Epicenter Team Description Paper

Daniel Epstein and Ana L. C. Bazzan
Instituto de Informática - Universidade Federal do Rio Grande do Sul
Caixa Postal 15.064, CEP 91501-970, Porto Alegre, RS, Brasil
{depstein;bazzan}@inf.ufrgs.br

Abstract— **In this paper we describe the functionality of the Robocup Rescue agents developed by Epicenter Team, at Universidade Federal do Rio Grande do Sul. We will describe which strategy each of the three type of agents will use and will focus on the way those agents plan their path along the maps. Furthermore, we will describe how we use the center stations to distribute knowledge among agents and how each type of agent decides which task to do first.**

I. INTRODUCTION

In this short-paper we propose a decentralized approach where the agents use the communication to have a better view of the environment. A path planning approach based on wave propagation over a graph structure that represents the environment topology is also applied. This graph has edges that represent the avenues and nodes that are associated to intersection of avenues and hotspots.

This paper is organized as follows: Section II describes communication among agents and centrals; Section III describes the proposed method of path planning; Section IV describes the method by which agents select their tasks; and in Section V we present some conclusions and possible working directions.

II. COMMUNICATION

Given the need of several message types between the agents, we have defined a communication protocol, seen in Table I. Figure 1 shows the agents as rectangles and the messages are represented as arrows. Each message type indicates the message contents. Due the limitation in the radio communication of the simulator, this protocol was implemented using a byte code in order to reduce the size of the messages (limited to 256 bytes).

Messages of type `fire` and `clear` are exchanged between the Fire Brigade (FB) and the Fire Station (FS) and are used to send lists of buildings on fire and cleared, respectively. The Fire Brigade (FB) and the Fire Station (FS) agents use the messages of type `buried` and `saved` to share the information about buried and saved civilians, respectively. When the FB and Ambulance Team (AT) are blocked they send the `blocked` message to their respective stations. Using the information received in the `blocked` messages, the messages of type `blockedList` are built up. This kind of message is only used by the stations, where the Fire Station and Ambulance Center send the list of blocked agents to the Police Office (PO), and the PO sends to the Police Force (PF) agents.

All those messages are important to expand the agents perception and to remove the finished tasks from the agent's local list of task. We do not use the direct communication between agents since they change their positions very fast and this communication is not as reliable as the radio one.

III. PATH PLANNING

Before presenting the path planner, we will introduce some basic concepts to make the explanation clearer. Consider a graph $G = (\mathbf{V}, \mathbf{E})$ comprised of a set of vertex V and a set of non-oriented edges E. This graph is connected if there is a path between any two nodes, i. e., each node is reached by any other in the graph. As the agent move around the city, we extract the graph structure from the agent perception.

We consider the city streets as edges and the intersection of the streets as vertex. The agents use the i-neighborhood of vertex to determine the path to reach a specific position from its current position. We define an i-neighborhood $\mathcal{N}_i(v)$ as the set of vertex that are reached by a path with length i from v, i. e:

$$\mathcal{N}_i(v) = \{u \in \mathbf{V} \mid d(u, v) = i\}$$

where $d(u, v)$ corresponds to the length of shortest path between u and v in terms of number of edges, with $d(u, v) = d(v, u)$.

Initially, the agent determines its position in the graph and after it propagates a specific value to the other vertex in its i-neighborhood. For instance, if the agent is at vertex v, then each vertex $u \in \mathcal{N}_m(v)$, for $m = 1, 2, \ldots$, stores a propagated value $p_v(u) = m$. In this case, the value $p_v(u)$ indicates the number of edges that the agent should access from vertex v to reach the vertex u, i.e., $p_v(u) = d(u, v)$.

Using this method, the recovery of a path is straightforward. The agent determines the vertex associated to its current position v and the goal position, g. The path $\mathbf{P} \subseteq \mathbf{V}$ is built from the vertex g. This path corresponds to a sequence of vertices

$$\mathbf{P} = \{u_0, u_1, \ldots, u_{d(v,g)}\}$$

where $u_0 = v$, $u_{d(v,g)} = g$ and $|\mathbf{P}| = d(v, g) + 1$. Each vertex u in this path is determined using its propagated value, $p_v(u)$, from vertex v.

