Hierarchical Decision Making for Search and Rescue Teamwork

Mircea Trăichioiu Arnoud Visser

Intelligent Robotics Lab, University of Amsterdam
http://www.intelligentroboticslab.nl/

Abstract

This paper focuses on the development of a novel approach for modeling the behavior of agents in the RoboCup Rescue Agent Simulation Competition. A two layered approach is developed, with a macro-level behavior responsible for the strategic, high level decisions and a micro-level behavior dealing with the local particularities of the competition.

The micro-level behavior was implemented using a simple Markov Decision Process. For the macro-level behavior a Bayesian Game Approximation was compared against a Distributed Constraint Optimization problem formulation. Experimental results show a good overall performance of the methods considered, with their differences being better highlighted on harder, custom made configurations of the official contest maps.

1 Introduction

Stringent temporal constraints and limited situational awareness in disasters require efficient multi-agent task allocation methods and the ability to plan under uncertainty. In these types of environments, there is uncertainty on several aspects, such as the location, type and size of tasks, and agents must travel through possibly unsafe areas to reach the tasks. The environment is dynamic, fires will spread, buildings collapse, and civilians die if they are not rescued promptly. All these elements make urban search and rescue a very demanding testbed and, at the same time, a realistic and important application of artificial intelligence techniques.

The ultimate goal in urban search and rescue is to contain the damage, namely, to maximize the number of lives saved while minimizing their injuries, and maximize the amount of infrastructure preserved while minimizing the damage to these infrastructures. The proposed method has these goals as their optimization objectives, and the performance is measured with respect to those objectives.

The RoboCup Rescue Simulation League is introduced to foster research on this field. This competition is introduced in 2001 [5] and attracted many participants in the following years [1]. The competition makes it possible to benchmark different multi-agent coordination algorithms,such as the novel approach presented here.

1.1 Environment

The environment in which the agents take actions consists of (parts of) different cities, either real or computer generated. These cities contain a number of buildings and connecting roads, up to 10000 each. Apart from regular buildings, which make up for the majority of the buildings in the city, two types of buildings have special functions, namely the *refuges* and the *gas stations*. The former represents a safe zone where victims can be dropped and where fire brigades can refill their water tanks, while the latter are potentially dangerous entities, with increased negative effects on their surroundings, should they catch fire.

The spreading of fires is controlled by the fire simulator and is governed by the topology of the map (the adjacency of the buildings), the size of the buildings (both ground area and height) and the construction materials of each building (e.g. wooden buildings catch fire more rapidly than concrete buildings). As such, a measure of the fire level and fire related damage, called *fieriness*, is defined for each building and can take several levels: unburnt, heating, burning, inferno.

The road blockades are controlled by the collapse simulator and are influenced by the initial intensity of the earthquake, the height and fire state of the buildings, as well as the possible aftershocks which occur during the simulation.

Civilians are entities controlled by the environment and represent the general population of the city. Their behavior is fixed and depends on their status. A unburied, uninjured civilian will move towards the nearest refuge, although with a lower speed than the autonomous agents. Buried civilians need to be rescued from the rubble first by the ambulances and then transported to the refuge. Injured civilians are also unable to move by themselves and their condition worsens (represented by a gradually decreasing hitpoint measure) until they are safely returned to a refuge. Should their hitpoint measure drop to 0, they are considered to be dead.

Agents can communicate directly through voice messages up to a certain range, as well as through radio messages. Radio channels are limited in their number and bandwidth, i.e. the maximum amount of information that can be transmitted through the channel in a single time step.

The performance of the agents with respect to their tasks is estimated using a score measure computed for each time step of the simulation. This measure is influenced by the total number of civilians alive and their cumulative health, the speed of their discovery, the total building damage across the city, the speed and efficiency of fire extinguishing and the efficiency of road clearing. Additional details regarding the parameters and behaviors of the aforementioned simulators and the competition can be found in an overview article [11].

1.2 Agents

There are three categories of agents acting within this environment, ambulance agents, fire agents and police agents. *Ambulance teams* are responsible for rescuing civilians or other mobile agents trapped in collapsed buildings and transporting them to refuges. Multiple ambulance teams can work together for rescuing victims from collapsed buildings, resulting in a faster completion of the task, but only one agent can carry a single victim at a time.

