Hierarchical Decision Theoretic Planning for RoboCup Rescue Agent Simulation

Master Thesis

Mircea Trăichioiu

Universiteit van Amsterdam

Detail from *View of the Great Fire of Pittsburgh*, 1846, a painting by witness William Coventry Wall. The disaster occurred on April 10, 1845 and had man-made causes. Multiple factors such as the building composition and unfavourable weather conditions, combined with poor fire fighter response lead to the destruction of a third of the city.

Hierarchical Decision Theoretic Planning for Robocup Rescue Agent Simulation

Mircea Trăichioiu
10347542

Master thesis
Credits: 42 EC

Master Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dr. Arnoud Visser

Intelligent Systems Lab Amsterdam
Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

October 1st, 2014

Abstract

This thesis focuses on the development of a novel approach for modelling the behaviour of agents in the RoboCup Rescue Agent Simulation Competition. The scenarios used in this competition consist of multi-agent settings reflecting the challenges faced by ambulances, fire brigades and police forces in the aftermath of a natural disaster, such as an earthquake.

The underlying framework chosen for this task falls under the Decision Theoretic Planning paradigm. However, due to the complexity of the considered problem, using a single model for the agents' behaviour could prove unfeasible. A rich enough model taking into account all relevant particularities of the domain would be intractable, while a much simpler model would abstract away too much information. As such, a hierarchical control structure is considered, with a macro-level behaviour responsible for the strategic, high level decisions and a micro-level behaviour dealing with the local particularities of the environment.

Several method were considered in developing the macro level behaviour, including one based on the Bayesian Game Approximation, an algorithm for finding approximate solutions in multi-agent partially observable domains, and another following the DCOP formulation, a popular framework in the scholarly literature studying this domain. The micro level behaviour was implemented using a simpler MDP based method, due to the strict computation limits.

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.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Rescue Agent Simulation League

RoboCup is an annual international robotics competition focused on promoting research in the fields of robotics and Artificial Intelligence. Within RoboCup there are multiple leagues aimed at various tasks, such as football playing (RoboCup Soccer), disaster relief (RoboCup Rescue) or robot integration in domestic environments (RoboCup@Home). The subject of this thesis focuses on the RoboCup Rescue Agent Simulation league which was organised for the first time in 2001 as a reaction to the Great Hanshi-Awaji earthquake which hit Kobe City on the 17th of January 1995 with devastating consequences [1]. Within this league, multiple agents representing ambulances, fire brigades and police forces act within a city following an earthquake aiming at rescuing victims, controlling and extinguishing fires and clearing rubble from the roads.

1.1.1 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. Apart from the buildings and the roads, *fire hydrants* provide additional, albeit more limited points where the fire brigades can refill their water tanks.

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 behaviour 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. In order to simulate communication difficulties common in the aftermath of a disaster, various constraints are imposed in each scenario, adding noise or randomly dropping the messages partially or totally. In more extreme scenarios, communication is not available at all.

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 behaviours of the aforementioned simulators can be found in the official rules of the competition [2].

1.1.2 Agents

As previously mentioned, there are three categories of agents acting within this environment, ambulance agents, fire agents and police agents. Within each category two types of agents are defined: immobile and mobile.

Immobile agents, called *centre agents*, are represented by buildings and can only have

Agent type	Actions
Centre agent	Speak
Ambulance team	Speak, Move, Rescue, Load, Unload
Fire brigade	Speak, Move, Extinguish, Refill
Police force	Speak, Move, Clear

Table 1.1: Available agent actions according to their type.

a role in the coordination of the mobile agents, the only actions available to them being communication actions. Depending on their category, the centre agents are called *ambulance centres*, *fire stations* and *police offices*, respectively. To be noted is the fact that not all scenarios may include some or all of the centre agents.

Mobile agents, called *platoon agents*, have an active role in addressing the challenges posed by the environment and each have different particularities according to their role.

Thus, *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.

In addition to their specific actions, all platoon agents can take communication actions (*speak* commands) and movement actions. An overview of all the actions available to all the agents is given in table 1.1.

In figure 1.1, an intermediate stage if a simulation is presented. Platoon agents are represented by red, white and blue circles, depicting fire brigades, ambulance teams and police forces respectively. Victims are represented by green circles and the darker the colour shade, the lower their hit point measure. Buildings are usually depicted with varying shades of grey, indicating their collapse damage. Burning buildings are indicated by varying shades of yellow and red, according to their fieriness. Extinguished buildings are shown in shades of blue, depending on their damage, while completely

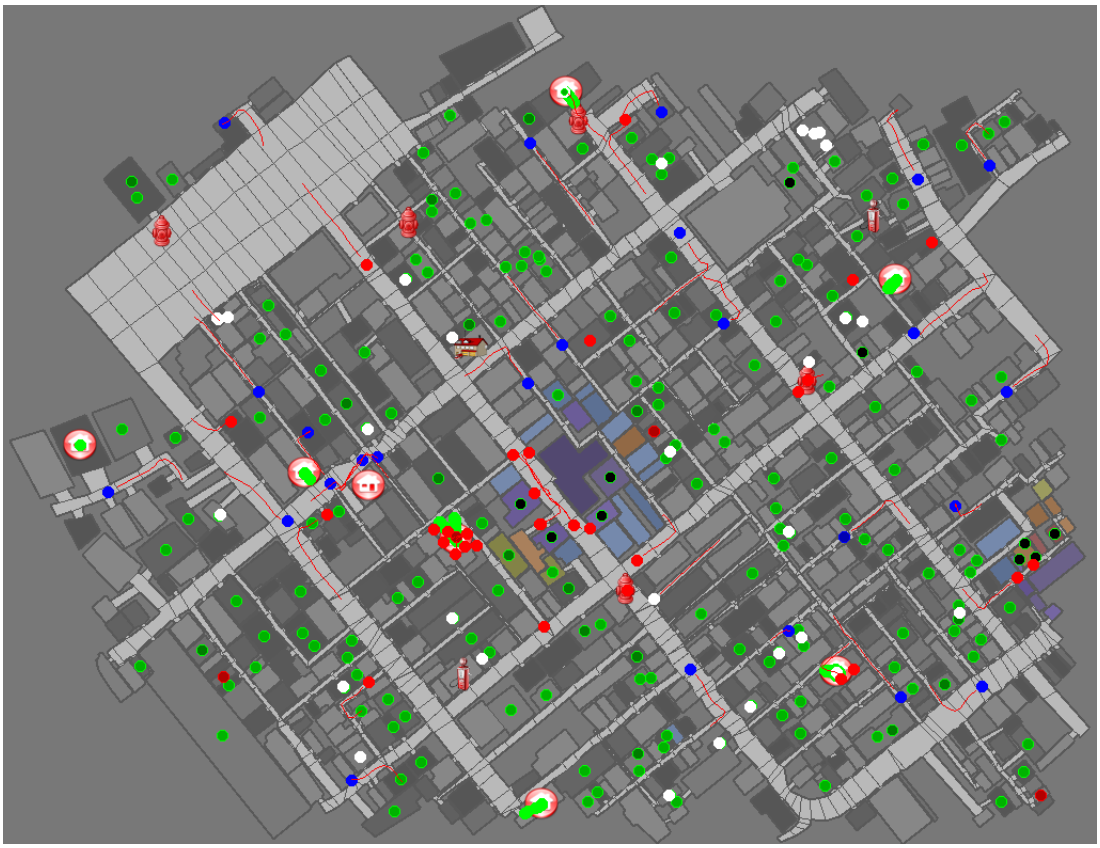

Figure 1.1: Example of an intermediate step in a simulation.

burnt buildings are shown in black. Special buildings such as refuges or centre agents are marked with specific icons.

Usually, the behaviour of a certain agent type also affects the performance of the other agent types. For example, the performance of police agents in clearing roads affects the mobility of fire brigades and ambulance teams, and thus, indirectly, their ability to fulfil their tasks efficiently. Also, the ability of fire brigades to contain and extinguish burning buildings influences the expansion of fires to areas with trapped victims and thus, affects the performance of the ambulance teams. However, the approaches studied in this thesis will only focus on the behaviour of fire brigades and ambulance teams. The main reason for this choice consists of the particularities of the road clearing method employed by the police, shown in figure 1.2. As opposed to fire brigades and ambulances, whose interactions with the environment are made through the use of entity IDs (which can be viewed as higher level semantic information), police force agents clear roads by specifying the location and rotation of the effective clearing rectangle. Thus, the main challenge of the police's behaviour is a geometric optimisation of the clearing area, which would translate poorly in the approaches studied for developing rational agent behaviour and cooperation. Furthermore, the cooperation aspect of the police force's

behaviour is relatively limited, as multiple agents clearing the same blockade do not provide additional benefits.

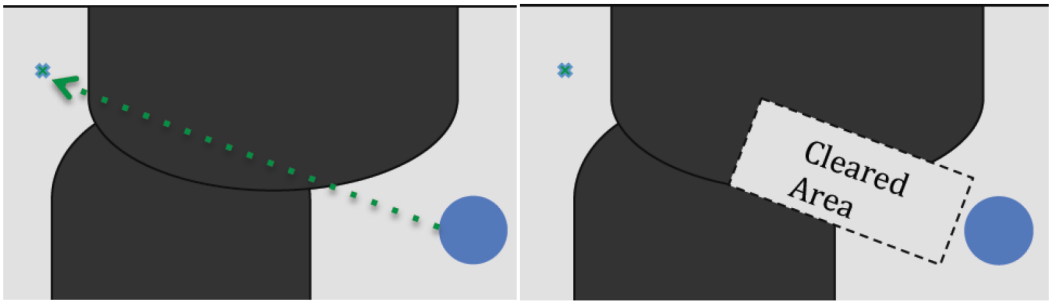

Figure 1.2: Police team clearing method.

1.2 Decision-theoretic planning and learning

According to [3], decision-theoretic planning (DTP) is a general approach for enabling autonomous agents to devise courses of action (policies) in environments which may involve uncertainty with respect to the outcome of the undertaken actions, incomplete information about the world and potentially multiple solutions of varying quality. These particularities make DTP an attractive choice for developing rational agent behaviour in the previously described context of the Rescue Agent Simulation competition.