This computation is straightforward, but it may access unnecessary vertices. For instance, consider a simple case, where the graph is Δ-regular (all vertex have the same degree Δ). If the goal node is at distance k from the agent

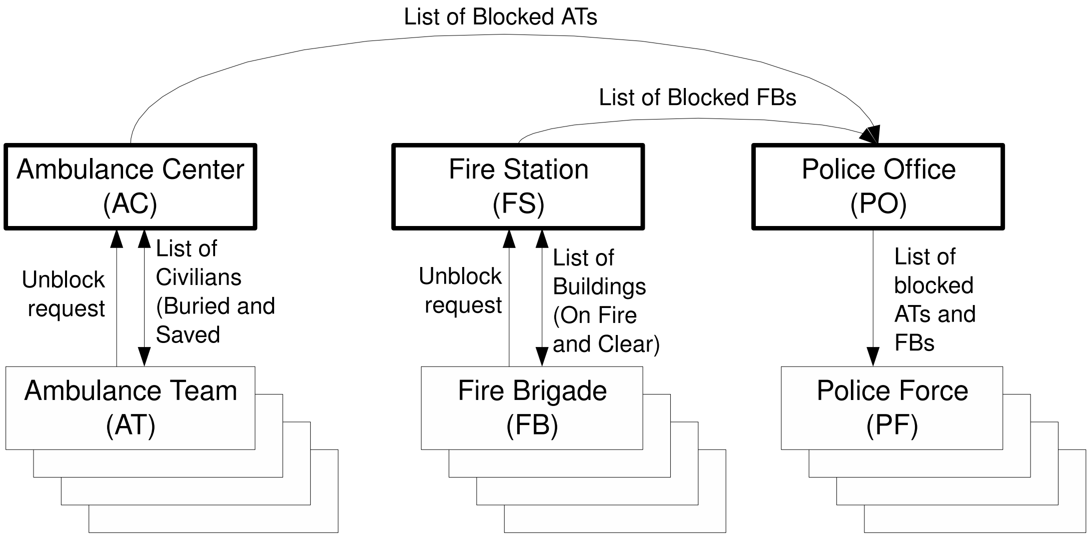

Fig. 1. Communication between agents and centrals.

TABLE I
MESSAGES EXCHANGED AMONG THE AGENTS.

Sender	Receiver	Message Type	Contents
FB	FS	`fire`	List with the IDs from the buildings on fire.
FS	FB		
FB	FS	`clear`	List with the IDs from the cleared buildings.
FS	FB		
FB	FS	`blocked`	Agent's own location.
AT	AC		
AT	AC	`buried`	List with the IDs of the buried civilians.
AC	AT		
AT	AC	`saved`	List with the IDs of the saved civilians.
AC	AT		
FS	PO	`blockedList`	List with the location of the blocked FBs.
AC	PO	`blockedList`	List with the location of the blocked ACs.
PO	PF	`blockedList`	List with the location of the blocked ACs and FBs.

position, then the method will access successively a number of nodes of order O(Δ^{k+1}). Furthermore, if an unexpected event happens during the path execution, the agent must re-plan the path towards its goal position, without considering the path already planned.

To solve this problem, we can divide the planning path in two parts. Instead of planning beginning from the agent position, it begins from both the agent and the goal positions. We determine the vertex v and g that are associated to the current agent position and goal position, respectively. After, we compute successively

$$\mathcal{N}_1(v), \mathcal{N}_1(g), \mathcal{N}_2(v), \mathcal{N}_2(g), \ldots, \mathcal{N}_m(v), \mathcal{N}_m(g)$$

until

$$\big(\mathcal{N}_m(v) \cap \mathcal{N}_m(g)\big) \bigcup \big(\mathcal{N}_m(v) \cap \mathcal{N}_{m-1}(g)\big) \neq \emptyset$$

The vertex set $I = \{u \mid u \in \big(\mathcal{N}_m(v) \cap \mathcal{N}_m(g)\big) \bigcup \big(\mathcal{N}_m(v) \cap \mathcal{N}_{m-1}(g)\big)\}$ contains meeting vertex far at most $\lfloor d(v,g)/2 \rfloor + 1$ from both v and g. These vertices are used to recovery the path from v to g using a similar procedure as shown previously. The cardinality of set I indicates the number of candidate paths from v to g.

To compute a path, initially, we choose a vertex $x \in I$ and compute the path $\mathbf{P_1} = \{u_0, u_1, \ldots, u_{d(v,x)}\}$ where $u_0 = v$, $u_{d(v,x)} = x$. After, we compute the path $\mathbf{P_2} = \{w_0, w_1, \ldots, w_{d(g,x)}\}$ where $w_0 = g$, $w_{d(g,x)} = x$. So, we have a path from current agent position v to vertex x and from goal position g to vertex x, with $|\mathbf{P_1}| - |\mathbf{P_2}| \leq 1$.

