitions.

1. Dead Time: “Dead Time” is a property of civilians. It is an estimate of the civilians’ death time. For example if an agent dies in 5 cycles later, value of this variable would be 5.

2. Free Time: “Free Time” is a property of ambulance agents. It is an estimate of a time that ambulance agent has no task to do. For example, when we assign a civilian to one agent, this property will be changed. The change of this variable depends on the buriedness of that civilian and estimated time for carrying that civilian to its nearest refuge.

Now the main problem is finding the best 1D array which describes the tasks of agents. A lot of permutation of tasks for agents is available, but we should find the best state. To solve this problem we use Genetic Algorithm. According to [3] “The GENETIC-ALGORITHM, which starts with a set of one or more individuals and applies selection and reproduction operators to "evolve" an individual that is successful”. Based on [3] there are some definitions in Genetic Algorithm. “The fitness function depends on the problem, but in any case, it is a function that takes an individual as input and returns a real number as output that represents the value of individual.” Here is description of individuals and fitness function, which is needed for genetic algorithm.

1. Individuals: Each individual is an array describing the tasks of agents. This array is sequence of civilians and describes a job-scheduling. To convert the sequence into job-scheduling, at first we assign the nearest ambulance to the first element of the state. So, summation of the time that ambulance agent spends to reach the civilian, rescue them and carry them to the nearest refuge, should be added to that ambulance free-time. Now we find the nearest ambulance to next element of the state. If free-time property of the nearest ambulance was more than dead-time of that civilian, we should not assign this ambulance to that civilian, because it does not have enough time to rescue them. So we should find the next nearest ambulance to that civilian and check this condition again.

2. Fitness: Fitness is the number of rescued civilians in each state.

The first step of genetic algorithm is creating the first generation, after that we remove improper individuals and then create new generations by using cross-over and mutation.

a. Creating first generation

In this step, 50 random permutations of civilians will be produced and stored in 50 states.

b. Removing improper individuals

In each generation half of the population with less fitness will be removed, because they are not close to the best result.

c. Cross-over

Cross-over in our model is a process that combines individual pairs. All the individual pairs are selected randomly but some of them have more chance to be selected for cross-over because of their fitness. So we “cross-over” these individual pairs to make better states. In each individual pair, each state takes part in making offspring depends on its fitness. For example to cross-over two individuals which have these selecting sequences :{8, 1, 6, 2, 7, 4, 5 ,3}, {6, 3, 2, 8, 5, 4, 1, 7}, “Selecting Sequence” of their offspring will be like this: {8, 1, 6, 2, 7, 3, 5, 4, 1}. In this model the first individual has more fitness, so it takes five cells of the offspring and the rest cells should fill with cells of the second individual based on their appearing in sequence.

d. Mutation

For mutation, a few of states will be selected randomly and swap two cells of its Selecting Sequence. For example if the selected individual has this “Selecting Sequence”: {2, 3, 8, 5, 1, 7, 4, 6}, after mutation it would be like this: {2, 7, 8, 5, 1, 3, 4, 6} and it means the second and the sixth cells of this array selected randomly and swapped.

As much as it takes time to make new generations, we achieve better results. But we have limitation of time. So, we run the algorithm concurrently with main program for a few cycles and then we inform the obtained state by radar to agents.

Sometimes an agent hears nothing for several cycles. In these situations agent does tasks of it the same as when it is in a map without communication.

