UvA Rescue
Team Description Paper
Agent competition
Rescue Simulation League
Iran Open 2014

Mircea Traichioiu and Arnoud Visser

Universiteit van Amsterdam, Science Park 904, 1098 XH Amsterdam, NL
http://www.jointrescueforces.eu

Abstract. This year's contribution of the UvA Rescue Team is twofold. On one hand a theoretical contribution is made by describing the planning and coordination problem formally as an POMDP problem, which will allow to apply POMDP-solution methods in this application area. On the other hand the impact of the introduction of flying agents will be studied. Flying agents, when applied correctly, have the potential to reduce the uncertainty in the planning process considerably.

1 Introduction

The UvA Rescue Team has a long history. The first participation in the Rescue Simulation League was by Stef Post and Maurits Fassaert, who competed in the 2003 competition in Paduva [10, 9]. In 2006 the first Virtual Robot competition was held. Max Pfingsthorn and Bayu Slamet participated in this competition and won the Best Mapping award [8]. The team from Amsterdam started a cooperation with Oxford University in 2008, which continued for 4 years [16, 13, 15, 1]. In 2012 the team operated again under its original name; the UvA Rescue Team [14], which resulted in the Infrastructure award.

During those years the team published several journal articles, book chapters and theses. The team described their approach every year in a Team Description Paper and published their source code[1] with a public license. Finally, the details and rational behind the code used in the Virtual Robot competition is described in a Technical Report [12], which also contains a complete overview of our publications (up to 2012).

2 Approach

The intention of our team is to formulate the coordination problem of the Agent competition in such a way that solution methods of the MultiAgent decision

[1] http://www.jointrescueforces.eu/wiki/tiki-index.php?page=Downloads

process toolbox [11] can be used. In earlier contributions [9] it was demonstrated that the coordination problem is too large for an optimal solution, yet in recent years much progress is made in approximate solutions [7]. For instance, the online Bayesian Game Approximation (Forward-Sweep Policy Computation) algorithm [2] has been demonstrated to be effective for large multi-agent problems [4], and has been reimplemented into the MADP Toolbox.

In a first attempt, the coordination problem of the Agent competition is described as a DEC-POMDP, following the description in Oliehoek *et al* [7]. A second attempt is made to formulate the problem in a hierarchical manner, inspired by Oliehoek *et al* [5].

2.1 General approach

State Description The state space for the task of fire fighting in RoboCup Rescue is limited to dynamic factors in the RoboCup Rescue world. Static factors, i.e. factors that do not change throughout the simulation, will not have to be explicitly incorporated in the state space, although they are implicitly present via the transition model. In RoboCup Rescue, the world is given by a map, consisting of buildings, roads and nodes, which act as the glue between roads and buildings. A typical map consists of more than 1000 buildings and the same number of roads and nodes.

This world is populated by civilians, rescue agents (platoons) and the immobile rescue centers. The rescue agents (both mobile and center agents) can in turn be divided into fire, police and ambulance agents. The center agents function as communication centers. In this description only mobile firefighting agents are considered; the same exercise has to be done for the police and ambulance agents.

The mobile agents can be located on roads, nodes or in buildings. So the number of valid positions on a map is the sum of these elements. Also these mobile agents have a particular health, expressed in health points (HPs). Firebrigade agents have a particular amount of water left.

The earthquake that takes place at the beginning of the simulation has a large effect on the state: buildings collapse and catch fire, these collapses can cause roads to get blocked (in either direction) and civilians to get trapped in debris. Starting from this initial state, fires will spread if left unattended. The fire simulator is based on heat energy as primary concept. Fire propagation's main component is heat radiation, which is simulated by dividing the open (non-building) area of the map in cells for which the air temperature is computed. Other factors that determine the spread of fire are properties of buildings. The dynamic factors are the heat of a building, whether it is burning, how much fuel is left and how much water is put in by extinguish actions. Static properties like the size and composition of a particular building (wood, steel or reinforced concrete) and how close it is to surrounding buildings also influence the spreading of fire. However, because these properties are static, they do not need to be incorporated in the state description. Instead the probability of a particular

building i catching fire given that a neighboring building j is on fire is modeled through the transition model.

Actions The actions for an agent i in RoboCup Rescue can be divided in domain level actions $\mathcal{A}_i^d$ and communication actions $\mathcal{A}_i^c$. A mobile agent can perform both a domain level action as communication within one time-step (e.g. a fire-brigade agent can move/extinguish and communicate). This means that the set of actions $\mathcal{A}_i$ for a mobile agent i is the Cartesian product of all domain level and communication actions $\mathcal{A}_i = \mathcal{A}_i^d \times \mathcal{A}_i^c$. In this section we will discuss the domain level actions $\mathcal{A}_i^d$. The subsection at the end of this page will deal with communication actions $\mathcal{A}_i^c$.

