RoboCup 2011
Rescue Agent Simulation Competition
RMAS_ArtSapience Team Description Paper

Amr Labib Hussein, Ahmed Abouraya, Dina Helal, Noha Khater, Mina Fahmy, and Carmen Gervet

German University in Cairo, Cairo,Egypt,
amr.labib-hussein@guc.edu.eg, [ahmed.abou-raya, dina.hussein, noha.khater, mina.adel]@student.guc.edu.eg , carmen.gervet@guc.edu.eg

Abstract. This paper describes the contribution to the Rescue Agent Simulation by the GUC RMAS_ArtSapience team; in terms of the current research approach implemented to prepare for the RoboCup 2011 competition. The approach is divided into two parts: clustering and planning. Fuzzy c-means clustering is used in preprocessing to divide the map into regions and assign a group of agents to each region. Then path planning and routing operations are performed using a graph library from the ECLiPSe constraint logic programming system. The outcome is a path for each agent in his allocated cluster which is a map. The approach provides an efficient methodology to divide the work among the different agents and to efficiently traverse the whole map in order to perform the rescue operations.

1 Introduction

Rescue planning and optimization is one of the emerging fields in Artificial Intelligence and Multi-Agent Systems. The RoboCup Rescue Agent Simulation provides an interesting test bench for many algorithms and techniques in this field. The simulation environment provides challenging problems that combine optimization (routing, planning, scheduling) and multi-agent systems (coordination, communication, noisy or missing communication).

The Robotics and Multi-Agent Systems (RMAS) research group at the German University in Cairo (GUC) was established in September 2010. The goal of the research group is to study and develop AI algorithms to solve problems in robotics and simulation systems. These fields include computational intelligence, constraint programming, computer vision, multi-agent systems, and classical AI approaches. The current research efforts investigate the following research directions:

- Clustering as a preprocessing step for multiple depot routing.
- Comparison of k-means and fuzzy c-means clustering algorithms.
- Implementation of routing algorithms using constraint programming technology.

- The use of constraint programming to generate and update routes for path planning agents.

The GUC RMAS_ArtSapience team is making its first participation to the Rescue Agent Simulation in 2011. This paper describes the current team's achievements in tackling the RAS problem. Section 2 describes our modeling and algorithmic approach. Section 3 describes the agent implementation Section 4 provides some empirical results. And finally section 5 outlines the ongoing tasks that are underdevelopment.

2 Our Approach

This section describes the approach we used to address the Agent Challenge. The first part discusses the use of Fuzzy C-means clustering in dividing the map into regions. The second part discusses map modeling and path finding using a Constraint Logic Programming system.

2.1 Clustering using Fuzzy c-means

The c-means clustering algorithm computes a collection of clusters based on a membership function that we define. In contrast with the k-means algorithm, it enables one to attach a data point to more than one cluster. The output of the c-means clustering algorithm is a set of fuzzy partitions with all point in the data belonging to them with relative membership values. A certain threshold is applied to the membership values and the final partitions are generated [4].

Clustering is used in this context as it provides a means of distributing the rescue operation on the different number of agents to cover the whole map. Fuzzy c-means was chosen as it will guarantee an overlap between the generated clusters. The overlap is required because it is not guaranteed that the agents assigned to one region can reach all roads and buildings due to the unexpected blockades, especially the ones at the edges of each region. The overlap will help overcoming this problem as agents from different regions will try to reach these areas from different paths, which will guarantee a higher chance of reaching them in time.

The first step in the preprocessing is to apply fuzzy c-means on the map. A list of all roads and buildings is given to the algorithm. The number of generated partitions is determined by the number of available agents and the map is clustered differently for each type of agent. The generated partitions represents regions in the map. Each region will have a group of agents responsible assigned to it. The agents will then carry their designated operation in its region as discussed later in this paper.

2.2 Map Modeling and Constraint Programming

Constraint Logic Programming

Constraint Logic Programming (CLP) is a quite recent programming paradigm with roots in Artificial Intelligence that combines two declarative paradigms:

constraint reasoning and logic programming. CLP has been very successful in the past 20 years to tackle combinatorial problems in fields such as planning, scheduling, timetabling, numerical analysis and more recently bioinformatics [2].

Our main motivation behind the use of constraint programming is its modeling abstraction, and the separation it offers between problem modeling and solving. The main task the user is required to fulfill is to be able to model the given problem as a constraint problem, in terms of variables, domains attached to these variables, and the constraints they must satisfy. The constraints are then resolved through the use of a constraint solver. In our case, we will model a map as a graph using facilities of a CLP system, called ECLiPSe [3]. This enables us to use the constraint solving engine associated with graph constraints such as finding one shortest path between two graph nodes, finding all shortest paths between a given set of points, discovering regions using breadth first search.