We join these paths into a unique path $\mathcal{P} = \{p_0, p_1, \ldots, p_{d(v,g)}\}$. The vertex from p_0 to $p_{d(v,x)}$ corresponds to the path from agent position to meeting vertex. That is, $p_i = u_i$, for $i = 0, 1, \ldots, d(v,x)$. Whereas the vertices from $p_{d(v,x)+1}$ to $p_{d(v,g)}$ are associated to the vertex of the path $\mathbf{P_2}$ in inverse order, i. e., $p_{d(v,x)+i} = w_{d(g,x)-i}$, for $i = 1, 2, \ldots, d(g,x)$.

The main advantage of this approach is that the number of accessed vertices is less than the previous approach. For instance, in a Δ-regular graph, as considered previously, the method will access a number of vertices of order O($\Delta^{\lfloor k/2 \rfloor + 1}$). Furthermore, it is possible to handle dynamic events in a more efficient way. For instance, consider that the agent is following the path and at vertex $q \in \mathcal{P}$, it finds a blockage. If $q \in P_1$, the agent needs to re-plan a path from its position q accessing its neighboring

$$\mathcal{N}_1(q), \mathcal{N}_2(q), \ldots, \mathcal{N}_t(q)$$

until

$$\mathcal{N}_t(q) \cap \mathcal{N}_{\lfloor d(v,g)/2 \rfloor}(g) \neq \emptyset$$

In this case, new meeting vertices are determined and the process is repeated. Observe that the path P_2 is not recomputed and, therefore, planning time is saved.

IV. TASK ALLOCATION

We will describe shortly the method implemented by each agent to select which task to execute first. The method used is detailed in [1] and we will the same algorithm here. In [2] there is a comparison between differents task allocations strategies.

First, we must consider that each agent have a limited ability to perform a task. Agents on different area of the map have different potential to perform a given task for many reasons. One could think that their ability would be proportional to the distance between the agent and the task. But there are also other factors that must be considerered. We will now explain what factors were considered for each type of agent in order to allocate a task to them.

A. Fire Brigade (FB)

In order for a fire brigade to perform properly a given task, he cannot be very far from it, otherwise the building would collapse before he could reach it. Also, the building he would attack must be important for the final score. So we consider that the competence of a fire brigade agent is proportional to his distance to the building on fire and to the size of this building. The larger building will have priority over the smaller one and the closest one will have priority over the rest. By doing this selection, we try to reach the biggest amount of building and focus on the most important ones.

B. Police Force (PF)

The PF don't have any direct influence on the final score, once the number of clear roads is not taken in consideration. However, it must perform his tasks properly so that the other agents can move freely through the map. We consider that the most important property to consider are the distance from a PF to the road he intend to unblock and the size of this blockade. We focus on unblocking as many roads as we can, instead of unblocking one big road.The reason is that we intend to ensure that a path between two points will exist, even if that path is not a direct line.

C. Ambulance Team (AT)

Ambulances have the task to save lives. Every moment an AT is moving, it is time wasted that it could be saving lives. So it is very important to consider the position of an AT before choosing what task to perform first. The other factor we choose to consider is the HP of the civilian. We choose to save as many lives as possible, even if all the civilian are badly injured.

V. CONCLUSIONS AND FUTURE WORKS

This TDP has described the strategy used by EPICENTER team at the Latin American Robotics Competition 2010. Although the agents performance was satisfactory, its important to notice that there are a few issues to be reviewed. For example, there is no communication between fire brigades and ambulance teams. This communication could be used so civilians who are near a burning building could be saved first or maybe direct fire brigades to an area with many civilians buried.

The path planning proposed could also be improved by combining the knowledge multiple agents (i.e. when two agents are going to the same part of the scenario it is possible to compute the path only once and inform it to the other agent).

Finally, using a different task allocation method could be more effective in some cases. A mix of techniques or even a mix of thresholds could be used to improve the final score.

REFERENCES

[1] P. R. Jr. Ferreira and F. Boffo and A. L. C. Bazzan. *Using Swarm-GAP for Distributed Task Allocation in Complex Scenarios.* Massively Multiagent Systems. Springer, 2008. n.5043, p.107121. (Lecture Notes in Artificial Intelligence).

[2] P. R. Jr. Ferreira and F. Santos and A. L. C. Bazzan and D. Epstein and S. J. Waskow. *Robocup Rescue as Multiagent Task Allocation among Teams: experiments with task interdependencies.* aama. vol.20. 3. May, 2010. 421-443.