RoboCup Rescue 2011 - Rescue Simulation League Team Description Epicenter (UFRGS, Brazil)

Daniel Epstein and Ana L. Bazzan

Instituto de Informática
Universidade de Rio Grande do Sul, 15064-Porto Alegre, RS, Brazil
{depstein,bazzan}@inf.ufrgs.br

Abstract. This paper describes the main features of the Epicenter simulation team. We describe the strategies used for each type of agent, as well as how each task type was handled. Our main approaches are based on i) use of wave propagation to find the shortest path for each agent when moving towards its goal, and ii) team formation based on coalitions. Since it is well-known that computing the optimal coalition structure is unfeasible given the time constraints, we use an heuristic approach that allows us to prune certain coalitions. Moreover, this process is done in a decentralized way, which is appropriated for scenarios where communication is not possible.

1 Introduction

The Robocup Rescue League is more import today than ever. Natural disasters such as the one that has just happened in the northeast of Japan shows the importance of having robot teams to perform rescue tasks or tasks that are dangerous to humans (such as work on damaged nuclear plants).

Since this is our first competition in the Robocup Rescue League, in the present paper we provide details about the approaches that are used in order to explain how our agents are built, as well as the main strategy underlying them. Regarding the former, the main approaches that are used here are wave propagation (as in [3]), and coalition formation for task allocation (used by us in [1]). Wave propagation aims at representing the environment topology and finding paths from agents' current locations to a goal location. Coalition formation is used for team building. Since it is well-known that computing the optimal coalition structure is unfeasible given the time constraints, we use an heuristic approach that allows us to prune certain coalitions. We calculate the value of a task (explained on section 3) and ignore those with low value. Also, several restrictions are imposed for an agent to participate in a coalition. Once the coalition structure is formed, we assign each agent to it designated task (more details on section 5). Moreover, this process is done in a decentralized way, which is appropriated for scenarios where communication is not possible.

The rest of this paper is organized as follow: Sections 2 and 3 describe our path planner using wave propagation and how to assign value to the tasks, respectively. In Section 4 we describe how agents act in this environment using the approaches previously presented. Section 5 discusses the issue of coalition formation and selection, while Section 6 summarizes our conclusions.

2 Path Planner - Dual Wavefront Propagation

The path planner used here is similar to the one used in [3]. Before presenting the path planner, we introduce the terminology used. Consider a connected graph $G = (\mathbf{V}, \mathbf{E})$ comprised of a set of vertices V and a set of non-oriented edges E. G is extracted from the given city map.

Agents use the i-neighborhood of a 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 vertices that are reached by any path with length i from v as in Eq. 1 where $d(u, v)$ corresponds to the length of shortest path between u and v, with $d(u, v) = d(v, u)$.

$$\mathcal{N}_i(v) = \{u \in \mathbf{V}\ |d(u, v) = i\} \tag{1}$$

Initially, the agent determines its position in the graph; then it propagates a specific value to the other vertices 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 distance to vertex u from vertex v, i.e., $p_v(u) = d(u, v)$, assuming that all vertices have the same distance between them. When this is not the case, the new propagation start by the vertex with lowest $d(u, v)$.

Using this method, the computation 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. For instance, if all vertices have equal distance, then
$u_i = \arg\min_{(a,u_{i+1})\in\mathbf{E}} p_v(a)$ with $i = 1, 2, \ldots, d(v, g) - 1$.

If an unexpected event happens while an agent is moving, it must re-plan (while also not taking the previous path into account). To solve this, we divide the planning path in two parts. Instead of planning only 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 repeatedly compute $\mathcal{N}_1(v), \mathcal{N}_1(g), \mathcal{N}_2(v), \mathcal{N}_2(g), \ldots, \mathcal{N}_m(v), \mathcal{N}_m(g)$ 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 cardinality of set $I = \{u \mid u \in \big(\mathcal{N}_m(v) \cap \mathcal{N}_m(g)\big) \bigcup \big(\mathcal{N}_m(v) \cap \mathcal{N}_{m-1}(g)\big)\}$ 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 from v to x. After, we compute a path from g to x.