Map Modeling

The following describes how a map is modeled in a CLP setting:

- All Areas (roads and buildings) are considered to be graph nodes.
- The lines connecting neighboring areas are considered to be edges.
- The weight of each edge is its length.
- Every edge is created twice in order to have an undirected graph.

Using the graph model created, it is possible to use the graph algorithms provided by ECLiPSe to compute the best routes for each agent in its region. All agents can connect to the ECLiPSe engine using their own graph model and can dynamically calculate new routes efficiently during the simulation run, which makes it easier for agents to recalculate paths in case of undesired blockades.

3 Agent Implementation

3.1 Tools and Packages

The following is the list of tools and packages we used in the implementation:

- Java is used to implement the core system and the fuzzy c-means clustering algorithm.
- Agents are based on the rescuecore2 StandardAgent provided with the simulator.
- The ECLiPSe Constraint Logic Programming System is used as the CP engine, mainly the `graph_algorithms` library. It is a Prolog based system.

3.2 Agents

Figure 1 shows the general agent structure implemented in all types of platoon agents. The following algorithm explains the general structure followed by the specifics of each agent.

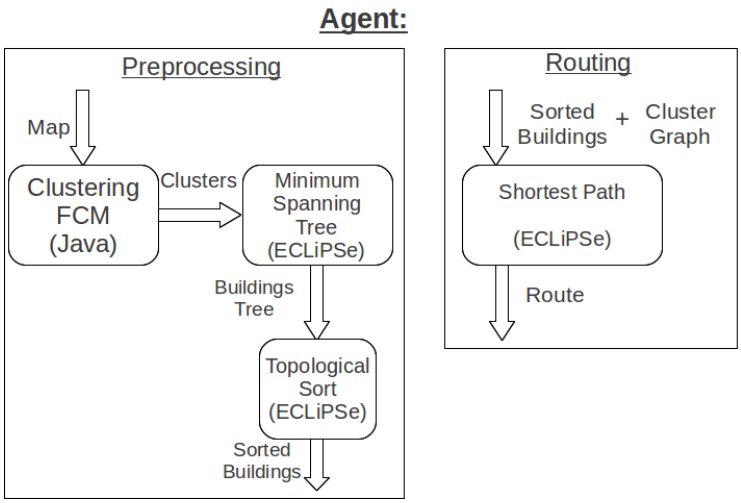

Fig. 1. General Agent Structure.

- **Shared Preprocessing:**
  - The map is divided into regions using fuzzy c-means clustering.
  - Each region is assigned a group of agents that will be effective to search through its roads and buildings.
  - The number of agents per region is proportional to the number of buildings and roads in the region.
- **Individual Preprocessing:**
  - Each agent retrieves all regions and finds the region it is assigned to.
  - The agent calculates the minimum spanning tree of the buildings in its region.
  - The agent finds the topological sort of the buildings tree.
- **Runtime (Think):**
  - Each agent will calculate its route by finding the shortest paths between the buildings in the order computed in the topological sort.
  - The agent will follow the route searching for events that requires rescue actions.
  - If the route is blocked, the agent will try to find alternative routes or will skip buildings.
  - If the agent finished a complete cycle in its region, covering all buildings and roads, and no events were found that required any action, the agent will move to a different region and repeats the same process starting with finding the topological sort of the new region.
  - If a communication exists, each agent will listen for informative and query messages (explained in section 4) with events that requires action. In the case of informative messages, free agents will respond if the event is within its region. An agent is considered free if it is not doing any rescue

action. In case if query messages, free agents will respond immediately to events in its region. For events outside the region, free agents will respond in case they finished scanning their regions.

Ambulance Team

Ambulance teams are interested in finding buried civilians. As each agent follows its route, it will keep scanning the building for civilians. When buried civilians are found, the rescue operation will begin. Each agent will keep track of unvisited buildings. Once a building is visited, it is removed from this list. If a building has several buried civilians, the agent will send a query message and keep track of this building to give it higher priority in the rescue operation. If all buildings are visited and no more civilians need to be rescued, the ambulance team does not have to do any more work.

Fire Brigade

Fire brigades will follow the same plan as the Ambulance Team but they will not need to enter the buildings, except when necessary. Each agent will scan all the buildings in its route and check for fires. Once a fire is found, the agent begins extinguishing. When it runs out of water, the agent goes to the nearest possible refuge to refill, keeping in mind any location it has passed that was on fire. If the agent finds several buildings on fire in its line of sight, it will consider the area to have a spreading fire. It will send a query message to ask for help controlling the spreading fire.