The main role of the *fire brigades* is extinguishing fires. They carry a limited amount of water which is depleted gradually while extinguishing buildings and which can be replenished at refuges (where multiple fire brigades can refill simultaneously and at a high rate) or hydrants (where a single fire brigade can refill at a time and at a limited rate). Naturally, multiple fire brigades extinguishing the same burning building will be able to complete the task more rapidly.

Finally, *police forces* are responsible for clearing out roads so that the other platoon agents and victims can move freely. They can clear only a limited area at a time and multiple police forces clearing the same area does not result in increased effectiveness in completing the task.

For the fire brigades coordination inside a team is crucial to contain larger clusters of burning buildings and limiting the expansion of fires. Compared to the fire brigades, the cooperation aspect of the ambulance teams and police forces is somewhat less important and their behavior is in this paper ignored. For more details on the influence of ambulance teams and police forces see the thesis [12].

2 Related work

As mentioned by Ramchurn *et al.* [10]; unfortunately most of the approaches employed by the teams in the Rescue Agent Simulation Competition do not follow a single, consistent formally defined framework, but rather aim at optimizing various isolated aspects of the agents' behavior, which, in turn, lead to measurable increases in the competition score. An exception is the work of Parker *et al.* [9], which actually also assesses the impact of forming heterogeneous teams of agents, either assigned statically, at the beginning of the simulation, or dynamically, as need arises.

3 Theory

In this section we assume that general concepts as Markov Decision Process (MDP) and a Partially Observable Markov Decision Process (POMDP) are well known [4]. Instead, the Bayesian Game Approximation and Distributed Constraint Optimization Problem (DCOP) are introduced.

3.1 Bayesian Game Approximation

As mentioned by Emery-Montemerlo *et al.* [3], while approximate solution algorithms for POMDPs can provide good results with reasonable computational costs, they generalize poorly to decentralized multi-agent settings. This is mainly due to the fact that agents need to maintain parallel POMDPs for tracking their peers' joint belief states, taking all other agents' observations as input. This requires computation costs exponential in number of agents and observations, as well as very demanding communication costs. In a search and rescue scenario communication channels are limited. The method proposed in [3] aims at mitigating these constraints and provides a solution applicable in wider domains.

The problem is formulated as a *Partially Observable Stochastic Game* (POSG), an extension handling uncertainty in world states to stochastic games, themselves a generalization for MDPs in multi-agent scenarios. Following the conventions laid by Emery-Montemerlo *et al.* [3], a POSG is defined as a tuple $< I, S, A, Z, T, R, O >$, each of the elements having similar meanings as their corresponding POMDP counterparts. Thus,

- $I = \{1, ..., n\}$ represents the set of agents;
- S represents the set of world states;
- $A = A_1 \times ... \times A_n$ represents the joint action set (cross product of individual agents' action sets);
- $Z = Z_1 \times ... \times Z_n$ represents the joint observation set (cross product of individual agents' observation sets);
- $T : S \times A \rightarrow S$ represents the transition function;
- $R : S \times A \rightarrow \mathcal{R}$ is the reward function;
- $O : S \times A \times Z \rightarrow \mathcal{R}$ represents the observation emission probability.

Furthermore, the algorithm is restricted only to POSGs with common payoffs (fully cooperative setting) and, as such, the solution concept considered is the Pareto-optimal Nash equilibrium. As it stands, this formulation of the problem is still intractable for reasonably sized problems, as the action and observation spaces are exponential in number of agents.

The solution suggested in [3] involves approximating the POSG as a series of single-step *Bayesian games*. Under the Bayesian game model, each agent has some kind of private information related to the decision making process. This information is called the *type* and generally can be related to uncertainty regarding the utility of the game. Formally, a Bayesian game is defined as a tuple $< I, \Theta, A, p, u >$ where I and A are defined the same as for POSG, Θ denotes the type profile space, i.e. $\Theta = \Theta_1 \times ... \times \Theta_n$, where Θ_i represents the type space of agent i, p is a probability distribution over the type profile space $p \in \Delta(\Theta)$, assumed to be commonly known, and finally $u = \{u_1, u_2, ..u_n\}$ is the utility with u_i being dependent on the action chosen by agent i, a_i and its type θ_i, as well as the actions selected by the other agents and their type profile. Thus this measure is defined as a function $u_i(a_i, a_{-i}, (\theta_i, \theta_{-i}))$.