All mobile agents can perform the *move* action. The argument of this *move* action is a path along which the agent should move. Clearly the *move* actions are dependent on the current position of the agent. Also, there is a maximum distance that an agent can travel in one time-step (333m). This means that two paths that deviate only after this point lead to the same action. Fire-brigades have 2 specialized actions: *extinguish* and *refill*. The *extinguish* action specifies a building and the amount of water (in liters) to direct to that building.The *refill* action restores the water supply of the brigade and can only be performed at 'refuges', these are special buildings were agents can find shelter.

Observations As with actions we specify the set of observations for agent i as the Cartesian product of domain and communication observations $\mathcal{O}_i = \mathcal{O}_i^d \times \mathcal{O}_i^c$. Here we treat the domain observations $\mathcal{O}_i = \mathcal{O}_i^d, communicationobservations\mathcal{O}_i^c$ are treated in The next subsection.

At each time-step, only objects within a range of 10m are seen, except for fiercely burning buildings, which can be observed from a larger distance. When an agent executed a *move* action, only observations of the new position are received (i.e. no observations are made 'en route'). On average 4-6 static objects (building and roads) can be visually observed during a time-step [9].

Observing an object means that the agent receives the object ID, its type and properties. For a road this property is whether or not it is blocked (in both ways), for a building, the so-called 'fieriness', is observed. This fieriness factor is a direct function of the amount of fuel and water left in the building and determines the part of the area counted as damaged.

Communication Communication is a transaction consisting of both an action (by the sender) $\mathbf{a}^c$ and an observation (for the receiver) $\mathbf{o}^c$. In RoboCup Rescue there are two forms of communication actions: *say* and *tell*. The *say* messages are directly transferred (i.e., shouted), the latter transmitted by radio. Both types of communication are broadcast: *say* messages can be picked up by agents of all types within 30m, *tell* messages can be received by all agents of the same type regardless of the distance. The restrictions that are posed on communication vary per competition. We assume that platoon agents can send 4 *tell* messages

and one *say* message and that all agents can receive all messages. Restrictions on the number of received messages can also be incorporated [6].

In a Dec-POMDP, we can model communication by introducing communication actions and observations. The basic idea is that for each joint communication action $\mathbf{a}^c$ one joint communication observation $\mathbf{o}^c$ can be introduced that for each agent contains the messages sent by the other agents. Restrictions with respect to communication distance can be modeled by making communication dependent on the (next) state s'. That is, it is possible to specify a communication model of the form $\Pr(\mathbf{o}^c|\mathbf{a}^c, s')$.

The complete observation model is then given as the product of this communication model and the regular, domain observation model:

$$\Pr(\langle \mathbf{o}^d, \mathbf{o}^c \rangle | \langle \mathbf{a}^d, \mathbf{a}^c \rangle, s') = \Pr(\mathbf{o}^c|\mathbf{a}^c, s') \cdot \Pr(\mathbf{o}^d|\mathbf{a}^d, s').$$

In a pure planning framework, messages have no a-priori semantics. Instead the planning process should embed the 'optimal meaning' in each communication action.

Transition, observation and reward model The transition model of a factored Dec-POMDP can be compactly described by a two-stage dynamic Bayesian network (DBN). Because the state description is the same as used by the simulator components, these structure and probabilities for this DBN can be found by analyzing the code of the simulation system. For the (domain) observation model we can make a similar argument.

The reward function is easily derived from the scoring function. A typical scoring function is

$$Score(s) = (P + S/S_0) \cdot \sqrt{B/B_0}, \tag{1}$$

where P is the number of living agents, S_0 is the total sum of health points (HPs) at start of the simulation, S is the remaining sum of HPs, B_0 is the total area of houses, B is the area of houses that remained undamaged.

This gives us the reward function of the Dec-POMDP in the following way:

$$R(s, \mathbf{a}, s') = R(s, s') = Score(s') - Score(s).$$

The horizon is finite in the RoboCup Rescue competition (300 time-steps). However in the real-life setting we will typically want to plan for a varying horizon (until all fire is extinguished and all trapped people are either rescued or dead). This can be accomplished by treating the problem as one of infinite horizon.

2.2 Hierarchical approach

As mentioned in subsection 2.1, the state description depends on a large number of factors. Unless an efficient encoding of these elements is devised, the state space becomes prohibitively large, as shown in [7]. One way to address this issue

is to consider an hierarchical approach, such as the one described in [5], under which the general problem is subdivided into multiple, simpler problems, organized hierarchically. For each of these problems, the state space and associated action space are greatly reduced, thus allowing efficient reasoning.

