Y-Rescue Team Description Paper

Anderson Tavares*, David Saldaña, Hayllander Blonski, Hector Azpúrua,
Jhielson Pimentel, Omar Pino, Rafael Colares,
Douglas Macharet, and Luiz Chaimowicz

VeRLab – Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

Abstract. This paper describes the strategy of Y-Rescue team to the RoboCup Rescue 2015 Agent & Multi-agent challenge competition. For the agent competition, our approach relies on field agents capable of individual decision making, relegating the center agents to an informational role. Our main contributions are the scheme for prediction of fire evolution done by fire brigades, the behaviour-based strategy for police forces and the simple recruiting mechanism in tasks that demand more than one agent. For the multi-agent competition, we choose to implement eXtreme Ants, a scalable, swarm intelligence based algorithm for task allocation in dynamic environments.

1 Introduction

Rescue operations in disaster situations, such as earthquakes, tsunamis, avalanches, for example, are a serious social issue. These operations involve different agents (police forces, firefighters, paramedics, among others) in a hostile environment: buildings collapse, civilians get wounded, streets get blocked, water and power supplies get compromised and communication is limited, making information about the problem scarce and imprecise.

Such situations demand systems that can create robust, dynamic and intelligent search and rescue plans to aid human effort. Within this context, the goal in RoboCup Rescue Agent & Multi-agent challenge competition is to foster research and development in the field of coordination in multiagent systems. In 2015, the challenge is to program teams of fire brigades, paramedic and police forces in the Agent RoboCup Rescue platform [2] and a decentralized constraint optimization problem (DCOP) algorithm in RMASBench [3].

This paper describes the strategies that will be used to guide Y-Rescue team in Robocup Rescue Agent & Multi-agent challenge competition. It is divided in two main parts. In the first part (Section 2), we describe our main goals in the Agent competition, which are: to develop a model to predict how the fire evolves in burning buildings, to define a behaviour-based strategy for the police force agents and to use a simple method for recruiting in tasks that demand more than one agent. In the second part (Section 3), we describe our approach

* Corresponding author: anderson@dcc.ufmg.br

to the multi-agent competition, which involves the implementation of eXtreme-Ants [4], a scalable algorithm based on swarm intelligence for task allocation in dynamic environments.

2 Agent challenge competition

2.1 Ambulance Team

This team is responsible by the most important task in a disaster environment, namely the search and rescue of injured civilians. Seeking for this goal, the ambulance team will be aided by the other agents with respect to the discovery of new victims.

The team of paramedics will be divided into sectors, defined by the number of available agents and map size. These regions will have a category of importance measured by the quantity of civilians not rescued yet, ordering the contingent of agents to be designated to most important areas. Separating the agents into regions of interest will make them cover all the map and save, in thesis, the majority of civilians.

Task Allocation The task prioritization of this team will be governed by the life estimation of the alive civilians. This estimation takes four factors into account: agent-civilian distance; civilian's life; civilian's damage and whether or not the civilian is buried in a burning building.

2.2 Police Force

This team has two main goals: to clear obstructed paths and to scout unvisited areas of the map. The first goal can be divided into three jobs with different priorities, in order: i) clear obstructed paths of important buildings, ii) clear paths that are obstructing other agents, iii) clear random obstructed paths. The second goal is to continuously search the map for survivors and fire spots.

To achieve these goals, police force agents use a behaviour-based controller. Each agent chooses a behaviour based on its position and the location of important features on the map. The behaviours are:

- *Sweeper*: the *sweeper* behaviour will try to clear important paths. These paths includes the vicinity of important map features;
- *By-demand*: the *by-demand* behaviour will clear their own vicinity and will constantly listen to the communication channels for help requests;
- *Scout*: the *scout* behaviour will keep searching inside buildings for survivors.

Task Allocation Initially, the map will be divided into n clusters, where n is the number of police force team agents. These clusters will be created taking into account the size of the map and the localization of each agent. Agents who are near an important feature of the map will be given the *sweeper* behaviour, while

agents who are on other areas will be given the *by-demand* behaviour. When an agent finishes its tasks, it will then be assigned to the *scout* behaviour.

The *sweeper* agent task is simple and direct: it will clear its cluster's paths until all paths are clear. It will not listen to any communication channel or stop its task for any reason. The *by-demand* agent task is more complicated, as it will be listening to a communication channel and responding to calls for help. An agent from any other team will be able to broadcast a distress message to the police force team informing that it is trapped and need assistance. This message contains the sender's current task and location. When a *by-demand* agent receives a help message and it is the closest to the sender agent, this *by-demand* agent will drop its current task and will respond to that message.