3 Police Force Agent

The most important duty of police force agent is clearing the roads. It also helps the other agents by searching the map and updating the information. The police force agent duties effect directly on other agents' works. In order to clear the roads, police force agents have some priorities. First they try to open the way for trapped agents. Then they clear the path of all agents to refuges. After that if there were burning buildings, they will open the path to that building. Afterward, they will start to search. Following tasks are intended for police force agents:

1. Zoning
2. Clearing the way of agents, which have no way to any refuge, to the nearest refuge
3. Clearing the way of fire brigades to fire areas
4. Searching in their zones to find civilians, ignited buildings and trapped agents

For zoning we use Poseidon2009 algorithm. In this structure we divide the map into same sectors which contain almost the same volume of roads and buildings. Then we determine a police force agent for each part. The advantage of this method is that the time needed for all agents is almost the same. For all agents we use this zoning. To assign a zone to each police force agent, minimum weighted bipartite matching algorithm [4] is used. A minimum weighted bipartite matching is defined as a perfect matching where the sum of the values of the edges in the matching has a minimal value. In this model, head of each edge in graph is one agent and tail of that is a zone. The value of each edge is distance between endpoints of it.
After zoning, the first priority of police force agents is opening a path from trapped agents to the nearest refuge. If each agent decides by itself, it will make some problems in coordination of agents. In order to reduce the time of this task and improve the coordination between agents, we have centralized decision making. Each agent which has no way to any refuge will send the help request by radar to the center which is chosen at the start of the simulation. The center will manage all police force agents so that they open the roads as fast as possible. After opening the roads, each police force agent starts to search in its zone. The volume of areas for searching for each police force agent is the same; because the volumes of their zones are the same. So that we can find all civilians, open the roads to burning buildings and trapped agents as soon as possible.

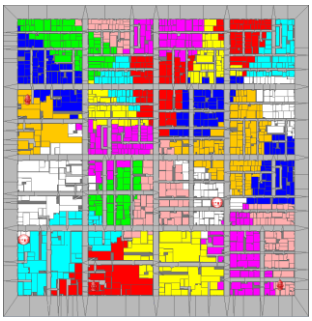

Fig2. Zoning algorithm

4 Fire Brigade Agent

The Fire Brigade agent is one of the most important agents because of its influence on preventing from fire spreading. One of their basic tasks is of course, extinguishing the fire. To control fire spreading, selecting the buildings to extinguish by each agent is substantial. For setting each fire brigade agent's task, we should make some groups of burning buildings (4.1 zoning) and assign a group of fire brigades for each. So now each fire brigade agent should select a burning building to extinguish it.

4.1 Zoning for burning buildings

The previous zoning algorithm of POSEIDON team (POSEIDON2009 for previous server) tried to divide the map into some parts (with some horizontal and vertical lines) that the number of parts was related to size of the map. But there were problems, i.e. some burning buildings were considered in different zones although they were connected to each other; however, in the newer zoning algorithm, we have overcome this problem. In this new zoning algorithm, each zone contains some burning buildings. Buildings of each zone are connected to each other, and there is not any unburned building between them. Fig3 and Fig4 show the differences between these two algorithms.

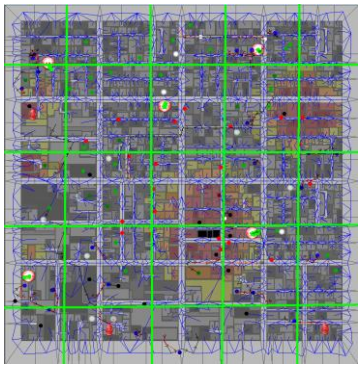

Fig3. Last zoning algorithm for fire

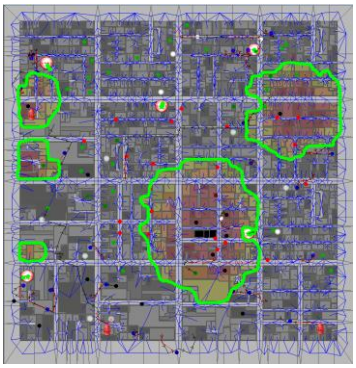

Fig4. New zoning algorithm

4.2 Grouping the Fire Brigade Agents

To choose a fire target for each fire brigade agent we should determine which zones and other fire brigade agents a fire brigade agent has access to. We have groups, each containing some fire brigades and some zones. Zones have different priorities for being chosen.
First we dedicate some fire brigades to small zones; then we leave the unimportant zones. After that we set value for other zones. The number of fire brigades that we assign to each zone is according to their values. Then if there were any fire brigades left we dedicate them to unimportant zones. To choose the most appropriate zone, we used the effects of several factors such as: number of around buildings of each zone, location of each zone in the map, numbers of civilians in each zone, amount of fire spreading in the zone and estimated time that is consumed to extinguish or control the fire in the zone [5].