We merge these paths into a unique path $\mathcal{P} = \{p_0, p_1, \ldots, p_{d(v,g)}\}$. The set of vertices from p_0 to $p_{d(v,x)}$ correspond to the path from the agent position to common vertex x. 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 with the vertex of the path from g to x in inverse order. This method handles dynamic events in a more efficient way than the simple wave propagation. For instance, consider that an agent is following the path and finds a blockage near vertex v. In this case, the agent must re-plan only the path from v to x. The path from x to g is not recomputed and, therefore, planning time is saved.

In order to improve the movement of a fire brigade around the scenario, we also consider the path from each possible task to all others. All tasks that are close to the first task that the agent has to execute will also be part of this process and the resulting path will lead the agent to the first task in a path that makes it easier to move to the subsequent task. Hence, once the first task is solved, it will be easier for the agent to move to the next one and so on.

3 Task Value

Each task in the Roboup Rescue have a different value (contribution) for the final score. Also, it affects the system in a different way. Choosing to perform task a or task b could entirely change the outcome of the simulation. Also the order with which tasks are performed is key. Therefore, is very important to be able to choose the correct task to perform first. We have developed several metrics that indicate how important one task is. Comparing the values of different tasks, agents can choose the one that is the most important.

3.1 Task: Buildings

Choosing which fire to extinguish first is a very tricky question. There are several factors that must be taken into account. Sometimes, its better to lose one single building than to lose a whole block. That's why our main idea is to prevent the fire from spreading instead of trying to save every building. Our metric for computing the value of a building (V_b) is based on the follow variables:

- size of the building: represents the total area of the building. A larger building has a more direct impact on the final score, but its is also harder to extinguish. In order to prevent the fire from spreading, the agent must give priority to the smallest one.
- degree of destruction: a building that is close to full destruction receives less priority.
- neighboring buildings: a single fire may spread to an entire block. Hence, we must consider the entire block when choosing a goal. We consider the total area of each neighbor building, as well as how destructed it is. Neighboring buildings that are on fire are not considered, since a building cannot spread fire to another building already on fire.

3.2 Task: Roads

Roads are difficult to evaluate since they do not have a direct effect on the final score. Metrics related to roads must be based on how a blockage will affect the movement of the agents. If a crucial portion of the network is unreachable because it is blocked, then links in this portion must have priority over other roads. Also, main roads are much more used and must be unblocked first to ensure that agents may move freely through the map. Another important information related to roads is how many lanes they have. Usually arterial roads in a city are those with a high number of lanes. Therefore, it is assumed that roads with the highest number of lanes are the most important ones. It is necessary to unblock at least parts of these roads first, even if they do not end up being completely unblocked.

Finally, the most important information regarding roads is how close they are from a refugee or a fire station. It is crucial to clean those roads that lie near important buildings, since these areas have a high traffic of agents during the whole simulation.

3.3 Task: Civilians

Although a civilian has many attributes, it is not difficult to formulate a metric for it when regarded as a task. A major issue is to minimize the number of fatal victims by the end of the simulation. To do so, one must consider the civilian hit points that indicate how long it can survive. Another issue that contributes to the value of a civilian task is how buried this civilian is, which indicates the number of cycles one ambulance team will take to save this civilian. These two attributes can be used to formulate what will be called "expectation of life" (V_c), that is how many cycles a civilian can remain buried before dying.

To find a civilian, one may see or listen to it. If some agent has seen or heard a civilian, that agent records the location and time step the observation was made. If this agent is not an ambulance itself, once it finds an ambulance center, it will inform this center about the location of known civilians, which then passes the information to an ambulance, helping it to find an injured civilian.

4 Agents

Platoon agents use the value of each task to determinate which one to do first. Each agent chooses a task for itself or adopts the one indicated by a central (if there is any). In the first cycles, the civilian makes this decision alone, without considering other agents. After, the central computes which group this agent belongs. This is the case when communication is used/allowed. Otherwise the agent may make this decision by himself.

All agents try to gather as much information as possible. Each time an agent meets another one, information is shared, together with the time step when this information was obtained. To avoid having too much communication, only the newest information is shared. Information that is older than 10 time steps is not transmitted.