The majority of DTP methods use the *Markov Decision Process* (MDP) framework as an underlying model [3]. Under this model, the agent interacts with the environment by taking *actions* which usually causes it to change its *state*. Taking an action also causes the agent to receive a *reward*, either positive or negative, reflecting how desirable the effect of the action was. For an environment with known dynamics, finding an optimal behaviour for the agent is a *planning* task, while for an environment with unknown dynamics rational behaviour can be achieved through *learning*. Although robust methods exist for both these kinds of tasks, an MDP model may prove to limiting for the entire Rescue Agent Simulation problem.

Fortunately, DTP also provides richer frameworks taking into account certain particularities of the problem. Partial observability and the multi-agent character are the most notable such particularities. Although several options exist incorporating these aspects, probably the most popular is the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [30]. However, the closer fidelity to the problem definition comes at a cost, as finding optimal or bounded approximate solutions to Dec-POMDP is proven to be NEXP-Complete [29]. This drawback is addressed by certain heuristic approximate methods, but even so, the domains onto which they obtain reasonable

results computationally-wise remain much smaller than the Rescue Agent Simulation problem.

The main contribution of this thesis is the development a hierarchical control method aimed at mitigating the limitations of using a single framework, either too limited as the MDP, or computationally intractable as a Dec-POMDP. As such, the macro-level behaviour is responsible for the higher strategic decisions and agent coordination, while the lower, micro-level behaviour addresses the local, tactical challenges of each agent and implements the suggestions received from the macro-level. This control scheme allows both levels to abstract complementary parts of the problem, making both layers of the decision making process manageable. A further contribution consists of the actual usage of DTP methods in developing an approach capable of performing in the real Rescue Agent Simulation competition and not only in benchmarks or artificial scenarios inspired by it.

This thesis will focus on the performance associated with the macro level behaviour based on the BaGA and DCOP methods. The former was chosen due to its good performance in multi-agent partially observable domains larger than typical benchmark problems. The latter approach was chosen due to its popularity and promising results in the scholarly work studying the RoboCup Rescue Agent Simulation problem, as shown in the following section.

1.3 Related work

As mentioned in [5], 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 optimising various isolated aspects of the agents' behaviour, which, in turn, lead to measurable increases in the competition score. Still, some valuable insight can be found in some of the participating teams' approaches. Regarding the extinguishing behaviour, several teams perform a clustering of the burning buildings and then make assignments to fire brigades based on heuristics [10], convex hulls [12] or a combination of both [7]. The more elaborate approaches use a lightweight simulator to assess the possible spreading of fire and make assignments accordingly [13] [16]. Ambulance behaviour usually involves ranking the victims and buried agents according to a heuristic and then attending to the most promising targets [12] [13] [17] [18]. Only one of the 2013 teams, MRL [16], reports using a learning mechanism for better estimating the relative importance of victims. Another common aspect considered by most teams concerns the map partitioning into smaller, more manageable areas which get assigned to subsets of agents. This partitioning is usually done using K-Means [12] [16] [18] or variants such as X-Means [14] or C-means [17]. One of the teams [11]

also assesses the impact of forming heterogeneous teams of agents, either assigned statically, at the beginning of the simulation, or dynamically, as need arises. Other aspects presented in the team description papers concern optimizations to the path planning algorithms [8] [18] and communications [14] [17].

One of the notable scholarly efforts to formalise and analyse the challenges and solutions to the Rescue Agent Simulation is [5]. The authors consider a *Coalition Formation with Spatial and Temporal constraints* (CFST) model for describing the general process of task allocation for ambulance and fire brigades. The authors claim that in many circumstances the formation of coalitions (teams of agents) is critical for the success of most tasks, as, for example, fires tend expand to multiple buildings, beyond a single agent's ability to contain and extinguish it. The spatial constraints of the CFST model encode the particularities regarding the positions of the agents relative to the position of the task, while the temporal constraints refer time sensitive properties of the task, such as the gradual decrease of victims' hitpoints or the incremental expansion of the fire clusters. This general model is further formalised as a Distributed Constraint Optimization problem, allowing an optimal task assignment with respect to the spatial and temporal constraints through message passing between the agents. This optimal assignment is computed using a variant of the Max-Sum algorithm, adapted to the particularities of the problem, for improved memory consumption and speed.

The authors of [20] also consider a task allocation model, Extended General Assignment Problem (E-GAP) previously introduced by [21], and describe two algorithms for solving it. The first, Low-communication Approximate DCOP (LA-DCOP), relies on creating tokens describing tasks based on the observed state of the environment and then passing them between agents. If an agent has enough capability to successfully address the task it retains the token, otherwise it passes to another agent. However, the choice of retaining or passing the token is founded only on local information and thus the global solution may not be optimal. The second algorithm, Swarm-GAP, follows a similar approach, but uses a probabilistic decision process for accepting or rejecting the token, making it less demanding computationally wise and slightly more efficient.

A further attempt at solving the coalition formation and task allocation is made in [22]. However, the method of choice in this case, *bee clustering*, is inspired by bee behaviour in foraging nectar. More specifically, bees travel far away from the hive to gather nectar and, depending on the sources found, perform a recruitment of other bees based on the gathered information. This metaphor resembles to a certain degree the formation of coalitions of agents with similar or complementary capabilities in order to address tasks of various requirements, scenario similar to the previously mentioned models for the Rescue Agent Simulation.

A thorough overview and synthesis of the DTP paradigm, highlighting its core assump-

tions, strengths and weaknesses is made in [3]. A more in depth analysis and discussion on various algorithms for solving problems under DTP models is given in [25]. One of the more popular frameworks taking into consideration the partial observability and multi agent characteristic of problems, Dec-POMDP, is presented in [30]. Finally, Bayesian Game Approximation, a robust algorithm aimed at mitigating the prohibitive costs associated with optimal solutions of Dec-POMDPs is introduced in [24].

1.4 Structure

The rest of this thesis is organised as follows. Chapter 2 gives a theoretical overview of the algorithms and techniques used in developing the approaches studied in this thesis. Chapter 3 describes the approaches in more detail, highlighting the particularities and adaptations of the algorithms to the actual problem of the Rescue Agent Simulation. Experiments assessing the aforementioned approaches are discussed in Chapter 4 and finally, conclusions are drawn in Chapter 5.

Chapter 2

Theory

The purpose of this chapter is to provide an overview of the algorithms and techniques employed in the studied approaches. Details on how these are used and adapted for the Rescue Simulation problem are given in chapter 3.

2.1 General concepts

The Decision Theoretic Planning (DTP) paradigm served as a starting point for designing the agents' behaviour and constitutes an important part of the overall approach studied in this thesis.

According to [3], DTP is an extension to the classical AI planning paradigm which allows the incorporation of various particularities of real-world problems such as uncertainty with respect to the outcome of actions or limited resources. Following the conventions used in [25], under DTP, the problem is formalised as involving one or more rational decision makers, called *agents*, which act within an *environment* towards achieving a *goal*. In most situations, as in the case studied in this thesis, the time scale is discrete. Thus, at each time step t, the agent is given a representation of the environment's state, $s_t \in \mathcal{S}$. Based on this information, the agent is capable of taking one or more actions $a_t \in \mathcal{A}(s_t)$. The quality of each action is encoded through a numerical reward obtained at the next time step, $r_{t+1} \in \mathcal{R}$. This cycle of observing the state s_t, taking action a_t and receiving the reward r_{t+1} is repeated for a fixed number of steps (finite *horizon*) or indefinitely (infinite *horizon*). The agent's behaviour is described through a *policy*, $\pi_t(s, a)$ which denotes the probability of action a being taken at time t if the environment is in state s.

As opposed to classical planning, where the goal of the agent specified as a world state in which it must arrive, the goal of the agent under DTP is to maximise its accumulated

reward over time. This measure, called *return* is defined for problems with finite horizons as

$$R_t = r_{t+1} + r_{t+2} + ... + r_T \tag{2.1}$$

where T is the final time step. In the case of infinite horizon, the concept of *discounting* is necessary to keep the return finite. Thus, a discount rate γ is introduced, $0 \leq \gamma \leq 1$, reflecting the fact that more immediate rewards are deemed more important than later ones:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.2}$$

2.1.1 Markov Decision Process

The underlying model for the vast majority of DTP methods is the Markov Decision Process (MDP). In addition to the state space $\mathcal{S}$, action set $\mathcal{A}$ and reward model, the problem is specified by the *transition probabilities*, reflecting the probability of the environment transitioning from s_t to a certain s_{t+1} when action a_t is taken. In the case of the MDP, this measure is conditioned only on the previous state s_t and action a_t, and not on the entire history $s_0, a_0, s_1, a_1, ...s_t, a_t$.

$$\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{2.3}$$

Similarly, the reward follows the same conditioning:

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \tag{2.4}$$

This is known as the *Markov property*, with s_t being a sufficient statistic for the entire history. The authors of [25] point out the fact that even if the state signal does not have this property, it is often useful to consider it as an approximation of a Markov state. Mainly, this is due to the benefits of considering the state to be a good basis for predicting future rewards or future states (in case the model is learnt and not specified beforehand).