Police Force

Police forces performs an extra step in both preprocessing and runtime. It was noticed that many agents are stuck at the beginning due to collapsed buildings and blocked roads. To solve this problem, Police agents start by clustering the initial locations of all fire brigades, ambulance teams, and refuges, in addition to clustering the map. Each agent is assigned to two clusters: one for the agents and refuges and the another for the map regions. Police forces start by clearing the roads to agents and refuges locations to make sure that all other agents can move freely (unless initially buried) and that all refuges are reachable. After this operation, each agent goes to its designated region in the map and start clearing the roads. Police forces will respond to query messages sent by stuck agents in their regions. If an agent's region has no more blockades, the agent can move to another region and/or respond to query messages in different regions.

4 Communication Model

Agent communication is built on top of the channel communication model provided by the rescue simulator. Each agent subscribes to a channel designated

only to agents of the same type. Communication messages are based on an abbreviated form of the small Agent Communication Language (sACL) [6]. The types of messages used are as follows:

- **Informative Messages** `inform<id,source,message>`: intended to inform other agents of sensed fires, blockades, and buried civilians. Informative messages are sent by agents that detect events they cannot act up on. For example, when a police force agent senses a buried civilian, it will send the appropriate informative message to the channel the ambulance team should be listening to. On the other hand, a police force agent will not send informative messages to other police force agents when it senses blockades. Informative messages do not require any acknowledgment and are only broadcasted once.
- **Query Messages** `query<id,source,message>`: intended to ask for help from agents of the same type as the sender. Query messages are sent when agent realizes that it cannot perform a rescue action on it's own. For example, an ambulance team agent may find itself in a building with many buried civilians. The agent will send query messages to all ambulance teams asking for help. Query messages are also sent when an agent is stuck and needs police forces to clear the blockades around it. Query messages are sent three times to guarantee message delivery and requires acknowledgment. If the agent does not receive an acknowledgment in a specific amount of time, the message is resent again.
- **Acknowledgments** `acknowledge<id,source>`: intended to acknowledge a response to a query message with the same ID. When an agent receives a query message, it will send an acknowledgment if it decides to respond to the query. If the agent is not capable of responding, it will not send any reply.

5 Test Runs

Most of the test runs were made using the 2010 competition maps and scenarios. Test runs were made on a single machine with the following specifications:

- CPU: Intel i7 1.6 GHz
- RAM: 6GB
- OS: Ubuntu 10.10
- RoboCup Competition 2010 Rescue Simulator

Figures 2 and 3 show the final state of the map Kobe1 in the map viewer and the final score in the score chart.

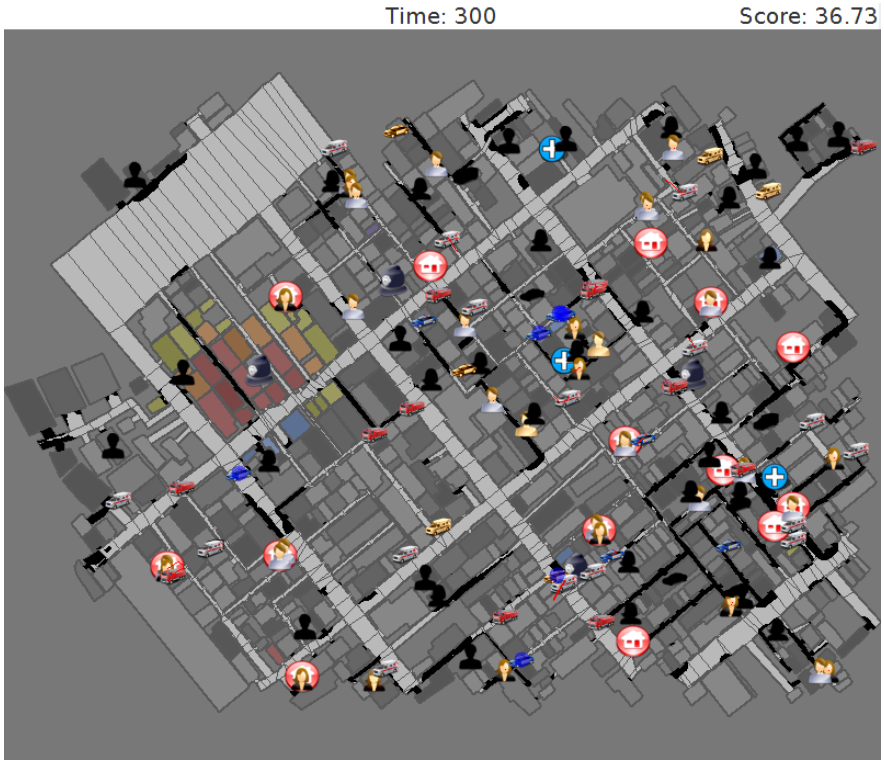

Fig. 2. Final state of Kobe1 map in map viewer.

