RoboCup 2015
Rescue Simulation League Team Description
GUC ArtSapience (Egypt)

Sameh Metias, Mahmoud Walid, Mina Sedra, Ahmed Jihad, Salman ElDash, Maggie Moheb, Mohab Ghanim, Fadwa Sakr, Ahmed Abouraya, Dina Helal and Slim Abdennadher

German University in Cairo, Cairo, Egypt,
[dina.helal, slim.abdennadher]@guc.edu.eg

Abstract

This paper describes the contribution of the GUC_ArtSapience team to the Rescue Agent Simulation in RoboCup 2015 competition. The changes this year include enhancing our civilian death time prediction algorithm which we implemented last year using machine learning. We optimize the Fire Brigades strategy to fill their tanks with water. In addition, task allocation is done more dynamically and efficiently than last year.

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 routing, planning, scheduling tasks, coordination and communication.[3].

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, machine learning, multi-agent systems, and classical AI approaches.

The GUC_ArtSapience team became the champions in the Rescue Agent Simulation in 2013 by ranking first place in our third participation in the competition. Our first participation (as RMAS ArtSapience) was in 2011 where we ranked 3rd place in the final round.

This paper describes the changes in our team's algorithms to produce better results compared to last year. The changes include enhancing the K-means++ clustering algorithm for Fire Brigade Agents and their filling water strategy as well as improving the civilians' death time prediction using supervised learning. We modify the task allocation and prioritization between Police Agents.

The paper is organized as the following, section 2 describes our clustering approach. Section 3 describes the enhancements in our agents' approach this year. Section 4 explains how agents update their world model efficiently. Finally, section 5 concludes our progress so far.

2 Clustering

In the preprocessing phase, we use kmeans++ clustering algorithm to divide the map into regions and assign each agent to a certain region. We continue to use K-means++ clustering algorithm from last year to calculate the initial cluster centroids which are selected from a uniform Gaussian distribution over the buildings/roads in the map. Figure 1 shows the result of K-means++ Clustering using Euclidean distance after 28 iterations

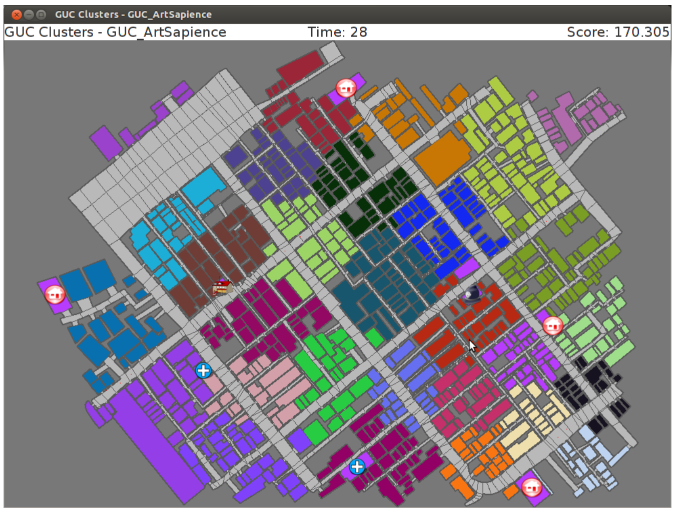
GUC Clusters - GUC_ArtSapience
GUC Clusters - GUC_ArtSapience
Time: 28
Score: 170.305

Figure 1: K-means++ Clustering using Euclidean distance

The clusters are equal to the number of agents in the map, the drawback here is that some roads are assigned to the same cluster even though the path between them is long. This is due to the fact that only Euclidean distance was taken into consideration while computing the clusters.

A more accurate result is acquired by defining members of the same cluster as those who are graph neighbors. Figure 2 shows Kmeans++ clustering using BFS after 14 iterations

In terms of execution time, the first method takes less than 10 seconds for 28 iterations, however the second method it takes about 240 seconds for only 14 iterations. Even though the second approach produces better results, the long execution time is a problem that we are currently working on overcoming.