In solving MDP models, an useful construct is the *value function*, reflecting how desirable is a state from an agent's point of view (or, alternatively, how desirable is an action to be taken from a certain state of the environment). Since these estimates are based on the expected return, and since the return is dependent on the agent's actions, the value function is therefore dependent on the agent's policy (as the policy determines which actions will be taken in certain states of the environment). Thus, for MDPs, the

value function associated with a certain state s under a certain policy π is defined as

$$V^{\pi}(s) = E_{\pi}\{R_t|s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\right\} \tag{2.5}$$

Similarly, for the value function reflecting the benefit of taking a certain action a in state s under policy π is defined as

$$Q^{\pi}(s,a) = E_{\pi}\{R_t|s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\right\} \tag{2.6}$$

These value functions follow a fundamental property which is exploited by many approaches for solving MDPs and other related domains. This property is expressed by the *Bellman equation for* V^{π} and it highlights the relationship between the value of a state and its possible successors, under a given policy π:

$$V^{\pi}(s) = \sum_{a}\pi(s,a)\sum_{s'}\mathcal{P}^{a}_{ss'}\left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right] \tag{2.7}$$

Similarly, for the action-value function, the Bellman equation takes the following form:

$$Q^{\pi}(s,a) = \sum_{s'}\mathcal{P}^{a}_{ss'}\left[\mathcal{R}^{a}_{ss'} + \gamma\sum_{a'}\pi(s',a')Q^{\pi}(s',a')\right] \tag{2.8}$$

Solving an MDP usually implies finding the best policy i.e. the policy which yields the highest return. Expressed in terms of value functions, this equates to finding an optimal policy π^* such that

$$V^{*}(s) = \max_{\pi} V^{\pi}(s), \forall s \in \mathcal{S} \tag{2.9}$$

Similarly, the optimal policies also yield the same optimal action-value function

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a), \forall s \in \mathcal{S} \text{ and } \forall a \in \mathcal{A}(s) \tag{2.10}$$

Since the optimal value functions also respect equations 2.7 and 2.8, these can be rewritten in an alternative form, the *Bellman optimality equations*, for V^* and $Q^*(s,a)$, respectively

$$V^{*}(s) = \max_{a}\sum_{s'}\mathcal{P}^{a}_{ss'}\left[\mathcal{R}^{a}_{ss'} + \gamma V^{*}(s')\right] \tag{2.11}$$

$$Q^*(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s',a') \right] \tag{2.12}$$

For a finite MDP with N states, the Bellman optimality equation yields a system of N equations with N variables which can be solved in principle if the dynamics of the environment are known, namely the transition probabilities $\mathcal{P}^a_{ss'}$ and rewards $\mathcal{R}^a_{ss'}$. However, this approach to solving MDPs is rarely useful, since usually the number of states is prohibitively large to allow reasonable computation times and memory requirements. Furthermore, the dynamics of the environment are not always accurately known. In order to mitigate this constraints, according to [25], three classes of algorithms are usually employed, depending on the particularities of the problem.

The first of these consists of dynamic programming (DP) methods, such as *policy iteration* and *value iteration*. These involve an iterative process for gradually approximating the optimal policy or value function. However, they also require the transition probabilities and rewards to be known in advance. For this reason, they cannot be used in the studied approaches (as will be discussed in chapter 3) and thus will not be discussed further.

The second class of algorithms mentioned in [25] consists of Monte Carlo methods. In contrast to dynamic programming methods, these do not require an a priori knowledge of the environment, but instead approximate value functions based on samples of online experience or simulations. However, these samples are defined only for episodic tasks and reflect complete returns. As such, the update of the value function and policy only occurs between episodes. In this case, a generic update of the value function for a certain state s_t can be expressed as

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \tag{2.13}$$

where R_t denotes the episode's return following time t and α a learning rate parameter. This can be cumbersome for relatively long simulations as the ones studied in this thesis and therefore will not be discussed further.

Finally, the third class of algorithms is Temporal Difference (TD) learning. TD methods combine advantages of both previously mentioned classes, being able to determine optimal policies from experience, without a model of the environment dynamics (trait at the core of Monte Carlo methods), while also bootstrapping on the estimates and taking advantage of the learnt information online, not waiting for the episodes to finish. Because of these particularities, part of the agent's behaviour includes TD methods and will be discussed in the following subsection.

2.2 Temporal Difference learning

Similar to Monte Carlo methods, Temporal Difference methods also perform value function estimates. However, these are not done at the end of an entire episode, but rather at every time step of the simulation. In order to be able to do this, it needs to bootstrap on to an existing estimate. In the simplest case, called TD(0) by the authors of [25], the update is expressed as

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2.14}$$

This update equation is similar to the MC version, equation 2.13, with the return R_t being replaced by the bootstrapped estimate $r_{t+1} + \gamma V(s_{t+1})$. The learning parameter α controls the impact of the new information over the already learned value function. A typical value for this parameter is $\alpha = 0.1$ [25].

Based on this principle, a control strategy can be devised, allowing the agent to find an optimal policy. Because under TD methods the agent needs to learn the environment dynamics while also benefiting off the learnt estimates, a balance must be found between *exploration* and *exploitation*. A total focus on exploration will result in a irrational behaviour of the agent, exploring all the states and actions, but not benefiting at all from what it learns. On the other hand, a total focus on exploitation will potentially not allow the agent to discover an optimal policy, remaining stuck in a local optimum. A commonly employed solution to this problem is the use of an ϵ-*greedy policy*, favouring the choice of the action yielding maximum gain, but not completely ignoring the other. As such, all nongreedy action have a $\frac{\epsilon}{|\mathcal{A}(s)|}$ probability of being chosen, while the remaining probability mass of $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}$ goes to the greedy action. A commonly used value for this parameter is $\epsilon = 0.1$, as it allows sufficient exploitation of the learned policy, while not completely stopping exploration. More elaborate approaches such a simulated annealing allow the variation of this parameter from a high value in the beginning, stimulating exploration, to a lower value as the simulation proceeds, favouring the exploitation of the more informed learned policy.

2.2.1 SARSA

SARSA represents an on-policy TD control method. Rather than estimating a state-value function, it is more convenient to learn an action-value function as it would allow a more accurate policy to be derived, by also learning about $\mathcal{P}^a_{ss'}$. Being an on-policy method, the goal is to estimate $Q^\pi(s, a)$ for the current policy π, for all states s and actions a. In this case, the transition occurs between state-action pairs, rather than

Algorithm 2.1 SARSA algorithm

Initialise $Q(s, a)$ arbitrarily
loop for each episode
    Initialise s
    Choose a from s using policy derived from Q (ϵ-greedy)
    loop for each step in episode
        Take action a, observe r, s'
        Choose a' from s' using policy derived from Q (ϵ-greedy)
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
        $s \leftarrow s'; a \leftarrow a'$
    end loop
end loop

states. As such, equation 2.14 becomes

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{2.15}$$

Based on this update equation, the on-policy control scheme continually estimates Q^π and shifts the policy π towards greediness with respect to this measure. This algorithm is presented in listing 2.1.

2.2.2 Q-Learning

Q-Learning represents an off-policy TD control algorithm. As opposed to SARSA, which estimates the action-value function Q_π for the current behaviour policy, Q-Learning aims at directly approximating the optimal action-value Q^*, irrespective of the policy followed. As such, the update equation becomes

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.16}$$

In this case, the main difference from equation 2.15 is the fact that the action chosen for s_{t+1} is not specified by the policy π, but instead is the one yielding the highest expected value, as selected by the max operator. This enables a faster convergence to the optimal policy, but in certain cases the online behaviour might be slightly worse than for SARSA, as highlighted in [25]. The pseudocode for the Q-Learning control scheme is given in listing 2.2.

Algorithm 2.2 Q-learning algorithm

```
Initialise Q(s, a) arbitrarily
loop for each episode
    Initialise s
    loop for each step in episode
        Choose a from s using policy derived from Q (ε-greedy)
        Take action a, observe r, s'
        Q(s, a) ← Q(s, a) + α[r + γ max_a' Q(s', a') − Q(s, a)]
        s ← s'
    end loop
end loop
```

2.3 Partially Observable Markov Decision Process

In many real world problems, the MDP model does not accurately reflect the agents capabilities with respect to sensing its environment. Often, its perception is limited and/or unreliable, thus conferring the *partial observability* property of the problem. This is also true for the Rescue Simulation problem, where agents are only able to sense the properties of entities in their immediate vicinity.

Incorporating partial observability into the MDP model yields the Partially Observable Markov Decision Process (POMDP). As described in [26], in addition to the state space $\mathcal{S}$, action set $\mathcal{A}(s)$, transition probability $\mathcal{P}^a_{ss'}$ and reward model $\mathcal{R}^a_{ss'}$, the POMDP also includes

- Ω , a finite set of *observations* that the agent can perceive about the environment;
- $O : \mathcal{S} \times \mathcal{A} \to \Pi(\Omega)$, an *observation function* giving the probability distribution over possible observations, for each action and resulting states.

In this new context, the agent needs to keep a *belief state* reflecting its previous experience. This is usually defined as a probability distribution over the state space. This belief is updated at each time step based on a and o using the observation function and transition probabilities:

$$b'(s') = \frac{O(s', a', o) \sum_{s \in \mathcal{S}} \mathcal{P}^a_{ss'} b(s)}{Pr(o|a, b)} \tag{2.17}$$

where $Pr(o|a, b)$ is a normalising factor, independent of s. As such, solving a POMDP is equivalent to finding an optimal policy over the continuous space "belief MDP". However, as shown in [4], finite-horizon POMDPs are PSPACE-complete and existing exact planning algorithms, such as the Witness algorithm described in [26], quickly become intractable even for modest size problems.

2.4 Bayesian Game Approximation

As mentioned in [24], while approximate solution algorithms for POMDPs can provide good results with reasonable computational costs, they generalise poorly to decentralised 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. The method proposed in [24] 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 generalisation for MDPs in multi-agent scenarios. Following the conventions laid in [24], 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);
- $T : S \times A \to S$ represents the transition function;
- $R : S \times A \to \mathcal{R}$ is the reward function;
- $O : S \times A \times Z \to \mathcal{R}$ 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 [24] 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 $u_i(a_i, a_{-i}, (\theta_i, \theta_{-i}))$.

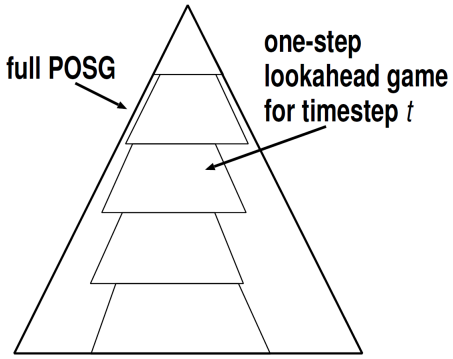

Figure 2.1: High-level representation of the Bayesian Game Approximation algorithm. Image source: [24].