4.3 Finding around buildings

For setting the value of each zone, the number of its surrounding buildings is needed. To find the surrounding buildings of each zone we obtain convex hull of buildings of zones (Fig5). Fig6 shows the final state of map after implementing this algorithm. "In computational geometry the convex hull or convex envelope for a set of points *X* in a real vector space V is the minimal convex set containing *X*. “The convex hull is then typically represented by a sequence of the vertices of the line segments forming the boundary of the polygon, ordered along that boundary" [6].

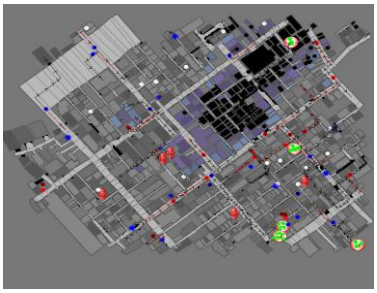

Fig6

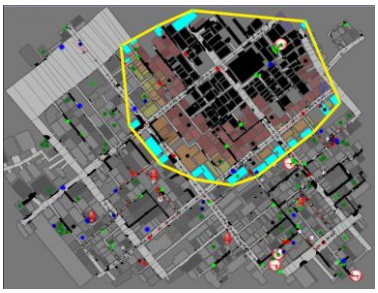

Fig5. The aqua buildings are around buildings

4.4 Set priority for choosing building in each zone

The value of buildings shows that which building should be chosen. It depends on the volume of each building, the number of floors, the distance between that building and the fire brigade agent and the number of other near buildings that are not burned, etc. To select the buildings to extinguish, buildings with minor values have more priority.

Building's value = W_1 * (volume of building) + W_2 * (the distance between the fire brigade and the building) – W_3 * (number of unburned near buildings)

4.5 Search

The search structure is for time that there is not any fire to extinguish or when fire brigade agent does not know where other ignited places are in the map. We assign a variable (LastTimeSeen) for each building that shows the last cycle that a fire brigade has seen it so the buildings which have less LastTimeSeen have more priority to be searched. Also buildings which are around a burning building have more priority to be searched.

For searching in maps without communication, we assign a fire brigade agent for each part of the map. For choosing a specific agent for each part we use zoning algorithm. Our new zoning is somehow similar to police zoning structure; however, but they have some differences. In fire brigade agent zoning according to difficulty and size of each map, we choose a time for fire brigades (in each group) to meet each other. They gather in an area which is the nearest one to the central point of each zone to exchange information. Fire brigades which are closer to the central point of the map meet each other in the closest area to there (central point of map) and all of them exchange their data and information.

References

1. Abadi, R., Behazin, B., Kaviani, P., Ramezanian, M., Rezayat, Z., Shah-Hosseini, M., Vafadoust, M., Vartanian, A.: Poseidon Team Description Paper, 2010
2. Xiao, J. Introduction to Robotics, Robot Motion Planning, Department of Electronic Engineering City College of New York
3. Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, Englewood Cliffs, New Jersey 07632 (1995)
4. Douglas B. West, Introduction to Graph Theory (2nd ed.), Chapter 3
5. Afzal, A., Arian Nezhad, M., Mosadegh, Z.,Shah Hoseini, M., Vafadoost, M.: Poseidon Team Description Paper, 2009.
6. (http://en.wikipedia.org/wiki/Convex_hull) last visited 25th of march 2011