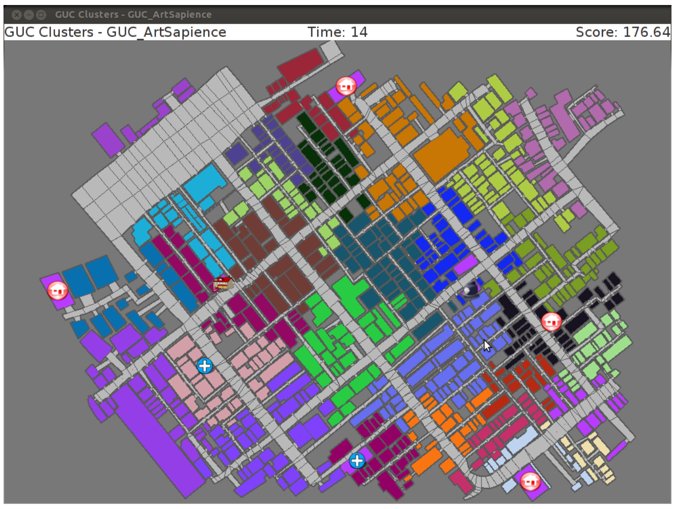
GUC Clusters - GUC_ArtSapience
GUC Clusters - GUC_ArtSapience
Time: 14
Score: 176.64

Figure 2: K-means++ Clustering using BFS

Clustering is used by the Ambulance Team and Police Force agents. Ambulance agents clusters consist of neighboring buildings to allow the agents to cover the map in the least possible time. As for Police agents, the same technique of clustering the map on graph neighbors is used, however clustering is done on roads. The next section explains clustering in Fire Brigades.

2.1 Fire Brigades Clustering

Until last year, Fire Brigades used the same technique as the other agents in terms of dividing the map into clusters and assigning each agent to a different cluster. However, this approach sometimes lead to a scenario where a fire starts in a cluster which is not discovered immediately by the Fire Brigade agent assigned to the same cluster leading the fire spreading uncontrollably in the map. So this year we decided to use MRL's approach in 2014 [2] to handle the re-ignition problem of recently put off buildings by mapping it to the Maximal Covering Location Problem. Using this approach the map is not divided into random clusters, instead using the Fire Brigade agents' line of sight it is going to be divided into a set of points from which all the buildings in the map can be observed. Furthermore these points are going to act as reference points where the agents wait and observe the buildings in the times where they cannot find a specific task to fulfill.

Getting the points from which all the buildings can be observed can be mapped to the unweighed Set Covering Problem [4] . The Problem is that we have a set containing all entities called the universe and a family of smaller sets whose union results in the universe. Our goal is to choose the minimum number of sets from this family of sets that would cover the whole universe. In our case, the universe corresponds to a list of all buildings and the family of sets is a hashmap that relates each and every road to all surrounding buildings that can be seen and extinguished from this specific road.

Since the Set Covering Problem is classified to be NP-Hard, and optimal solution would be reached through a brute force algorithm, which is not feasible since we are limited with pre-computation time. So an approximate sub-optimal solution can be reached through a greedy algorithm. This algorithm chooses at every iteration the road with the maximum unseen, associated buildings to be added to the list of roads. This list of roads is then clustered upon the agents which leads to a proper overview on the whole map.

3 Agents

After clustering, each agent is assigned to a list of tasks within its cluster. In this section we discuss the agents' techniques to prioritize their tasks and make well informed decisions.

3.1 Fire Brigades

3.1.1 Extinguishing Points

Fire Brigade agents prioritize fires by giving the highest priority to warm buildings rather than buildings on fire in order to contain the fires before spreading. However, sometimes agents are not correctly located when extinguishing a fire because they are positioned at one place, stopping the fire form one direction and letting it spread from the other ones. Hence, an enhanced technique is developed to contain fires. A virtual circle is drawn around the center of a fire zone (refer to last year's team description paper in order to know more about the fire zones [1]) and according to the number of agents assigned to the fire zone they are distributed along this circle. In addition, extinguishing points at a hydrant or a refuge are preferred so that the agent does not have to leave its position to fill water.