For our hierarchical approach we will consider only two levels of abstraction, with a less conventional method for the macro level and a Dec-POMDP model for the micro-level. The goal of reasoning at the macro level is to efficiently distribute the agents along the map in order to address the existent threats (i.e. rescue victims, extinguish fires and clear roads). Consequently, the role of the agents at the micro level is to effectively adress the threats in the area assigned at the macro level.

Macro level To illustrate the macro-level approach, let us first consider just the problem of rescuing burried and/or injured civilians. Each discovered civilian can be viewed as a sample drawn from a hidden "threat" distribution over the 2D space of the map (a hidden distribution which could be discovered by the flying agents described in section 3). The ambulance agents can then be considered to "generate" a "help" distribution around their position (for example a Gaussian, albeit better choices may be possible). Thus, the goal at a macro level would be to fit the "help" distributions such that they approximate as well as possible the "threat" distribution, which can be done using Expectation Maximization or a similar algorithm.

For this approach to yield satisfactory results, additional constraints must be taken into consideration, such as restricting the values of the covariances of "help" distributions, such that they reflect reasonably small areas which can be dealt with efficiently by an agent at the micro-level. Also, in order to be able to deal with the undiscovered victims, a total number of citizens can be assumed (e.g. a function of the total number of buildings). Subtracting the observed ones from this total number, gives an estimative number of unknown victims, which can then be distributed uniformly over the unexplored space. As more victims get observed, these randomly distributed points decrease in number and the estimation of the "threat" distribution becomes more accurate. After fitting, the means of the "help" distributions would indicate towards which points of the map should the ambulances focus.

Similar methods can be employed for the police and firefighting agents as well. In particular to the firefighting case, the burning buildings can be considered as samples weighted by their fieriness.

Micro level In order to enable an efficient behaviour of the agents at a tactical level, regardless of the actual position on the map where they get assigned by the macro level reasoning, a new state space must be devised. This state space must be general enough so as not to depend on the particularities of the map, yet informative enough to enable meaningful reasoning. However, balancing between these two characteristics, as well as choosing relevant "sufficient statistics" to include in the state description is not straight forward.

Taking into account the fact that each agent type has different specific goals (e.g. ambulances save civilians), as well as specific actions (e.g. only firefighting agents can extinguish fires), it seems reasonable to have specific state and action definitions for each agent type.

Thus, a possible implementation of the state space for the ambulance agents, incorporating the above desired properties, would include the following information:

- distance to the nearest refuge
- distance to the nearest victim
- distance to the nearest unexplored node
- victim on board
- HP of victim on board
- number of citizens in effective range

The first three properties can be discretized as needed to avoid the complications of a continuous state space. Operating upon this state space requires also a new domain action set definition:

- go to closest unexplored node
- pick up nearest victim
- drop carried victim at nearest refuge
- drop victim immediately

Translating the observations received from the environment such as to be able to reason about the newly defined state space and transforming the new actions into standard actions to be sent to the simulation kernel is straight forward. The partial observability character of the model still holds because, even if the observations themselves may be correct (no faulty sensors), the agent cannot observe the entire surrounding area (up to it's "effective" range) entirely, being conditioned by the particularities of the map (which makes this problem partial observable). Thus, for example, it may not be able to observe a victim which is closer to it than a previously observed victim, considered so far as being the closest one. Therefore, it's observations may not allow it to determine the true state.

Since the goals in the sub-problem are the same as in the general problem (i.e. rescue as many victims as possible and prevent as much fire damage as possible), the reward function remains unchanged. However, given the new definitions of the state and action space, the transition probabilities must be learned. Also, the communication observations and actions remain the same.

The fact that each agent makes its own observations and, in general, is aware of only its own actions, along with the particularities highlighted in the previous paragraphs, indicate that the model of the problem at the micro-level is still a Dec-POMDP.

Similar to the ambulance agents, the firefighting agents can incorporate the following elements in their state space definition:

- carried water volume
- distance to nearest refuge
- distance to nearest low burning building
- distance to nearest medium burning building
- distance to nearest high burning building
- number of other firefighting agents in effective range
- distance to nearest unexplored node

Again, we need to define a new domain action set:

- extinguish nearest low burning building
- extinguish nearest medium burning building
- extinguish nearest high burning building
- go to nearest refuge and refill
- go to nearest unexplored node

Finally, the state space definition for the police agents can include the following information:

- distance to nearest unattended blockade
- distance to nearest unexplored node

The domain action set for police agents becomes:

- attend nearest unattended blockade
- go to nearest unexplored node

3 New challenges

Our team is interested in the new challenge of flying agents, which can be used as explorers, as introduced in the Infrastructure competition [3]. This challenge will force to explicitly reason about information gathering about the dynamics of the fire front.