3.2 Distributed Constraint Optimisation and Max-Sum algorithm

Parting away from the decision-theoretic planning (DTP) methods, one of the other approaches considered concerns the formulation of the behavior as a Distributed Constraint Optimization Problem (DCOP). An example for this is introduced in [10], where the authors use this formulation to solve the issue of "coalition formation with spatial and temporal constraints". As such, the DCOP problem is defined as a tuple $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F} >$ where $\mathcal{A} = \{a_1, a_2, ..., a_k\}$ represents the set of agents, $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ denotes the set of variables, each variable x_i being assigned to exactly one agent (but an agent potentially owning more variables), $\mathcal{D} = \{D_1, D_2, ..., D_n\}$ represents the set of domains for each variable and $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ is the set of functions characterizing the constraints. Thus, each function $f_i : D_{i_1} \times ... \times D_{i_{r_i}} \rightarrow \mathcal{R}$ is defined on the cross product of the domains of the variable set $\mathbf{x_i} \subseteq \mathcal{X}$ onto which depends, with $r_i = |\mathbf{x_i}|$. In the context described in [10], the variable domains consist of tasks which are reachable fast enough by the corresponding agent for it to make a meaningful contribution. Furthermore, the constraint functions reflect the utility of each task, taking into account all variables which can be assigned the given task.

One of the algorithms commonly used for solving such a problem is the Max-Sum algorithm [2]. In order to employ this technique, the problem is formulated as a *factor graph*, a bipartite graph containing two types of nodes, representing variables (from $\mathcal{X}$) and functions (from $\mathcal{F}$). Following the conventions laid out by Bishop [2], under the Max-Sum algorithm, messages are passed between adjacent nodes of the factor graph. There are two distinct types of messages, from variable nodes to function nodes and from function nodes to variable nodes. The first kind, from variable nodes to function nodes is defined as

$$\mu_{x \rightarrow f}(x) = \sum_{l \in ne(x) \setminus f} \mu_{f_l \rightarrow f}(x) \tag{1}$$

where $x_1, ..., x_M$ represent the argument variables of f, other than x and $ne(f)$ represents the neighbor nodes of f. As pointed out by Ramchurn *et al.* [10], the messages to and from variable nodes are sets of values reflecting the total utility of the network for each possible assignment of the respective variable. Once all messages have been passed, each agent can compute on their own the maximum utility based on the received messages, and determine the optimal task it should attend (i.e. the value assigned to its corresponding variable x_i):

$$x_i = \arg\max_x \left[\sum_{s \in ne(x)} \mu_{f_s \rightarrow x}(x) \right] \tag{2}$$

The algorithm is guaranteed to converge to the global optimal solution for cycle-free factor graphs. However in practice, as suggested by MacKay [7], the algorithm generates also good approximate solutions for cyclic graphs.

4 Approach

4.1 Domain challenges

The considered problem involves a map with a large number of buildings and roads onto which multiple agents with limited sensing capabilities act. A straight-forward model of the problem would involve at the very least encoding in the state space the joint set of the fieriness of each building and the positions of each agent. With this information alone, ignoring the partial observability character, the problem becomes intractable even for the simplest of algorithms. A more detailed analysis of the complexity involved is done by Oliehoek *et al.* [8].

Reducing the size of the problem, while still retaining sufficient information for efficient behavior may prove too difficult, if not impossible, using a single model. As such, the solution described in this paper involves using two separate approaches, organized hierarchically, for micro-level and macro-level behavior, respectively. The details of the micro-level can be found in [12], here only the macro-level is described.

4.2 Fire brigade behavior

The following behaviors share a common pattern, with a micro-level approach based on MDP model and a macro-level responsible for suggesting the optimal fire to extinguish for each agent. With the exception of the "Support requests" and "Sample agent" approaches, all macro-level behaviors involve a central agent (either a fire station or a commonly agreed fire brigade leader) gathering fire reports and creating clusters of burning buildings. For each of these clusters an auction is run for determining which agents should be assigned to them. The fire brigades which are not currently engaged in extinguishing fires bid to the requests with their distance to the center of the cluster. The number of 'winners' for each cluster auction is proportional to the total area of the burning buildings in the cluster, and also capped at a maximum value, so as to avoid engaging too many agents in a large cluster and completely ignoring smaller clusters. As the size of the clusters evolves over time, either by new buildings catching fire or by buildings being extinguished, the number of required agents is adjusted accordingly. Thus, subsequent auctions may be run for assigning new agents, or agents can be released. In order for the macro-level decisions to be relevant, platoon agents continuously send updates regarding relevant buildings and positions.