Both behaviours, *sweeper* and *by-demand*, will keep listening to calls for help from civilians and will keep searching for fire spots, in order to report these events to the proper agent team. Once they finish their tasks, they will be assigned to the *scout* behaviour. Agents with this behaviour will wander through the map helping other police force team agents and searching for survivors and fire spots.

2.3 Fire brigade

Similarly to the ambulance team and police force, fire brigade agents will be divided in sectors of the map. A score will be assigned to each sector, depending on its importance. For the fire brigades, the importance of a sector depends on the number of burning buildings and the disaster potential of the sector. The disaster potential is related to buried civilians, gas stations and building density of a sector. Building density takes into account the total area (ground area $\times$ number of floors) of all buildings in a sector. We want to prevent fire from spreading to buildings with buried civilians, to gas stations and to areas with high building density.

Firefighters will assign a score to each burning building depending on the "effort" needed to extinguish the fire on the building. The effort is related to the amount of water and time required to control the fire on a building. To calculate the effort, we need a model of how fire evolves in a building along time. This model will be given by a regression over the attributes of the building that the fire brigade agent can observe: fieryness, temperature, material and total area.

Data for the regression is extracted from the simulator logs. After several runs on different maps, a significant amount of data from different buildings can be obtained and used as input for the regression.

Task Allocation Firefighters assign a score to each building. The score takes into account the importance of the building (the ones with buried civilians and/or located near gas stations are more important), the agent-to-building distance and effort needed to extinguish the fire. Effort estimation takes as input the prediction of how fire evolves (obtained from the regression model previously discussed) and returns a measure of water and time needed to control the fire. The importance of the building increases its score whereas the distance and the

effort decreases its score (a fire brigade agent will prefer to extinguish fires on close, important buildings that do not require too much effort). Figure 1 shows how a firefighter calculates the score of a building.

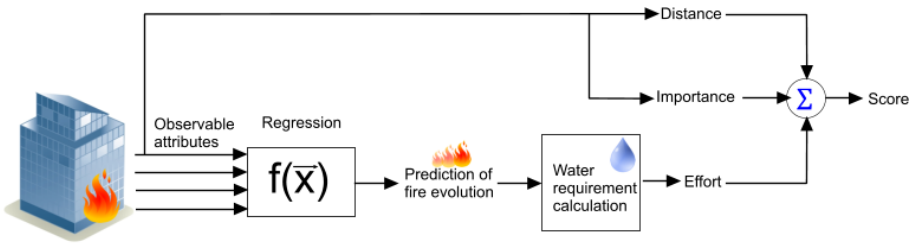

Fig. 1. Building score calculation procedure performed by fire brigade agents.

After deciding which fire to fight, the firefighter calculates whether it is able to extinguish the fire alone. If not, the agent requests help from teammates, using the recruitment method discussed in Section 2.5.

2.4 Center agents

In our strategy, center agents listen to radio communication between agents, accumulating knowledge contained in the messages and replicating important messages to avoid loss of information due to communication unreliability. Using the accumulated knowledge from listening to multiple communication channels, center agents are able to build a more accurate world view. Eventually, center agents send messages to agents in specific channels to give them information previously sent in other channels. This is done in order to make knowledge among agents more uniform.

Center agents do not assign tasks to field agents. Their role is merely informational. The decision of engaging in a task is made individually by each field agent. Therefore, our field agents are autonomous decision-makers. The advantage of this approach is that our strategy does not depend heavily on the center agents. Although their knowledge bases will lack some information provided by the center agents, the decision-making process of our field agents will not change, thus they should not be heavily affected by the absence of center agents.

2.5 Communication

Message construction The goal of communication is to increase the knowledge base of agents and to improve coordination among them. However, in a disaster scenario, communication can be unreliable and bandwidth is limited. To address these problems, our communication strategy is based on a simple protocol. We define three message types for each kind of problem (blockades, burning buildings and buried civilians) the agents may encounter:

- *Report*: the agent found a problem and reports it along with relevant information to its teammates.
- *Engaging*: the agent decided to engage in a problem. *Engaging* messages also contain information about the problem so that other agents can update their knowledge bases even if the problem was reported earlier.
- *Solved*: the agent solved the problem and no further action about it is required.