4 Conclusion

The UvA Rescue Team looks forward to be active again in the Agent competition of the Rescue Simulation League.

References

1. N. Dijkshoorn, H. Flynn, O. Formsma, S. van Noort, C. van Weelden, C. Bastiaan, N. Out, O. Zwennes, S. S. Otárola, J. de Hoog, S. Cameron and A. Visser, "Amsterdam Oxford Joint Rescue Forces - Team Description Paper - RoboCup 2011", in "Proc. CD of the 15th RoboCup International Symposium", June 2011.
2. R. Emery-Montemerlo, G. Gordon, J. Schneider and S. Thrun, "Game theoretic control for robot teams", in "Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on", pp. 1163–1169, IEEE, 2005.
3. P. D. Gohardani, P. Ardestani, S. Mehrabi and M. A. Yousefi, "Flying Agent: An Improvement to Urban Disaster Mitigation in RoboCup Rescue Simulation System", in "Proceedings of the 17th RoboCup Symposium", July 2013.
4. Štefan Konečný, *Lossless clustering of multi-agent beliefs to approximate large horizon Dec-POMDPs*, Master's thesis, Universiteit van Amsterdam, October 2011.
5. F. A. Oliehoek and A. Visser, "A hierarchical model for decentralized fighting of large scale urban fires", in "International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)", May 2006.
6. F. A. Oliehoek and A. Visser, "A hierarchical model for decentralized fighting of large scale urban fires", in "Proc. of the AAMAS'06 Workshop on Hierarchical Autonomous Agents and Multi-Agent Systems (H-AAMAS)", pp. 14–21, May 2006.
7. F. A. Oliehoek and A. Visser, *Interactive Collaborative Informations Systems*, *Studies in Computational Intelligence*, volume 281, chapter A Decision-Theoretic Approach to Collaboration: Principal Description Methods and Efficient Heuristic Approximations, pp. 87–124, Springer-Verlag, Berlin Heidelberg, March 2010, ISBN 978-3-642-11687-2, doi:10.1007/978-3-642-11688-9_4.
8. M. Pfingsthorn, B. Slamet, A. Visser and N. Vlassis, "UvA Rescue Team 2006; RoboCup Rescue - Simulation League", in "Proc. CD of the 10th RoboCup International Symposium", 2006.
9. S. B. M. Post and M. L. Fassaert, *A communication and coordination model for 'RoboCupRescue' agents*, Master's thesis, Universiteit van Amsterdam, June 2004.
10. S. B. M. Post, M. L. Fassaert and A. Visser, "The high-level communication model for multiagent coordination in the RoboCupRescue Simulator", in "7th RoboCup International Symposium", (edited by D. Polani, B. Browning, A. Bonarini and K. Yoshida), *Lecture Notes on Artificial Intelligence*, volume 3020, pp. 503–509, Springer-Verlag, 2004.
11. M. T. J. Spaan and F. A. Oliehoek, "The MultiAgent Decision Process toolbox: software for decision-theoretic planning in multiagent systems", in "AAMAS Workshop on Multi-agent Sequential Decision Making in Uncertain Domains", pp. 107–121, 2008.
12. A. Visser, "UvA Rescue Technical Report: A description of the methods and algorithms implemented in the UvA Rescue code release", Technical Report IAS-UVA-12-02, Informatics Institute, University of Amsterdam, The Netherlands, December 2012.
13. A. Visser, G. E. M. de Buy Wenniger, H. Nijhuis, F. Alnajar, B. Huijten, M. van der Velden, W. Josemans, B. Terwijn, C. Walraven, Q. Nguyen, R. Sobolewski, H. Flynn, M. Jankowska and J. de Hoog, "Amsterdam Oxford Joint Rescue Forces - Team Description Paper - RoboCup 2009", in "Proc. CD of the 13th RoboCup International Symposium", July 2009.

14. A. Visser, N. Dijkshoorn, S. van Noort, O. Zwennes, M. de Waard, S. Katt and R. Rozeboom, "UvA Rescue - Team Description Paper - RoboCup 2012", June 2012.
15. A. Visser, Q. Nguyen, B. Terwijn, M. Hueting, R. Jurriaans, M. van de Veen, O. Formsma, N. Dijkshoorn, S. van Noort, R. Sobolewski, H. Flynn, M. Jankowska, S. Rath and J. de Hoog, "Amsterdam Oxford Joint Rescue Forces - Team Description Paper - RoboCup 2010 and Iran Open", in "Proc. CD of the 14th RoboCup International Symposium", July 2010.
16. A. Visser, T. Schmits, S. Roebert and J. de Hoog, "Amsterdam Oxford Joint Rescue Forces - Team Description Paper - RoboCup 2008", in "Proc. CD of the 12th RoboCup International Symposium", July 2008.