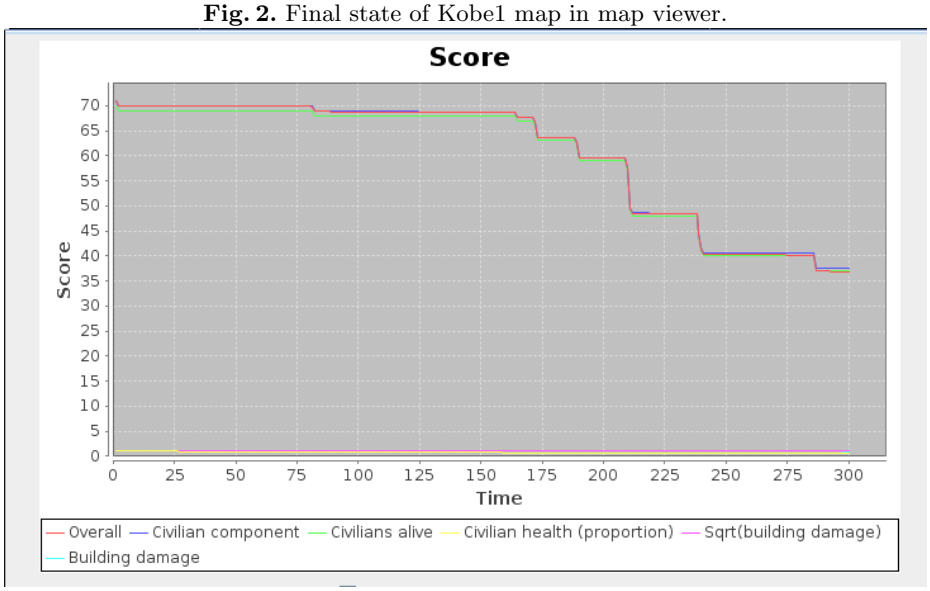

Fig. 3. Final score of Kobe1 map in score chart.

Table 1 shows the results obtained in some of the 2010 competition maps. The second column shows the results we obtained in each map. Third to fourth columns shows the scores achieved by the top three teams in each map including the name of each team. The last column shows the rank expected to be achieved if our team had participated in the 2010 competition in the mentioned maps.

Map	**RMAS ArtSapience**	Top 3			Expected Rank
		1	2	3	
Kobe1	**36.73**	38.862 (IAMRescue)	38.303 (Ri-One)	36.828 (MRL)	4
VC1	**4.681**	4.508 (RoboAKUT)	4.362 (Ri-One)	4.054 (MRL)	1
Berlin1	**61.777**	67.200 (IAMRescue)	58.142 (SBCe Saviour)	55.272 (Ri-One)	2
Kobe2	**10.128**	11.155 (Ri-One)	9.500 (RoboAKUT)	9.284 (ZJUBase)	2
VC2	**18.311**	15.602 (ZJUBase)	10.962 (SEU_Redsun)	10.852 (IAMRescue)	1

Table 1. Results obtained in some 2010 maps.

References

1. RoboCup Rescue Website. http://sourceforge.net/apps/mediawiki/roborescue/
2. Rossi, F., Van Beek, P., Walsh, T., Handbook of Constraint Programming. Elsevier, Foundations of Ariticial Intelligence.(2006), ISSN 1574-6525
3. Apt, K.R., Wallace, M.G., Constraint Logic Programming using ECLiPSe, Cambridge. (2006), ISBN-13 978-0-511-34966-9.
4. Bezdek, J.C., Ehrlich, R., Full, W., FCM: The fuzzy c-means clustering algorithm, Computers and Geosciences, Volume 10, Issues 2-3, 1984, Pages 191-203, ISSN 0098-3004, DOI: 10.1016/0098-3004(84)90020-7.
5. Psaraftis, H. N., Dynamic vehicle routing: Status and prospects, Annals of Operations Research, Springer Netherlands, Volume 61, Issue 1, 1995-12-01, Pages 143-164 ISSN 0254-5330, DOI 10.1007/BF02098286
6. Pitt, J., Mamdani, A., Designing Agent Communication Languages for Multi-agent Systems, Multi-Agent System Engineering, Volume 1647, 1999, Springer Berlin / Heidelberg