In order to save radio bandwidth and maximize the amount of data shared between agents, our communication will use a data compression algorithm. Each agent is responsible for compressing/decompressing the communication data.

Recruiting protocol When an agent estimates that it needs help to perform a task, it starts a recruitment process consisting of the following three steps: first, a *request* message is sent by the recruiter agent to other agents. Agents who receive the *request* message and are available to help send back a *committed* message. Receiving *committed* messages allows the recruiter agent to select the teammates that will help best in the task. The recruiter sends *engage* messages to selected teammates and *release* to non-selected ones. Note that *engage* messages used in the recruitment protocol are different from *engaging* messages used in general communication (previously described). This recruitment process is a simplification of the recruitment protocol of eXtreme-Ants algorithm (Section 3). Figure 2 in Section 3 illustrates the recruitment process.

2.6 Path planning

Path planning is divided in static and dynamic path planning. Static path planning occurs at the pre-processing stage, where we calculate the shortest path between all pairs of map nodes. This is done by repeated application of Dijkstra's algorithm. Static path planning generates a table with origin-destination pairs and the route between them. During simulation, agents perform a lookup on this table and save processing time to know the shortest path between two map nodes.

Dynamic path planning occurs when a blockade is detected in a road. When this happens, A* algorithm is executed to calculate the new shortest path between the agent's origin and destination ignoring the blocked road. The new route is updated in the agent's origin-destinations table and a message is sent to other agents to warn them about the blockade. Upon receipt of this kind of message, agents mark the routes containing the blocked road as invalid on their origin-destination table. When the blockade is cleared by police forces, a new message is sent so that the agents can trust their origin-destination table on the entries containing the cleared road again.

3 Multi-agent competition

In the multi-agent competition, the goal is to develop a DCOP algorithm in RMASBench. We choose to implement eXtreme-Ants [4]. This algorithm fol-

lows the extended generalized assignment problem (E-GAP) model of [5] and extends Swarm-GAP [1], a swarm intelligence-inspired task allocation algorithm for dynamic scenarios by explicitly employing a recruiting mechanism.

3.1 Extended Generalized Assignment Problem (E-GAP)

In E-GAP, a set of agents must be assigned to a set of tasks along discrete timesteps. Each agent has a given capability to perform each task. Capability can be regarded as the skill of the agent to perform the task. Each task consumes resources from the agent who performs it.

An agent may perform multiple tasks but a task cannot be performed by more than one agent. Thus, a task that would require multiple agents to execute it must be broken down into smaller, inter-related tasks.

The partial reward given by the allocation of a task by an agent depends on the agent capability to perform the task and whether the task is inter-related with others.

In E-GAP, the total reward is calculated as the sum of the partial rewards of the agents along discrete timesteps. A delay cost is applied as a penalty for not allocating a task in a given timestep. The constraints are that the agents must allocate tasks within their resource limits and that a task can be performed by at most one agent.

3.2 Description of eXtreme-Ants

Inspired by the division of labor in social insects, eXtreme-Ants is an approximate algorithm for the E-GAP.

Observations about swarm behaviors are the base of the model presented in [6], where tasks have associated stimulus and individuals have response thresholds for each task. Let $s_j \in [0, 1]$ be the stimulus associated with task j and $\theta_{ij} \in [0, 1]$ be the response threshold of individual (agent) i to task j. The tendency, or probability, of individual i to engage in task j is denoted by $T_{ij} \in [0, 1]$. The response threshold θ_{ij} of individual i to task j depends on i's capability to perform j ($k_{ij} \in [0, 1]$). The calculation of θ_{ij} and T_{ij} is shown in Eq. 1.

$$T_{ij} = \frac{s_j^2}{s_j^2 + \theta_{ij}^2} \qquad \text{and} \qquad \theta_{ij} = 1 - k_{ij} \tag{1}$$

Regarding independent, i.e., tasks that are not inter-related, in eXtreme-Ants, agents individually decide which task they will engage in a simple and efficient way (via Eq. 1), minimizing computational effort and communication between agents. Agents communicate using a token mechanism. When a given agent perceives new tasks, it creates a token with these tasks. The agent can receive tokens from other agents too. Either way, the holder of a token has the right to decide in which tasks of the token it will engage. The token with the remaining tasks is passed to a random agent that has not held the token before.