The high-level representation of the Bayesian Game Approximation (BaGA) algorithm is reflected in figure 2.1. The POSG, represented by the outer triangle is approximated by a succession of smaller games at each timestep of the simulation, along a certain subpath of the tree. As described in [24], each subpath of the tree corresponds to a specific set of observation and action histories up to time t for all agents. If the specific path which actually occurred is known by all the agents, the problem becomes fully observable and the payoffs associated with each joint action at time t are known with certainty. This, in turn, allows the utility considered in the Bayesian games to be conditioned on specific type profiles. Therefore, each path in the POSG up to time t (i.e. observation and action histories) is represented as a specific type profile θ^t. Furthermore, in order for the approximation to be viable, the agents must hold a common prior over the type profile space Θ. This can be achieved if a common knowledge of the starting conditions of the original POSG is assumed to be available to all agents. As such, the algorithm can iteratively find Θ^{t+1} and $p(\Theta^{t+1})$ using information from $\Theta^t, p(\Theta^t), A, T, Z$ and O. Furthermore, the solution to the Bayesian Game, σ describes the optimal next-step policy of all agents, this observation also being used for updating the type profile space. Lastly, an important remark made in [24] regarding the utility is that it must reflect not only the immediate benefit of a certain action, but rather the long-term expected return. In other domains such as MDPs or POMDPs, these future values are computed by backing up action values from the final time step T down to the current time step t. However, in this case this procedure would involve just as much workload as solving the entire POSG, and as such, any advantages of the Bayesian approximation would be lost.

Algorithm 2.3 BaGA: PolicyConstructionAndExecution

procedure PolicyConstructionAndExecution($I, \Theta^0, A, p(\Theta^0), Z, S, T, R, O$)
  Output: $r, s^t, \sigma^t, \forall t$
  $h_i \leftarrow \emptyset, \forall i \in I$
  $r \leftarrow 0$
  $initializeState(s^0)$
  for $t \leftarrow 0$ **to** t_{max} **do**
    for $i \in I$ **do**
      $setRandSeed(rs^t)$
      $\sigma^t, \Theta^{t+1}, p(\Theta^{t+1}) \leftarrow$ BayesianGame$(I, I, \Theta^t, A, p(\Theta^t), Z, S, T, R, O, rs^t)$
      $h_i \leftarrow h_i \cup z_i^t \cup a_i^{t-1}$
      $\theta_i^t \leftarrow$ matchToType(h_i, Θ_i^t)
      $a_i^t \leftarrow \sigma_i^t(\theta_i^t)$
    end for
    $s^{t+1} \leftarrow T(s^t, a_1^t, ..., a_n^t)$
    $r \leftarrow r + R(s^t, a_1^t, ..., a_n^t)$
  end for
end procedure

The pseudocode of the BaGA approach, as described in [24], is presented in listings 2.3 - 2.5.

2.5 Distributed Constraint Optimisation and Max-Sum algorithm

Parting away from the DTP methods, one of the other approaches considered concerns the formulation of the behaviour as a Distributed Constraint Optimization Problem (DCOP). An example for this is introduced in [5], where the authors use this formulation to solve the issue of "coalition formation with spatial and temporal constraints" (CFST). 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 characterising 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 [5], 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.

Algorithm 2.4 BaGA: BayesianGame

procedure BAYESIANGAME($I, \Theta, A, p(\Theta), Z, S, T, R, O, randSeed$)
  Output: $\sigma, \Theta', p(\Theta')$
  setSeed($randSeed$)
  for $a \in A, \theta \in \Theta$ **do**
    $u(a, \theta) \leftarrow qmdpValue(a, beliefState(\theta))$
  end for
  $\sigma \leftarrow findPolicies(I, \Theta, A, p(\Theta), u)$
  $\Theta' \leftarrow \emptyset$
  $\Theta'_i \leftarrow \emptyset, \forall i \in I$
  for $\theta \in \Theta, z \in Z, a \in A$ **do**
    $\phi \leftarrow \theta \cup z \cup a$
    $p(\phi) \leftarrow p(z, a|\theta)p(\theta)$
    if $p(\phi) >$ pruningThreshold **then**
      $\theta' \leftarrow \phi$
      $p(\theta') \leftarrow p(\phi)$
      $\Theta' \leftarrow \Theta' \cup \theta'$
      $\Theta'_i \leftarrow \Theta'_i \cup \theta'_i, \forall i \in I$
    end if
  end for
end procedure

One of the algorithms commonly used for solving such a problem is the Max-Sum algorithm [27]. In order to employ this technique, the problem is formulated as a *factor graph*, a bipartite graph containing 2 types of nodes, representing variables (from $\mathcal{X}$) and functions (from $\mathcal{F}$). Following the conventions laid out in [27], under the Max-Sum algorithm, messages are passed between adjacent nodes of the factor graph. There are 2 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

$$\mu_{x \to f}(x) = \sum_{l \in ne(x) \setminus f} \mu_{f_l \to f}(x) \tag{2.18}$$

where $ne(x)$ denotes the neighbour nodes of x. The second kind of message, from function to variable nodes are defined as

$$\mu_{f \to x}(x) = \max_{x_1, ..., x_M} \left[\ln f(x, x_1, ..., x_M) + \sum_{m \in ne(f) \setminus x} \mu_{x_m \to f}(x_m) \right] \tag{2.19}$$

where $x_1, ..., x_M$ represent the argument variables of f, other than x and $ne(f)$ represents the neighbour nodes of f. As pointed out in [5], the messages to and from variable nodes are sets of values reflecting the total utility of the network for each possible as-

Algorithm 2.5 BaGA: findPolicies

procedure FINDPOLICIES($I, \Theta, A, p(\Theta), u$)
  Output: $\sigma_i, \forall i \in I$
  for $j \leftarrow 0$ **to** maxNumRestarts **do**
    $\pi_i \leftarrow random, \forall i \in I$
    while !$converged(\pi)$ **do**
      for $i \in I$ **do**
        $\pi_i \leftarrow \arg\max \left[\sum_{\theta \in \Theta} p(\theta) \cdot u([\pi_i(\theta_i), \pi_{-i}(\theta_{-i})], \theta) \right]$
      end for
    end while
    if bestSolution **then**
      $\sigma_i \leftarrow \pi_i, \forall i \in I$
    end if
  end for
end procedure

signment 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):

$$x_i = \arg\max_x \left[\sum_{s \in ne(x)} \mu_{f_s \to x}(x) \right] \tag{2.20}$$

The algorithm is guaranteed to converge to the global optimal solution for cycle-free factor graphs. However, as suggested by [19] [23], the algorithm generates good approximate solutions also for cyclic graphs. The Max-Sum algorithm developed from the sum-product algorithm, initially conceived for inference in graphical models. However, the dynamic nature of the RCR simulation can yield frequent changes in the structure of the factor graph. The Max-Sum algorithm, in its standard form, cannot adapt to these changes and the entire procedure of message passing must be run again entirely. These issues are addressed in [5] with the Fast Max-Sum algorithm, by adapting certain features of the standard Max-Sum, such as introducing new message types to accommodate disruptions in the graph or by restricting the domains of variables to reduce communication and computation overhead.

2.6 Gaussian Mixture Model

Of the non-DTP techniques employed in the agents' behaviour, probably the most peculiar is the use of Gaussian Mixture Models (GMM). Although more details on how this method is used in the agents' behaviour will be given in section 3.4.3, the general

idea behind the usage of GMM here is to help ambulance agents to keep an estimate on the most probable victim locations, thus optimising exploration.

More commonly employed in Machine Learning and Pattern Recognition, GMM allows a reasonable modelling of an unknown probability distribution. As described in [27], the probability distribution generated by a GMM is

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_\mathbf{j}, \mathbf{\Sigma_j}) \tag{2.21}$$

where K is the total number of components, π_k represent the mixing coefficients reflecting the weight of each component, with $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$, and $\mu_\mathbf{j}$ and $\mathbf{\Sigma_j}$ are the mean and covariance of each Gaussian component, respectively.

Usually, the GMM parameters are computed using the Expectation Maximization algorithm, based on a set of samples drawn from the unknown distribution. This is an iterative general optimisation algorithm, whose goal in particular for the GMM is to maximise the likelihood of the sample data points with respect to the parameters, consisting of mixing coefficients, means and covariances of each component.

An outline of the algorithm, as presented in [27], is given below:

1. Initialise the means μ_k, covariances $\sum_k$ and mixing coefficients π_k, and evaluate the initial value of the likelihood.

2. **E step.** Evaluate the responsibilities using the current parameter values. The weighting factor for data point x_n is given by the posterior probability $\gamma(z_{nk})$ that component k^{th} was responsible for generating data point x_n.

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

3. **M step.** Re-estimate the parameters using the current responsibilities

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

4. Evaluate the log likelihood

$$\ln p(X|\mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Chapter 3

Approach

3.1 Domain challenges

The Decision Theoretic Planning (DTP) paradigm offers a large array of algorithms and approaches to suit a broad range of problems, addressing particularities such as observability or number of agents. However, as problem specifications become closer and closer to real-world applications, the computational costs for approaches accommodating their particularities become prohibitive. Thus, in such cases, optimal trade-offs must be found between the accuracy of the models employed and the feasibility of the algorithms and techniques [33].

As previously mentioned in section 1.1, 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 by a similar straight-forward approach is done in [28].

Apart from the challenges imposed by the size of the considered domain, agents have a further time constraint, being required to submit their commands to the simulator within a limited time interval (typically between 1000 ms and 1300 ms) at each time step of the simulation. This further restricts the range of algorithms which can be considered for solving the problem.

Reducing the size of the problem, while still retaining sufficient information for efficient behaviour may prove too difficult, if not impossible, using a single model. As such, the solution described in this thesis involves using two separate approaches, organised

hierarchically, for micro-level and macro-level behaviour, respectively.

3.2 Overview

The above mentioned challenges imposed by the problem help in narrowing down and focusing the search for a suitable approach. This section presents some key observations and assumptions which motivate the choices made for the agents' behaviour. Regardless of the chosen methods, all DTP models require at the very least a state space, and action set and a reward model. These aspects will be discussed in the following paragraphs.

One important observation regarding the particularities of the problem is that the actions of the agents have a more pronounced and immediate effect in the close proximity of the agent and much less so in the parts of the map further away. Thus, information about the world that has little influence on the behaviour of the agent (e.g. the state of the buildings much further away) can be omitted from the model. This allows a concise enough domain to be defined for the micro-level behaviour. However, this model must be sufficiently general to be effective regardless of the agents' position along the map. As such, information to be included in the state space cannot be tied to particular buildings or other elements of the map, but rather generic and relative to the agent. This assumption seems reasonable since it also allows the agent to have similar behaviour in similar contexts.