3.1.2 Filling water strategy

Each Fire Brigade keeps track of all hydrants and refuges in the map. First, the agent searches for all available options to fill water, if the nearest option is a refuge, the agent heads for it right away. Otherwise, the agents looks for the nearest available hydrant. Unlike refuges, hydrants can be used by one agent at a time. Agents going to a hydrant must coordinate together by acquiring the lock to this hydrant. The lock is given to the agent nearest to the hydrant.

3.2 Ambulance Team

One of the main challenges that face the Ambulance team during the rescuing process is prioritizing their targets according to some factors in order to maximize the number of saved civilians as much as possible. One main factor is the estimated death time of a civilian which optimally can be calculated according to some of the parameters provided by the simulator like the damage and health points. Unfortunately, the damage value provided by the simulator is not accurate but rounded to the nearest 10 and the HP (health points) value is rounded to the nearest 1000. Moreover, the growth rate of damage also differs a lot from one civilian to the other even if their original damage values were the same. To deal with these implicit factors which can not be obtained, there must be another way to calculate the estimated death time. The goal is to use learning algorithms [5] to learn the factors that are hard to be calculated using those implicit parameters. Hence having realistic estimations that will lead to better performance of the Ambulance team by prioritizing the agents tasks accordingly.

The proposed approach will be using supervised learning techniques to do so. Supervised learning depends on a training data set to classify the problem in hand into various categories or classes. This is done by running the system multiple times while knowing the final outcome. Obtaining a data set can be easily achieved in our testing environment since we have multiple maps with different settings and scenarios that we can exhaustively run to obtain a respectively huge data set to use later on in the training process. So the work here was divided into two phases, the first one is obtaining a training set that is accurate enough. That was obtained by running different maps and scenarios while disabling all rescuing tactics by agents and only using the agents to retrieve all the data we need for the training. Our training dataset consists of the following parameters: current time step, civilians burridness, civilians HP, civilians damage and the civilians status (Alive or Dead).

All of which is reported by every single agent in the map. Figure 3 plots our training data set during the preporcessing phase that takes place before the learning. The data there is classified into two classes which are Alive (red) or Dead (blue) both against the time in the x-axis. The second phase of the work done is the learning phase. As mentioned above we are using supervised learning to tackle the problem. However, there are many classifiers that could be used, what we are currently doing is that we are trying all possible classifiers and testing them against each other by using each classifier output in a running map and determining which one leads to better result.

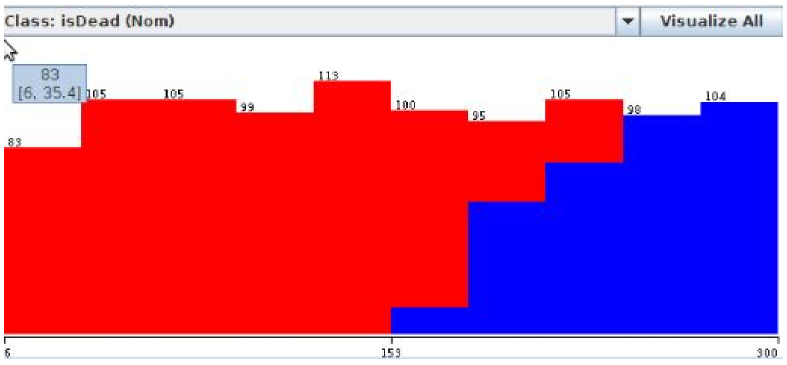

Figure 3: Data preprocessing

3.3 Police Agents

The Police agents are distributed across the map, whereby, each agent is assigned to a certain cluster consisting of a set of roads. Sometimes the police agents spawn close to each other, consequently the same set of tasks are given to several agents, which is considered a waste of agent resources. Given the fact that the number of agents clearing the same blockade does not change the clearing rate, we use avoid assigning the same blockade to multiple agents using communication between the agents.

3.3.1 Dynamic Task Allocation

We set the maximum number of agents to be on the same road to 2 agents, if a third agent joins, we calculate and assign another BFS tree with the condition of excluding this certain road from his path. This is implemented by the use of semaphores which limits the number of agents that are simultaneously accessing the same resource, in our case it's the roads, this problem could be visualized better in figure 4.