To deal with inter-related tasks, i.e. groups of tasks that demand simultaneous execution, eXtreme-Ants uses a process inspired in the recruitment process for cooperative transport observed among ants. When an agent perceives a set of inter-related tasks, it acts as a scout ant. Firstly it attempts to allocate all the constrained tasks. If it fails, then it begins a recruitment process via communication. There are five kinds of messages used in the recruitment protocol of eXtreme-Ants: *request*, *committed*, *engage*, *release* and *timeout*. Figure 2 illustrates the recruitment process, except the timeout mechanism.

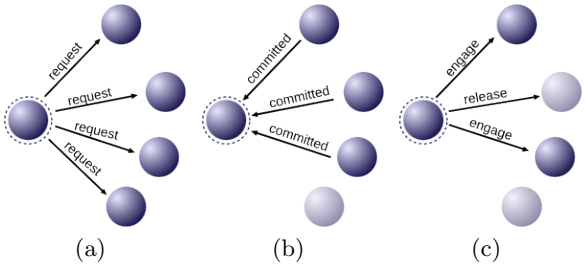

Fig. 2. Recruitment process: (a) recruiter (circled with dashed line) sends *request* messages; (b) agents who decide to help respond with *committed* messages; (c) recruiter sends *engage* to selected teammates and dismisses non-selected ones with *release* messages.

In addition to the steps presented in Fig. 2, in eXtreme-Ants, a *request* message is forwarded when the agent who received it decides to not commit to the task. Besides, there is a mechanism to prevent excessive forwarding of *request* messages: when a *request* message is forwarded too many times, it expires. The agent that received an expired *request* message sends a *timeout* message to the scout (recruiter) agent. The scout, upon receipt of *timeout* messages, sends *release* messages to dismiss agents that had previously committed to the task.

This TDP presents a brief description of eXtreme-Ants. For further details, the reader can refer to [4].

4 Conclusion

To address the challenge posed by the agent competition, we present an approach that relies on autonomous decision-making agents. This means that no field or center agent can directly assign a task to other agent.

Our approach relies on map sectorization in order to limit the scope of action of each field agent. The task allocation of field agents is based on the score that they assign to each task, which is associated with the importance and difficulty level of solving the task.

For the multi-agent competition, we choose to implement eXtreme-Ants in RMASBench. The algorithm compares favorably to other E-GAP based approaches in terms of team reward, computational effort and exchanged messages. This implementation represents a scientific contribution as we can establish the first comparison of E-GAP and factor graph based approaches (e.g. max-sum already implemented in RMASBench) for task allocation in dynamic environments.

Acknowledgment

Authors would like to thank FAPEMIG for supporting the team through its "Santos Dummont" program and to CAPES and CNPq for the researchers' fellowships.

Bibliography

[1] Ferreira, Jr., P.R., Boffo, F., Bazzan, A.L.C.: Using Swarm-GAP for distributed task allocation in complex scenarios. In: Jamali, N., Scerri, P., Sugawara, T. (eds.) Massively Multiagent Systems, pp. 107–121. No. 5043 in Lecture Notes in Artificial Intelligence, Springer, Berlin (2008)

[2] Kitano, H.: Robocup rescue: A grand challenge for multi-agent systems. In: Proc. of the 4th International Conference on MultiAgent Systems. pp. 5–12. Los Alamitos, IEEE Computer Society, Boston, USA (2000)

[3] Kleiner, A., Farinelli, A., Ramchurn, S., Shi, B., Maffioletti, F., Reffato, R.: RMASBench: benchmarking dynamic multi-agent coordination in urban search and rescue. In: Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems. pp. 1195–1196. International Foundation for Autonomous Agents and Multiagent Systems (2013)

[4] dos Santos, F., Bazzan, A.L.C.: eXtreme-ants: Ant Based Algorithm for Task Allocation in Extreme Teams. In: Jennings, N.R., Rogers, A., Aguilar, J.A.R., Farinelli, A., Ramchurn, S.D. (eds.) Proceedings of the Second International Workshop on Optimisation in Multi-Agent Systems. pp. 1–8. Budapest, Hungary (May 2009)

[5] Scerri, P., Farinelli, A., Okamoto, S., Tambe, M.: Allocating tasks in extreme teams. In: Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M.P., Wooldridge, M. (eds.) Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 727–734. ACM Press, New York, USA (2005)

[6] Theraulaz, G., Bonabeau, E., Deneubourg, J.: Response Threshold Reinforcement and Division of Labour in Insect Societies. In: Royal Society of London Series B - Biological Sciences. vol. 265, pp. 327–332 (2 1998)