The method used for communication is the use of `say`. Each time two agents meet, if the cycle is an odd number, the agent with highest ID will be the one transmitting the information. Otherwise the agent with lower ID will be the sender. This way, if two agents meet for more than one cycle, both can send information. If they meet for only one cycle, then the flow of information follows a random pattern.

The information shared is coded by the Huffman code. If it is not possible to send the whole message an agent wants, it will first send the information that directly affects its type (i.e. buildings on fire for FB, blocked roads for PF, or civilians injured for AT).

4.1 Agent: Fire Brigade

The first cycles of the fire brigade (FB) are used to extinguish as much fire as possible. It is very important that a FB moves quickly in the beginning of the simulation so the fire does not spread. After 20 cycles, this agent tries to reach a fire station and determines whether it is blocked or not. If it is blocked or cannot reach a fire station, it contacts the central or any other agent to request help from a police force.

Choosing which building task to handle first depends on V_b (as defined before), as well as the distance from the FB to the building and the agent ability to extinguish a fire. If a FB cannot extinguish a certain fire, it tries to extinguish one in a smaller building.

4.2 Agent: Police Force

At first, all police forces (PF) have one task: to move to the nearest refuge and fire station they can find, and unblock the path to them. This task will take as many cycles as necessary.

After this initial task is completed, the PF priority is to unblock any agent trapped. This can be done by receiving messages that have the location of an blocked agent, or when it realizes that some agent is not moving. Once a request from a trapped agent is received, the two closest PF move towards that agent in order to quickly release it. They will work together, thus reducing the time it takes to unblock a road.

Finally, once these two kinds of tasks are handled, each police force tries to unblock other roads. As mentioned, they do not need to completely unblock a road. Using the previously defined metrics, police forces decide which are the priority tasks. The result of the metric is combined with the following heuristic: All roads that are close to a fire spot have higher priority. This way, roads closest to the highest number of fire spots will be unblocked first. This provides fire brigades a higher number of possible paths.

4.3 Agent: Ambulance Team

Ambulance teams (AT) must balance their resources so that the highest possible number of civilians is saved before it is too late. The ambulance team uses the metric V_c combined with the number of cycles it takes to reach each civilian to decide who to save first. Ambulances prioritize those civilian that can be saved within the available time, and that have a lower V_c. Also, civilian who are close to another civilian also have priority. If there is a group of civilian close to an AT, they probably have a much higher priority than those civilians that are isolated or very far.

Not all civilian saved will be immediately taken to the refuges. An AT will perform rescue tasks first. After, when there is none left to be rescue, civilians will be delivered to refuges. Hereby, those severely injured have priority.

Another way for an AT to choose its task is by receiving a message from the centrals. These messages have a higher priority and the task received must be executed. This is so because sometimes a civilian cannot be saved by only one ambulance and then the central has to send more than one to do this rescue.

4.4 Agent: Centrals

The main objective of a central is to gather information from all agents and coordinate the flow of information and their actions. It also works as a task allocator once it has information about all agents and can provide a better overview of the whole scenario. It can also indicate the existence of some high priority task and request help from multiple agents.

Once communication by radio affects the score, not every single new data will be sent. Rather, this kind of communication is used only a given quantity of data is gathered. It is also used to decide which coalition structure is formed, as described next.

5 Coalition Structures

As mentioned, the generation of all possible coalition structures (CS) is exponential in the number of agents. This is an important issue because it was demonstrated that finding the optimal coalition structure is NP-complete [5].

In the Robocup Rescue the number of coalition structures is affected not only by the number of agents but also by the number of tasks. Hence, low-priority tasks are not considered in the set of coalition structures. From the point of view of an agent, some tasks may also be discarded because the agent either has no resources to perform it, or tasks are located far away from it. This way agents that cannot be allocated to important tasks are removed from the set of agents to form coalitions, further reducing the number of possible coalition structures. After we find a reduced number of agents and tasks, the search of the actual optimal coalition structure is performed by the anytime algorithm proposed in [4]. We also remark that coalition structures are generated only considering ambulance teams and fire brigades. This stems from the fact that police forces do not need to work in groups (they can perform their tasks alone and are not hardly constrained by time).

To ensure that the corresponding central of agents will be able to generate a good coalition structure, it must first discard most of those possible CS's that are not valid. The central cannot find an efficient CS when there are too many agents because it has a very limited time to search for it.