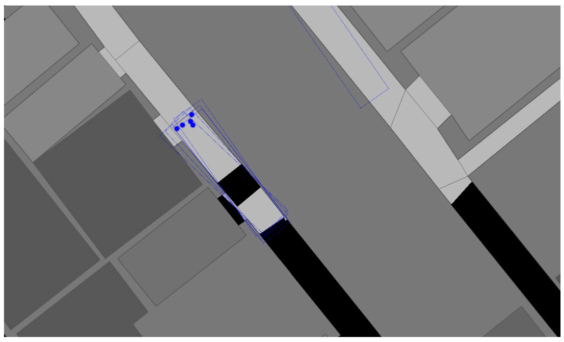

Figure 4: Five agents are on the same road

Another scenario is multiple Police agents have the same path to the destination, we limit this number of agents to 2 also, otherwise the agent gets assigned to another task or keeps roaming in his cluster. For such coordination to occur, we use the line of sight as well as voice communication, where agents nearby tell each other their destination and their path. Police agents now ignore and do not clear building entrances if the buildings were not reported before or the agents line of sight does not see any buried agents or civilians inside, hence focusing on the path to their destination.

4 World modeling

As mentioned in the previous sections, each agent has a set of tasks they are assigned to. So, we needed to define and set up the world for each agent according to his tasks. In this section we explain how each agent perceives the world model.

4.1 Communication Message Handler

Each agent has initially a set of all entities that needs to be updated individually. An agent also has its own message handler where each handler handles both voice messages heard by the agent or radio messages received through the channel(s) the agent is subscribed to. It handles both incoming and outgoing messages by analyzing the messages, compressing and/or decompressing them and handling dropped messages (refer to GUC_Artsapience 2014 TDP [1] for detailed description of our communication model). Using this handler, the agent is able to communicate with other agents in order to propagate knowledge.

The dependence on the Communication Message Handler to update the agents world is somehow risky because of the probability that the messages will be dropped or because of the scenarios with no communication. Hence, we handle this case by adding an expiring threshold for each value that needs to be updated. If this threshold is exceeded, the agent updates his world with a default value. For example, Fire brigades discard every blocked road from their graphs when calculating a path to a certain fire. However, if the agent is not notified within 20 time steps that the road is cleared, it automatically assumes that it is cleared and updates its world model accordingly. This technique works well taking into consideration that the agent probably would have moved away from that blocked road in those 20 time steps and does not need it anymore.

4.2 ChangeSet Analyzer

An agents decision is based on various factors and changes that occur to his world model, these changes have two sources as mentioned before, communication channels and radio messages. In addition, every agent has a change set each time step which reflects what falls in the agents line of sight. This is where the change set analyzer comes handy, it takes the change set and updates the agents world model regarding buildings and roads status which also gets propagated to other agents using the communication message handler.

5 Conclusion

In conclusion, this paper explains the updates in our approach since last year. We improve the Ambulance team's rescuing efficiency by estimating the expected civilians death time using supervised learning, hence correctly prioritizing the Ambulance agents' tasks. We modified the Firebrigade Agents clustering technique using the Set Covering Problem to find the most optimal points for the Fire Brigade Agents to be positioned in order to have a full view of the map and be aware of any fire as soon as it starts. We optimize the Police Agents' performance by avoiding duplicate tasks being assigned to the same agents using locks on the tasks that are shared between the agent using our communication model.

References

[1] Guc artsapience team description paper 2014.

[2] Mrl team description paper 2014.

[3] Robocup rescue website. http://sourceforge.net/apps/mediawiki/roborescue/.

[4] Asaf Levin. Approximating the unweighted k-set cover problem: greedy meets local search. In *In Proceedings of the 4th International Workshop on Approximation and Online Algorithms (WAOA 06), LNCS 4368*, pages 290–310. Springer, 2006.

[5] Lichen Liang and V. Cherkassky. Connection between svm+ and multi-task learning. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2048–2054, June 2008.