A consequence which arises from the above mentioned generality of the state space is the fact that actions need to be defined in a similar fashion. Since the commands sent to the simulator must target specific entities of the map, and since these entities are not specified in the state space, the actions with which the algorithm operates need to have a higher "semantic" content, which then gets translated into individual commands.

Also because of the generic state space, the reward model definition cannot be straight forwardly inferred from the scoring function which needs to be optimised. Thus, a custom reward model must be defined as well, reflecting desirable behaviour in the context of the chosen state space and action set.

Although the agents have limited sensing capabilities with respect to map environment, considering partial observability, at least in the micro-level behaviour, can prove too demanding computationally-wise. A popular choice for approaching problems with partial observability in the multi-agent setting is the Dec-POMDP framework. While finding optimal solutions or bounded approximations is proven to be NEXP-Complete [29], various heuristic algorithms, such as the GMAA* family [31] [32] yield good results in typical benchmark problems. However, these typical benchmark problems are significantly different than the considered Rescue Agent Simulation problem, often in-

volving only 2 agents and just a few states and actions. Moreover, the horizons for which the computation times remain manageable are much more restricted than the typical simulation scenarios of 250-300 time steps. As such, given the short time frame of 1000-1300ms available for computation at each time step of the simulation and considering the amount of information needed to be incorporated in the definitions of the state space and action set, partial observability will not be considered for the micro-level behaviour.

3.3 Fire brigade behaviour

The following behaviours 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" approach, all macro-level behaviours 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 centre 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. As the simulation progresses and conditions change, these assignments may change over time.

3.3.1 Micro level approach

The micro level behaviour of the fire brigade allows it to make rational decisions, given the orders from the higher level macro controller, as well as local circumstances, such as the amount of water available or the state of other buildings. As previously mentioned, the state space needs to be general enough not to depend on particularities of the surroundings of the agent or on the map, and needs to include sufficient relevant local

information based on the observable domain of the agent. For all the studied approaches, with the exception of "Support requests", the following model was used for the micro level behaviour.

State space

The state space consists of a Carthesian product of the domains of the following variables:

Assigned building boolean, indicates whether a building has been assigned from the macro level controller;

Cluster size integer, indicates the total number buildings in the assigned cluster. If there is no assigned cluster, this has a fixed value (-1);

Other agents integer, indicates the total number of other agents assigned to the same cluster. If there is no assigned cluster, this has a fixed value (-1);

Buildings to check boolean, indicates whether there are buildings that need checking for fires (see the "Check area" action in the following section);

Other fire boolean, indicates whether there is a burning building in the reachable vicinity of the agent, other than the assigned building, if such an assignment exists;

Water volume integer, indicates the amount of water carried relative to the total capacity. This value has been discretised in 10 steps to keep the dimension of the state space manageable.

Action set

The actions associated with the above state space are:

Explore The behaviour associated with this action causes the agent to explore (patrol) a region of the map assigned to it. Details regarding this behaviour are given in section 3.3.1.

Extinguish assigned Taking this action causes the agent to move towards the assigned building until it is in range and then extinguishing it;

Extinguish other Taking this action causes the agent to choose a "best fire" (according to a heuristic) from among the observable burning buildings and extinguishing it (after moving to get close enough, if necessary);

Check area When taking this action, a list is firstly created, containing all buildings in close proximity of the last extinguished building. Subsequently, the agent moves towards them and removes them from the list as they are observed. The action ends when all buildings in the list have been removed.

Refill Taking this action causes the agent to go to the nearest available refill point and refill. If this refill point is a hydrant (where a single agent can refill at a time), a message will also be sent to peer agents, informing that the respective hydrant is occupied. This is required since only one agent can refill at a hydrant at any given time. When the refilling is complete another message is sent to the peers, pointing out that the hydrant has been released.

Micro level decision process

Given the previous state space and action definitions, analytically creating a suitable transition model for planning would prove unfeasible. This is mainly due to the complex spreading of fire and particularities of the maps. One option would be to learn the transition probabilities from experience. Better yet, using a Temporal Difference control algorithm, such as Sarsa or Q-Learning allows the agents to learn the dynamics of the environment at runtime, while still enabling (reasonably) efficient behaviour. This also guides the exploration of the state space towards the most promising areas.

Due to previously mentioned constraints of the micro level behaviour, explicit multi-agent cooperation is not considered at micro level. Since the number of peer agents present in a given cluster is encoded in the state description, one way to enable an implicit cooperation is to consider stochastic action selection. For example, if 3 agents are engaged in fighting fires within a cluster and all take a certain action with probability 0.33, only one of them will end up taking that action, despite all having the same policy.

Micro level: Exploration

An important aspect of the agents' micro level behaviour is the exploration. Following the commonly employed practice among participating teams, the map is partitioned using K-Means into a number of sectors prior to the beginning of the simulation. Subsequently, agents get assigned to clusters based on their proximity and such that all clusters have an equal number of assigned agents. During the simulation each agent keeps track of the time step it last observed a certain building in within its assigned cluster. As such, whenever the agent takes the "Explore" action, the earliest visited building is selected as a target and the agent begins moving towards it. This behaviour

is also common to the ambulance team's explore action in the "Support requests" approach.

3.3.2 Macro level behaviour: Heuristic pairing

The heuristic pairing macro behaviour 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 favours extinguishing the outer buildings and containing the fire to a manageable size, ultimately extinguishing it completely.

3.3.3 Macro level behaviour: 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.

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.

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.

The quality of the assignments is given by an **utility function** tying the agent, its type and the action. Four utility functions have been considered. The first function, seen as a baseline, only takes into consideration the distance between the agent and each fire. The second function takes into account the fieriness of the buildings, lower being considered better, with the ties being broke by the distance between the agent and the building. Finally, the last two utility functions are inspired by similar measures used in different approaches by two of the 2013 finalist teams. The first of those, inspired by team LTI [10], takes into account the fireriness of the building and the distance to the agent, as well as the ground area and the number of unburning adjacent buildings. The second utility measure, inspired by team MRL [16], takes into account the the fireiness of the building, its area, the distance to the agent, as well as the temperature. In the above described utility functions the building (or fire) mentioned is the one given by the action code.

To summarise, the problem model onto which the BaGA approach is employed is described by:

- $I = \{1, 2, ...n\}$ - the set of indices of agents engaged in the cluster;
- $S =< F_1 \times F_2 \times ... \times F_m >$ - the state space representing the joint fieriness of the buildings in the cluster;
- $Z_i =< F_1 \times F_2 \times ... \times F_m >$ - the observation space of agent i representing the fieriness of the buildings observed by agent i;
- $A_i = \{B_1, B_2, ...B_m\}$ - the action set of agent i, containing the indices of the buildings in the assigned cluster;
- $\Theta_i^t =< z_i^0, a_i^0, z_i^1, a_i^1, ...z_i^t, a_i^t >$ - the type of agent i at time t, consisting of its history of observations and actions taken by agent i from time 0 until time t.

3.3.4 Macro level behaviour: DCOP pairing

Following the general description of [5] 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 behaviour 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 described in [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 2.5. A benchmark framework, introduced in [6], provides implementations for several state of the art algorithms aimed at solving DCOP formulations of certain tasks in the RoboCup Rescue simulation. Of these, the Max-Sum algorithm was chosen due to its superior performance for this domain, compared to other algorithms.

3.3.5 Support requests

This approach assumes a very simple macro level behaviour, 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 behaviour is needed, particularly to take into account the fieriness of the buildings in the immediate vicinity of the agent. As such, the state space of the micro level model for this approach contains the following information:

Distance to nearest low-burning building integer, indicates the discretised distance to the nearest low burning building (fieriness 1). If such a building does not exist, this variable has a fixed value (-1);

Distance to nearest mid-burning building integer, indicates the discretised distance to the nearest mid burning building (fieriness 2). If such a building does not exist, this variable has a fixed value (-1);

Distance to nearest high-burning building integer, indicates the discretised distance to the nearest high burning building (fieriness 3). If such a building does not exist, this variable has a fixed value (-1);

Distance to nearest unexplored position integer, indicates the discretised distance to the nearest unexplored position;

Distance to nearest refuge integer, indicates the discretised distance to the nearest refuge or hydrant;

Other agents integer, indicates the total number of other peer agents present in the close vicinity;

Water volume integer, indicates the amount of water carried relative to the total capacity. This value has been discretised in 10 steps to keep the dimension of the state space manageable.

The action set for this approach is adapted accordingly,

Explore The behaviour associated with this action causes the agent to explore (patrol) a region of the map assigned to it. Details regarding this behaviour are given in section 3.3.1;

Extinguish low-burning Taking this action causes the agent to move towards the closest low burning building until it is in range and then extinguishing it;

Extinguish mid-burning Taking this action causes the agent to move towards the closest mid burning building until it is in range and then extinguishing it;

Extinguish high-burning Taking this action causes the agent to move towards the closest high burning building until it is in range and then extinguishing it;

Refill Taking this action causes the agent to go to the nearest available refill point and refill. If this refill point is a hydrant (where a single agent can refill at a time), a message will also be sent to peer agents, informing that the respective hydrant is occupied.

3.4 Ambulance behaviour

Compared to the fire brigades, where agent coordination was crucial in containing larger clusters of burning buildings and limiting the expansion of fires, the cooperation aspect of the ambulance teams is somewhat less important. This flows from the fact that a single ambulance can carry a single victim at a time and the only benefit of multiple ambulances rescuing a buried victim is a reduced completion time of the unbury task. This has a limited utility, mostly in cases where a fire expansion threatens the building containing a victim and thus a faster evacuation would be desirable. However, since the number of buried civilians is usually large enough not to allow a complete evacuation before the end of the simulation, the extra ambulances aiding a single victim would be put to better use in discovering and rescuing other victims.

Even though ambulance coordination does not impact greatly the actual victim rescuing and transportation, it does affect the exploration of the map. As such, the macro level behaviour of the ambulance teams is focused on victim discovery.