Once agents are assigned to clusters, each of them receives an assigned building from within the cluster. The method for determining these assignments varies with each approach.

4.2.1 Macro-level behavior: Heuristic pairing

The heuristic pairing method takes into account the order in which the buildings have been reported within each cluster. Empirically, one can observe that the most recently reported buildings usually correspond to the edge of the fire front of the cluster. Thus, assigning the most recent buildings first favors extinguishing the outer buildings and containing the fire to a manageable size, ultimately extinguishing it completely.

4.2.2 Macro level behavior: BaGA pairing

The BaGA algorithm involves solving a series of Bayesian games, thus determining at each time step a best response strategy for each agent. These Bayesian games reflect the problems faced at each time step by the agents in their partial observable context. Under the BaGA approach, the types, encoding specific information which distinguishes each of the agents, are restricted to contain the individual histories of observations and actions of the agents.

Since the goal of the macro level is to efficiently allocate agents to fires, a reasonable definition of the **state space** would encode the joint fieriness of all the buildings in the cluster. Because of the limited sensing capabilities of each agent, the **observations** will reflect the fieriness of the observed buildings from within the cluster. Each building from the cluster has an action associated with it, thus **actions** representing final assignments.

4.2.3 Macro level behavior: DCOP pairing

Following the general description of [10] and [6], the final method considered for pairing is based on the task allocation formulation. As such, each agent has a variable associated with it, whose value domain reflects the buildings in the cluster. The problem of the macro-level behavior then translates into finding the optimal-value assignment for these variables, with respect to an utility function. In particular for the studied approach, the utility function used is part of the benchmark software [6] and depends on the fieriness of the building and distance between the building and the agent. The optimal assignment is computed through message passing between the agents (hence the ”distributed” nature of the approach), as described in Section 3.2.

4.2.4 Support requests

This approach assumes a very simple macro level behavior, where the nearest fire brigade is called to a previously unreported fire. Because under this model there is no coordination with respect to building allocation, a more complex micro-level behavior is needed, particularly to take into account the fieriness of the buildings in the immediate vicinity of the agent.

4.2.5 Sample agents

This is the example code given with simulation environment. It is purely reactive; there is no coordination nor communication. With this behavior the fire brigade moves randomly until a fire is observed. It plans a path to this fire and starts to extinguish. A simple, but sometimes surprisely effective approach.

5 Results

5.1 Experimental setup

The maps used in this study based on the official maps of the 2013 RoboCup Competition[1]. To be able to see difference in the performance of the proposed approaches, more difficult versions of the Kobe and Paris map are created. The **Kobe-hard**[2] and **Paris-hard**[2] maps are versions in which the likelihood of additional ignitions throughout the simulation is greatly increased. Additionally, we have also created a reduced version of the **Paris-hard**[2], called **Paris-mini-hard**[2], as the original one proved to be too computationally expensive for one of the tested methods.

An example of such initial configuration of such map, showing the topology, initial fires and agent positions are shown in Figure 1, where one can recognize the Île de la Cité with the Notre Dame de Paris in the lower middle.

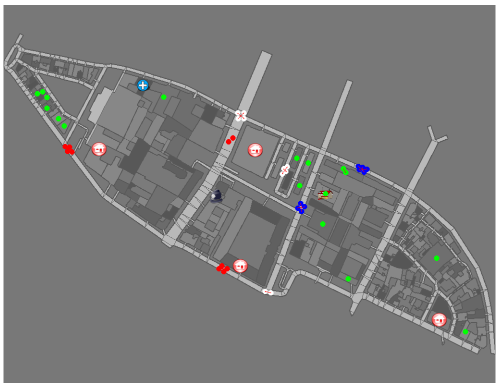

Figure 1: Initial configuration for the Paris-mini-hard map.