The main principles underlying our method (reducing the number of possible CS) are based on the idea that tasks having low value, plus agents that cannot complete a task, should not be considered in the search for a CS. The number of tasks is at most the number of possible agents. If there are more tasks than this number, those tasks with with the lowest values are ignored. To prune the number of agents we disregard those that are many cycles away from the most valuable tasks and those that cannot deal with the given tasks (for instance, a fire brigade that does not have enough water on its tank). Further details can be found in [1]. Due to the dynamic nature of the task allocation process, the set of CS's changes every 10 cycles.

It is important to remark that there are many scenarios where the communication between the central and the agents is not possible. For these cases we have developed a strategy to form groups of agents that is communication-free. It is a two-cycle strategy. In the first one, agents contact any other agent in their range (if any) to update its knowledge base (i.e. agents memory). In the second cycle, agents rely on their knowledge bases to assume where the other agents are, and what they are doing.

Each agent keeps track of where others are. Using this information, they assume an agent's location and which tasks have already been completed. With a given probability ρ, an agent assumes what are the other agent's priority tasks. The more recent the information about a certain region, the most reliable is this information.

Once an agent has a hypothesis about where other agents are, it uses the same algorithm that would be used by the central for generating coalition structures, and decides which is the most probable CS to be formed.

When no assumption of this kind is made, this means the agent has not enough information about the others and ignore them when trying to find a coalition structure. This drastically reduces the number of possible CS's.

After deciding which task to perform, each agent moves towards its goal. If it realizes that no other agent from the assumed coalition is also committed with its task, it tries to change task deciding which is the best task to perform in an isolated way. It also removes from its knowledge base those information that led it to make the previous assumption (for example, a building that was thought to be on fire but is not). In case the agent manages to contact other agent from its assumed coalition, these agents exchange information and try to act together for the next coalitions by increasing the value of those CS where they are together, until one of them has to leave (to transport a civilian to a refugee or refuel the tank of water).

6 Results and Concluding Remarks

In [1] we presented results for a similar strategy (henceforth DAS), based on coalition structures. The following results refer to the Kobe map, using version 0.49 of the simulator. For comparison, three other algorithms were used: $LA-DCOP$ ([6]), $Swarm-Gap$ ([2]) and a greedy agent. Table 1 shows the results.

Table 1. Scores for version 0.49

DAS	LA-DCOP	Swarm-Gap	Greedy
70.28 ± 6.94	49.69 ± 6.31	44.97 ± 1.76	43.78 ± 7.19

Since that publication we have improved our agents in many aspects and have also added new strategies to make them even more competitive. It is important to note that in [1] our focus was only on the coalition structure and we did not used other strategies described here (e.g. the decentralize coalition assumption).

References

1. Daniel Epstein and Ana L. C. Bazzan. Dealing with coalition formation in the RoboCup Rescue: an heuristic approach. In *Proc. of the 3rd International Conference on Agents and Artificial Intelligence*, volume 2, pages 717–720, Roma, Jan. 2011.
2. Paulo R. Ferreira, Jr., Fernando dos Santos, Ana L. C. Bazzan, Daniel Epstein, and Samuel J. Waskow. Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies. *Journal of Autonomous Agents and Multiagent Systems*, 20(3):421–443, May 2010.
3. Lynne E. Parker, Ben Birch, and Chris Reardon. Indoor target intercept using an acoustic sensor network and dual wavefront path planning. In *In Proceedings of IEEE International Symposium on Intelligent Robots and Systems (IROS 03*, pages 278–283, 2003.
4. Talal Rahwan, Sarvapali D. Ramchurn, Viet Dung Dang, and Nicholas R. Jennings. Near-optimal anytime coalition structure generation. In *Proc. of the Int. Joint Conf. on Art. Intelligence (IJCAI 07)*, pages 2365–2371, January 2007. available at http://ijcai.org/proceedings07.php.
5. Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1–2):209–238, 1999.
6. Paul Scerri, Alessandro Farinelli, Steven Okamoto, and Milind Tambe. Allocating tasks in extreme teams. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 727–734, New York, USA, 2005. ACM Press.