3.4.1 Micro level approach

Similarly to the fire brigade micro behaviour, the ambulance micro level model follows a MDP structure and deals with the local, tactical decisions. Since the ambulance behaviour revolves around buried and/or injured victims, the state space encodes information regarding the number and distance to the observed victims and the carying availability.

State space

The state space consists of a Carthesian product of the domains of the following variables:

Distance to refuge integer, discretised distance to the nearest refuge;

Distance to victim integer, discretised distance to the nearest alive victim;

Distance to unexplored vertex integer, discretised distance to the nearest unexplored position;

Victim on board boolean, whether there is a victim on board;

Victims in range integer, the number of victims in the immediate vicinity.

Action set

The actions associated with the above state space are:

Explore The behaviour associated with this action causes the agent to explore (patrol) a region of the map assigned to it. Details about this behaviour are given in subsection 3.3.1.

Pick up Taking this action causes the agent to move towards the nearest victim and attempt to load it. If it is buried, the victim is first rescued;

Drop at refuge Taking this action causes the agent to move towards the nearest refuge and, once there, to unload the victim;

3.4.2 Macro level behaviour: Support requests

One of the first approaches tested for the macro level behaviour of the ambulances is closely related to the "Support requests" behaviour of the fire brigades, described in section 3.3.5. As such, by default the ambulances explore their associated cluster (through the micro-level behaviour, as described in section 3.3.1), until a victim is reported by another agent (police force, fire brigade or already engaged ambulance team). An auction then takes place to determine the closest ambulance to the reported victim and the "winner" is moves then towards the victim (with the micro-level behaviour taking care of the actual rescuing process). Although this approach is relatively straight forward, it directly leverages the exploration done by all the agents on the map, while also not being constrained by a central agent which could form a bottle-neck.

3.4.3 Macro level behaviour: GMM-based

Another approach directed at improving the actual exploration behaviour of the ambulances is founded on the idea of maintaining a distribution of the probabilities of victims' positions. As such, the victims can be seen as samples drawn from an unknown "threat distribution" over the bidimensional space of the city. The area onto which each ambulance team can efficiently search and rescue victims can also be modelled as a "help distribution". As such, the problem of the ambulance's macro-level behaviour translates into the problem of efficiently fitting the "help distributions" to the "threat distribution". The method of choice for this task is using the Expectation Maximization algorithm for Gaussian Mixture Models.

Naturally, the threat distribution is unknown and the real samples are also not given (actual victims need to be discovered). Since the competition rules do not impose any

restrictions on the positioning of the victims across the buildings of the city, the "threat distribution" can be assumed to uniform. From this assumed distribution a number of virtual samples are drawn, describing a potential placement of the actual victims. Acting upon this information, the "help distributions" are fitted to approximate the assumed samples, thus driving exploration across the city. As ambulance teams patrol the buildings, they continually report their position to the central agent. Consequently, this agent deletes any possible virtual samples which are not in accordance with the observed circumstances of the ambulance team (i.e. virtual samples in the close proximity of the respective ambulance team which do not correspond to real victims). As such, the virtual samples describe the real threat distribution increasingly more accurate as the simulation progresses. After a certain number of virtual samples has been deleted (10 per cent of the total number of virtual samples) the fitting is run again so as to reflect the updates in the knowledge with respect to the victim placement. Consequently, this drives the exploration of the agents towards the less explored regions of the map (which, naturally, have more virtual samples).

Because of the particularities of this method, the explore action in the micro level behaviour is slightly modified. Thus, the estimation of the timestep when each of the buildings within the cluster is visited, mentioned in section 3.3.1, is refreshed whenever the GMM is re-estimated, such that it reflects the probability distribution of the assigned component. Therefore, whenever an explore action is chosen at the micro level behaviour, the building with the highest probability is chosen, instead of the earliest visited one.

An example of the evolution of the GMM is shown in figure 3.1. At the beginning of the simulation, the virtual samples are uniformly distributed across the city and the corresponding Gaussian components are somewhat broad. As the ambulance teams explore the map and the virtual samples are eliminated, the components shift their position reflecting the less explored regions, thus favouring these more promising areas of the map.

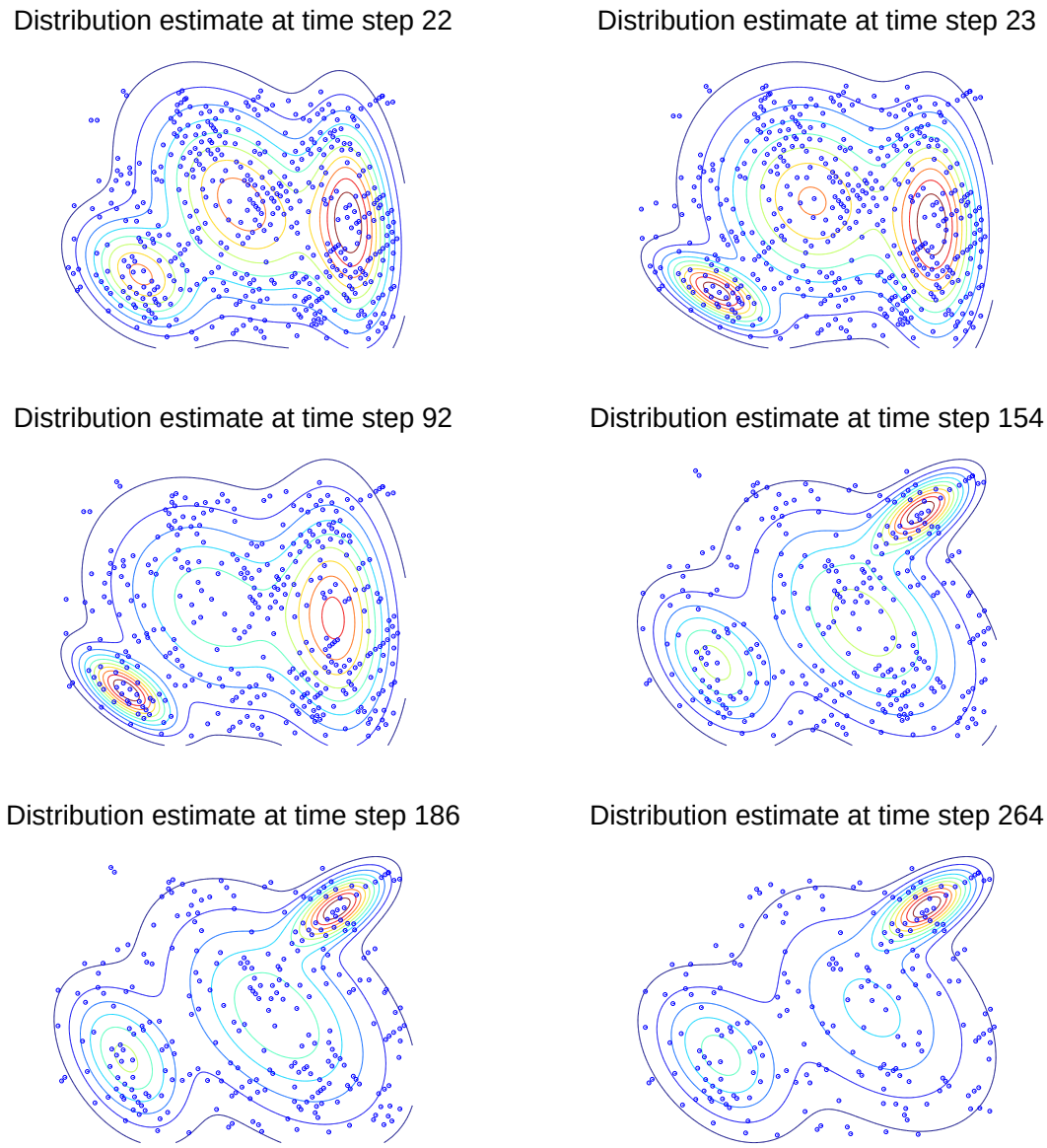

Figure 3.1: Evolution of the GMM estimation of victim position probabilities.

Chapter 4

Experiments

4.1 Fire brigade experimental setup

The maps used in the experiments are part of the official maps of the 2013 RoboCup Competition, namely **Kobe1**[1], **Kobe2**[1], **Paris1**[1] and **Paris2**[1] . Differences between the versions of the same city maps concern properties of the fires, such as the position and fierceness of the initial fires and the likelihood of further ignitions, the distribution of agents, both civilian and non-civilian and the availability and position of special buildings, such as refuges and centre agents. In addition to the original maps, I have also created custom, more difficult, versions. 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, I 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. Finally, the **Kobe2-f20**[2] and **Paris2-f20**[2] maps have the same configurations as their respective originals with the difference that the agents are not allowed to act ("frozen") for the first 20 time steps. Under this restriction, the initial fires grow larger and are harder to extinguish without efficient agent collaboration. The initial configuration of the maps, showing the map topology, initial fires and agent positions are shown in figures 4.1 - 4.5.

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. As such, the measured performance of the fire brigades and ambulance teams depends heavily on the performance of the

[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 .

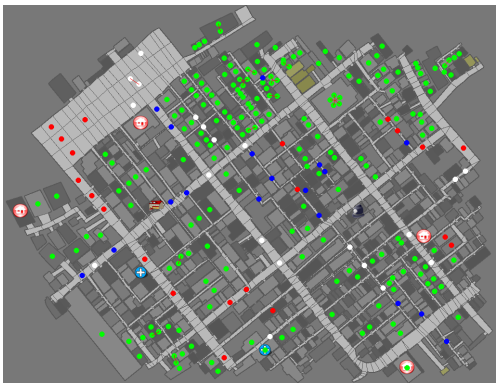

Figure 4.1: Initial configuration for the Kobe1 map.

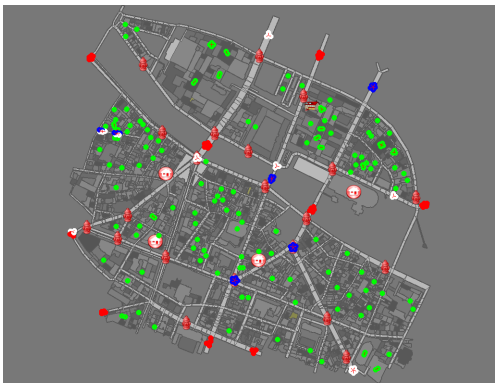

Figure 4.2: Initial configuration for the Paris1 map.