In order to optimally assess the performance of the proposed approaches, there are two important changes from the official competition simulations configuration. The first concerns road blockages. Under normal competition circumstances, road blocks impede the free movement of the agents throughout the city and it is the duty of the police agents to clear and maintain proper navigable roads. Since the behavior of the police agents was not the object of this study the road blockages have been disabled. Another particularity of the competition simulations regards the communication model. The maps have a wide range of communication restrictions, with varying numbers of channels, varying bandwidth and reliability. Again, since the topic of this study was agent behavior all the simulations have been run using a single high bandwidth (400Kb) reliable (no dropped messages or noise) radio channel.

In order to ensure consistency, for all experiments related to the macro-level behavior, the micro-level behavior used a fixed deterministic policy.

5.2 Fire brigade macro level behavior

The results obtained for the macro level approaches of the fire brigades are shown in Figures 2 - 3 for two maps. Experiments have been performed [12] on more maps, but those two maps were selected because their results gave the most insight. Designing the right difficulty of a map can only be done emperically; when the map is too easy no difference can be seen between smart and simple algorithms, because every algorithm can solve the puzzle. When the map is too hard, everybody algorithm fails, the difference is that the smart algorithms can cope with the challenge a bit longer. The solution of the RoboCup Rescue is to give more weight the maps where the performance of the teams shows the largest spread, which are maps with appropriate level of difficulty. The *Kobe-hard* and *Paris-hard* maps were selected because they showed the largest spread between the algorithm with the best and worst performance.

As expected, the weakest performance on all the maps is yielded by the "Sample agents". Only on maps with low difficulty their reactive approach manage an acceptable result. In the presented maps,

[1]The official maps are available for download at `http://roborescue.sourceforge.net/2013/results/`.
[2]The custom maps are available for download at `http://goo.gl/gKLCHK`

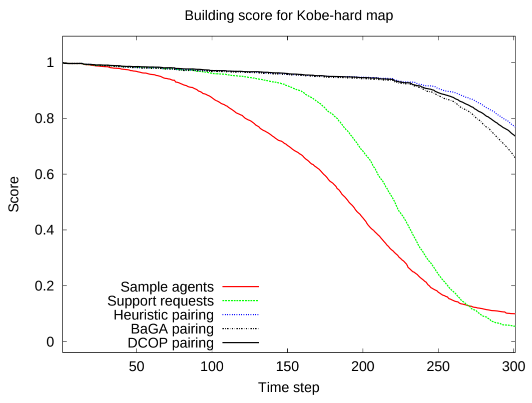

Figure 2: Building score associated with the performance of various macro level approaches on the Kobe-hard map.

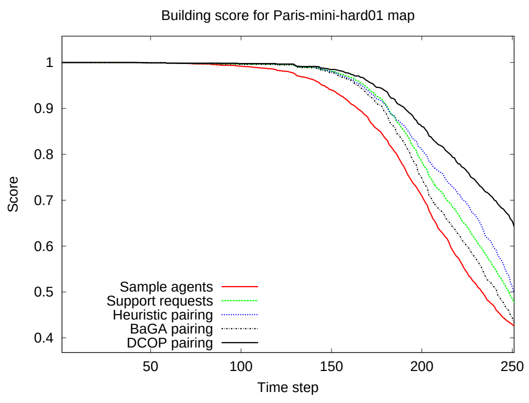

Figure 3: Building score associated with the performance of various macro level approaches on the Paris-mini-hard map.

Kobe-hard and Paris-mini-hard, fires ignite randomly throughout the simulation, the agents are constantly engaged in extinguishing them and the score does not settle to a fixed value, as is the case in the other maps typically used during competition. For Kobe-hard, the steady controlled decrease of the score reflects the agents' ability to contain the fires. As soon as the fires go out of control the score rapidly decreases. This happens more quickly for the "Support requests" approach, starting at around timestep 130. Apart from "Sample agents", all other three approaches yield better performance, being able to contain the fires until roughly timestep 230, with close performance scores afterwards. The typical situation when the fires go out of control, for the "Heuristic pairing", "BaGA pairing" and "DCOP pairing", consists of a large number of agents refilling at refuges and not being able to contain rapidly expanding clusters of burning buildings while they still are of manageable size.

An important note must be made regarding the "DCOP pairing" method. Because of the very high memory demands of the Max-Sum algorithm [6], when clusters grow too large and numerous agents are involved, the thread responsible for computing the pairings runs out of memory (in the experiment the agents are allocated a total of 4GB of memory). For the runs used for plotting the results in Figure 2, the average time of running out of memory is around timestep 232. Judging by the evolution of the score prior to that point and following that point, it is likely that higher level pairing done would not have impacted greatly outcome for the interval 232-300.

The Paris-hard-mini map was especially designed that the DCOP pairing method also ran out of memory on average around time step 230 and, as such, the performance shown in Figure 3 also reflects the macro-level behavior. With the exception of "Sample agents", all approaches manage to contain the fires until around timestep 140. Following that point, "DCOP pairing" yields the best performance, while for the other approaches the performance drops to similar values towards the end. The rate of the descend suggests that the fires are too large to be controlled between timesteps 160-250, and therefore for this interval the differences between the approaches might not be relevant.

As a general conclusion regarding the performance of the various coordination methods for Fire Brigades, all the studied approaches yield acceptable performance on most maps. "Heuristic pairing" yields its best results in the scenarios where its core assumption holds. "BaGA pairing", although not the best method on all maps, has a good overall performance. Lastly, lacking more elaborate coordination, "Support requests" usually has the lowest performance of the considered approaches, though still leading to acceptable scores and considerably outperforming the "Sample agents".

6 Conclusion

In this paper a novel approach for modeling the behavior of agents in the RoboCup Rescue Simulation competition is evaluated. The approach follows a hierarchical structure, with a macro-level behavior focusing on the higher strategic decisions and agent collaboration and a micro-level behavior managing the local, tactical behavior, and rapidly responding to changes in the environment.

While micro-level behavior was restricted to a simpler MDP-based approach, several methods have been investigated and tested for developing the macro-level behavior of fire brigades. Following the decision-theoretic planning paradigm, a method based on the BaGA algorithm allowed the mitigation of the limitations usually associated with multi-agent partially observable domains and yielded consistently good results on the maps tested. Another method, popular in the scholarly works studying the RoboCup Rescue domain, involved formulating the macro-level behavior under the DCOP framework, subsequently solved using the Max-Sum algorithm. This approach had a similar performance to the BaGA based method, sometimes even outperforming it, but its use was limited in certain cases due to the high memory demands. However, this limitation could be addressed by employing adaptations of the original algorithm specifically designed for this problem, such as the Fast Max-Sum algorithm [10].

The described methods were evaluated at the 2014 RoboCup competition and resulted in a high score in the multi-agent challenge of the Rescue Simulation league. Our team had the highest score on the map where the performance of the teams shows the largest spread. These results are also socially relevant. Inside the RoboCup Rescue community several researchers have used the developed algorithms to train and support incident commanders [1].

References

[1] H Levent Akın, Nobuhiro Ito, Adam Jacoff, Alexander Kleiner, Johannes Pellenz, and Arnoud Visser. Robocup rescue robot and simulation leagues. *AI magazine*, 34(1):78–86, 2013.

[2] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[3] Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, and Sebastian Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the international conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 136–143, 2004.

[4] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[5] Hiroaki Kitano and Satoshi Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI magazine*, 22(1):39–52, 2001.

[6] Alexander Kleiner, Alessandro Farinelli, Sarvapali Ramchurn, Bing Shi, Fabio Maffioletti, and Riccardo Reffato. Rmasbench: benchmarking dynamic multi-agent coordination in urban search and rescue. In *Proceedings of the international conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2013)*, pages 1195–1196, 2013.

[7] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[8] Frans Oliehoek and Arnoud Visser. A hierarchical model for decentralized fighting of large scale urban fires. In *Proc. of the Workshop on Hierarchical Autonomous Agents and Multi-Agent Systems (H-AAMAS)*, pages 14–21, 2006.

[9] James Parker, Ernesto Nunes, Julio Godoy, and Maria Gini. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *Journal of Field Robotics*, 2015.

[10] Sarvapali D. Ramchurn, Alessandro Farinelli, Kathryn S. Macarthur, and Nicholas R. Jennings. Decentralized coordination in robocup rescue. *The Computer Journal*, 53(9):1447–1461, 2010.

[11] Cameron Skinner and Sarvapali Ramchurn. The robocup rescue simulation platform. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1647–1648, 2010.

[12] Mircea Trăichioiu. Hierarchical decision theoretic planning for robocup rescue agent simulation. Master's thesis, Universiteit van Amsterdam, October 2014.