Figure 4.3: Initial configuration for the Kobe2 map.

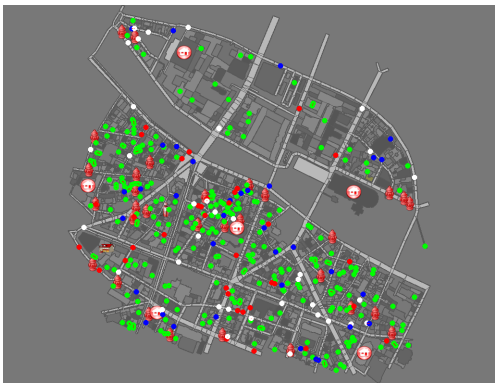

Figure 4.4: Initial configuration for the Paris2 map.

police teams. However, since the behaviour of the police agents was not the object of this thesis, and because the goal of the experiments is to assess the performance of the individual approaches discussed in Chapter 3, 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 thesis was agent behaviour and not the development of high-performance, fail-safe communication protocols and since the proper execution of (part of) the previously described approaches relies on communication, all the simulations have been run using a single high bandwidth (400Kb) reliable (no dropped messages or noise) radio channel.

The performance measure used in the following experiments is, unless otherwise noted, the scoring function used in the competition scenarios. While this function takes into account the state of the civilians' health and the buildings throughout the city, only the building component of the scoring function will be reported, as it is more relevant for

Figure 4.5: Initial configuration for the Paris-mini-hard map.

the performance of the fire brigades. The building score equation is

$$score_t = \frac{\sum_{f=1}^{8} n_t^f * w_f}{N} \tag{4.1}$$

where $score_t$ represents the overall score at time step t, n_t^f represents the number of buildings of fieriness f at time t, w_f is a parameter reflecting the score of a building at a given fieriness f and N represents the total number of buildings in the city. The actual numeric values of the parameters w_f are given in table 4.1.

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

4.1.1 Fire brigade macro level behaviour

The results obtained for the macro level approaches of the fire brigades are shown in figures 4.6 - 4.14. In several cases, the performance of the "Sample agents" descends to very low values (0.2 - 0.4), while the other approaches' performance remains much higher. In such cases the axis of the figure were restricted to better highlight the score differences of the approaches other than "Sample agents".

Fieriness (f)	Description	w_f
0	Intact	1
1	Low burning	0.66
2	Moderately burning	0.33
3	Fiercely burning	0
4	Water damage	0.9
5	Extinguished, minor damage	0.75
6	Extinguished, moderate damage	0.5
7	Extinguished, severe damage	0.25
8	Burnt out	0

Table 4.1: Fieriness score parameters.

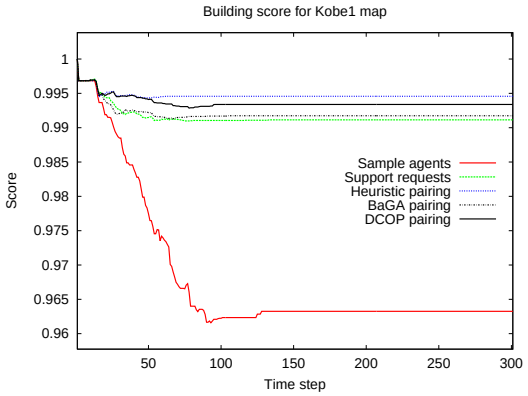

Figure 4.6: Building score associated with the performance of various macro level approaches on the Kobe1 map.

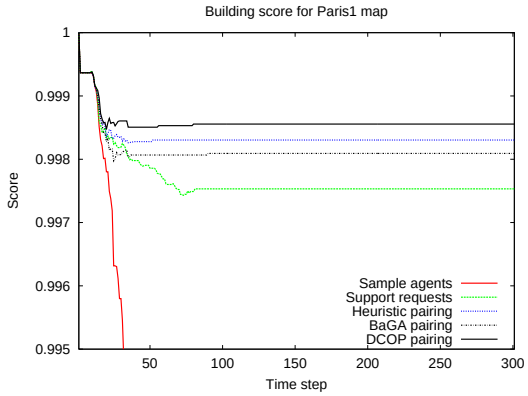

Figure 4.7: Building score associated with the performance of various macro level approaches on the Paris1 map.

As expected, the weakest performance on all the maps is yielded by the "Sample agents". However, they do manage an acceptable result on the Kobe1 map, suggesting its low difficulty. Another interesting result regarding the sample agents is found towards the end of the Kobe-hard simulation (figure 4.10), where they slightly outperform the "Support requests" approach. As indicated by the numerical values and shape of score function, at that stage the fires are out of control and the majority of the buildings in the city are burning. As such, the "Support requests" agents face a significant overhead of requests and offers, with the agents not being able to react fast enough. In contrast, the sample agents have a fixed reactive behaviour and no communication. This simpler and more straight forward approach proves to be marginally more efficient in this context.

The lower difficulty of Kobe1 is also suggested by the very close performance of all other approaches, with a slight advantage of "Heuristic pairing" approach. This approach yields better performance on the other versions of the Kobe map as well. Since this is not the case for the Paris maps, the result suggests that the assumption of the heuristic

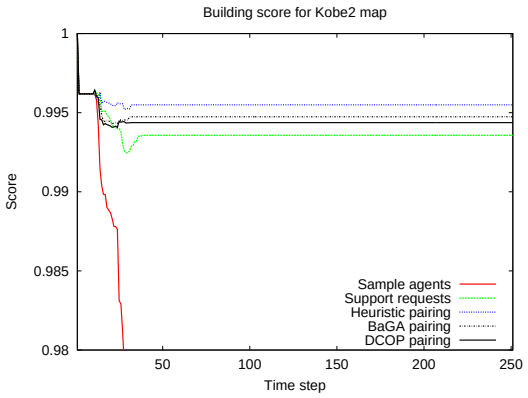

Figure 4.8: Building score associated with the performance of various macro level approaches on the Kobe2 map.

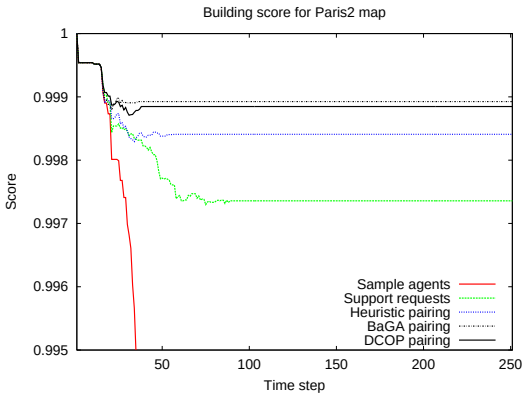

Figure 4.9: Building score associated with the performance of various macro level approaches on the Paris2 map.

pairing method holds better for the particularities of the Kobe map. More specifically, recently reported fires tend to correspond to the outer front of the burning building cluster as the buildings are generally smaller (better granularity with respect to the shape of the outer front) and the road network used by the agents is a (mostly) uniform grid. As such, regardless of their destination, the agents are more likely to traverse the city uniformly and encounter actual fronts of the fires, as opposed to the Paris-hard map, where the optimal routes tend to make use of large avenues, making the agents less likely to encounter fires in their early stages.

For the Paris maps, better performance is obtained by the "DCOP pairing" and "BaGA pairing" methods, as the buildings are somewhat larger and harder to extinguish. In such cases, the benefits of more elaborate coordination are more visible. This is also suggested by the performance of "Support requests" which is further apart from the other approaches than in the case of the Kobe maps.

Nonetheless, with the exception of "Sample agents", all the studied approaches manage to successfully contain and extinguish the fires on the Kobe1, Kobe2, Paris1 and Paris2 maps, yielding good overall scores.

An interesting result concerns the Kobe-hard, Paris-hard and Paris-mini-hard maps. Since the 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 for most approaches. 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 3 approaches yield better performance, being able to contain the

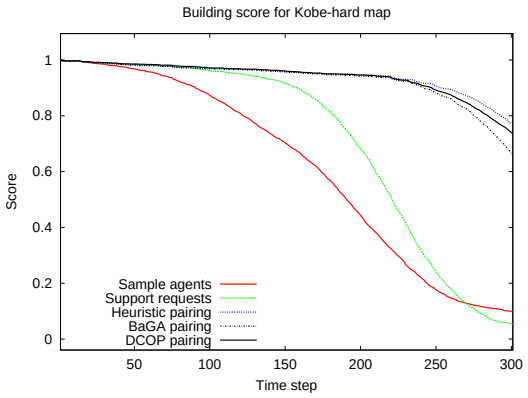

Figure 4.10: Building score associated with the performance of various macro level approaches on the Kobe-hard map.

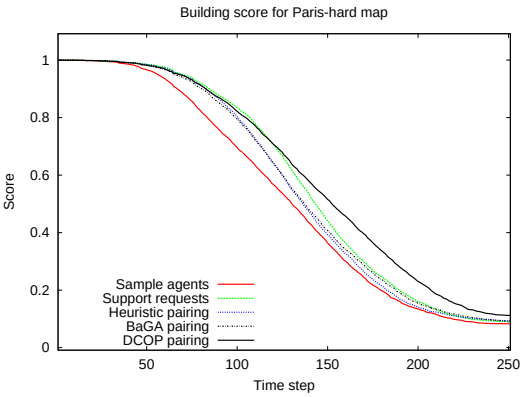

Figure 4.11: Building score associated with the performance of various macro level approaches on the Paris-hard map.

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 Kobe-hard this happens on average at around timestep 135. However, in order to reflect the performance of the "DCOP pairing" as accurately as possible, the runs used for plotting the results in figure 4.10 have an average time of running out of memory at 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.

Paris-hard represents a considerably larger map than Kobe-hard and the memory issues causing the DCOP pairing to stop working manifest themselves at a very early stage, on average at around timestep 28. Because of this, the performance associated with the DCOP pairing shown in figure 4.11 is only given by the micro-level behaviour of the agents (as the thread responsible for the pairing is not running anymore and thus no assignments are issued to the agents). As such, it is interesting that although the final score is the roughly the same for all studied approaches, the agents focusing on only the fires in their immediate observable surroundings yield the best performance during the intermediate steps of the simulation. Most probably this is due to the fact that fire assignments received from the macro-level behaviour cause the agents to move across the

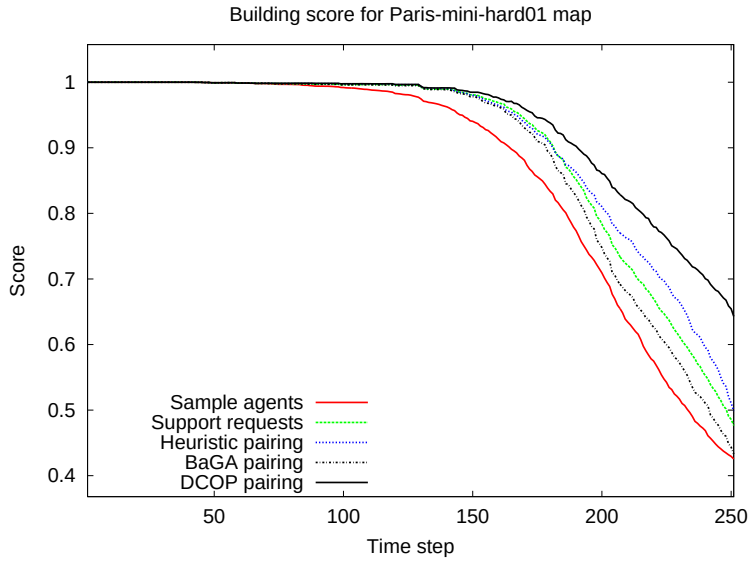

Figure 4.12: Building score associated with the performance of various macro level approaches on the Paris-mini-hard map.

map, effectively losing time while other fires grow beyond control, reducing the benefits of the coordinated approaches. This explanation is also supported by the fact that the "Support requests" approach has a similar performance to "DCOP pairing" until timestep 115 and still outperforms the others until the end of the simulation, which is never the case for any other map. In the "Support requests" approach, since the agents are engaged in fighting fires throughout the simulation, they do not respond to any support requests and thus also only focus on their immediate surroundings, effectively relying on their micro-level behaviour. The reasoning on why this happens for the Paris-hard map and not on the Kobe-hard map can be explained by the difference in size, with the travelling time of the agents having a greater impact on the Paris-hard map, as well as the nature of the buildings.

In order to still have an estimation for the DCOP pairing approach for a map other than Kobe-hard, the Paris-mini-hard map was created, containing only the northern part of the Paris-hard map, with adjusted parameters. In this case, the DCOP pairing method ran out of memory on average around time step 230 and, as such, the performance shown in figure 4.12 also reflects the macro-level behaviour. 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, as was the case for other Paris maps, 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.

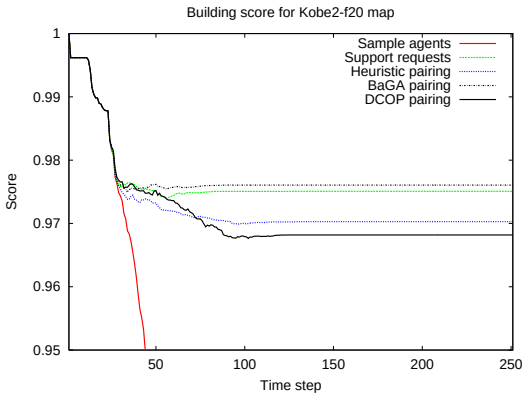

Figure 4.13: Building score associated with the performance of various macro level approaches on the Kobe2-f20 map.

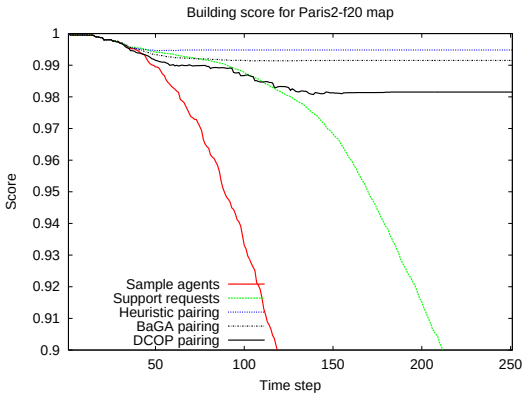

Figure 4.14: Building score associated with the performance of various macro level approaches on the Paris2-f20 map.

For the Kobe2-f20 and Paris-f20 maps, the "DCOP pairing" method also ran out of memory in the early stages of the simulation and the reported performance is only due to the micro-level behaviour. Since there is no higher level coordination between the agents, this would explain poor performance compared to the other methods on both maps. Conversely, having the more elaborate agent coordination, the "BaGA pairing" yields the best results on Kobe-f20 and second best on Paris-f20. Regarding the "Heuristic pairing" method, on the Kobe-f20 map one of the initial fires has central position and grows in the first 20 steps such that some of the agents are inside the the cluster of burning buildings. As such, the assumption of the "Heuristic pairing" regarding the correlation of the order of the reported fires and fire front does not hold as well, reflected in the relatively poorer performance on the Kobe-f20 map. However, this does not happen on the Paris-f20 map, as the fires, although larger, are discovered naturally through exploration.

As a general conclusion regarding the performance of the various macro 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. Taking into account the results on the harder maps, despite the performance associated with DCOP pairing being very close to that of the other approaches, sometimes even outperforming them, the very demanding memory requirements can prove to be a considerable disadvantage of the method, making it impractical in certain cases. Providing additional memory for this algorithm does not solve the problem, as for the very demanding scenarios the computation time required significantly exceeds the allotted time. A solution to this problem would be employing an any-time adaptation or a faster variant of the algorithm, such as Fast Max-Sum [5]. 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".

4.1.2 Fire brigade BaGA-pairing utility function experiment

In section 3.3.3 four utility functions have been introduced. The aim of this experiment is to determine the influence of functions over the general performance and to help choose the best one for use while comparing the BaGA pairing macro level behaviour with the others. As a reminder, the utility functions used were **closest** emphasising just the distance between the agents and the burning buildings in the cluster, **lowest-closest** taking into consideration first the fieriness of the buildings (with lower burning buildings being preferred) and then breaking ties with the distance and finally **LTI** and **MRL**, inspired by the approach of the LTI and MRL teams respectively, finalists in the 2013 competition.

The building score component associated with these utility functions for the considered maps are shown in figures 4.15 - 4.22.

Surprisingly, on most maps the simplest utility function considered, "closest", yields the best or second best performance. As all other utility functions include the metric used in the "closest" approach, namely the distance between the agent and the fire, this suggests that the other elements, such as the fieriness or the area of the burning buildings usually do not have significant benefits, even worsening the performance in certain cases. This is particularly true in the cases with larger clusters of burning buildings, such as Kobe2-f20 and Paris2-f20, where the more elaborate utility functions, "LTI" and "MRL", yield considerably worse performance than the other two. "LTI" tends to loose control of larger fires faster than other methods, suggested by the earlier beginning of the score descend in Paris2, Kobe-hard, Paris-hard and Paris2-f20. This may be due to the fact that the various metrics used are not normalised and just added together, which, for clusters with numerous buildings, may lead to inconsistencies. Another interesting result is the good performance of the "MRL" method on Kobe-hard. Although it outperforms all the other methods, the shape of the score evolution suggests a similar evolution to that of the "closest" approach, just being able to contain the smaller fires which appear continuously for slightly longer (until roughly time step 220 as opposed to 180). Lastly, a possible explanation for the good performance of the "closest" method is the fact that agents approaching the fire from different directions and aiming at the nearest building of the cluster end up extinguishing first the outer buildings, limiting its spread.

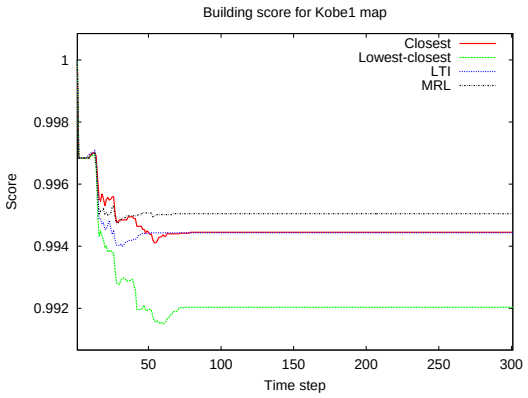

Figure 4.15: Building score associated with the performance of the BaGA-based fire brigade macro behaviour using different utility functions on the Kobe1 map.

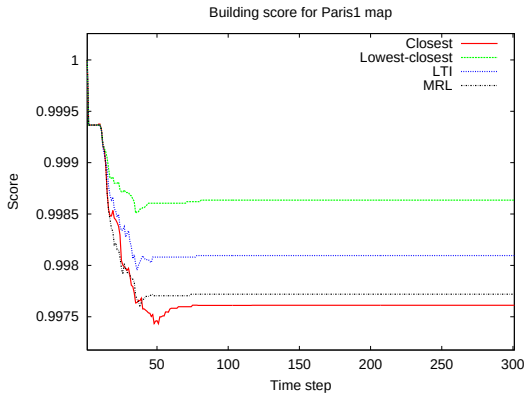

Figure 4.16: Building score associated with the performance of the BaGA-based fire brigade macro behaviour using different utility functions on the Paris1 map.

4.1.3 Fire brigade micro level behaviour

Since the micro level behaviour is not as sensitive to the overall particularities of the maps tested, only one of them were used for assessing its learning capabilities, namely Kobe1. In order to ensure a higher degree of consistency, the macro level behaviour used was the most straight forward of the approaches, "Heuristic pairing". The metric used for this experiment is the evolution of the final score over 10 consecutive runs (thus, at each run the agent benefiting from its past experience). Furthermore, these values are averaged over 10 series of runs in order to reduce the influence of the inherent random elements of the method. The results are reported in figures 4.23. Lastly, the values of the parameters used were:

- $\epsilon = 0.1$, the probability of a non-optimal action being taken (ϵ-greedy policy) ;
- $\alpha = 0.1$, the learning rate;
- $\gamma = 0.3$, the discount factor.

The agents obtain a good performance even within the first episode, which then rises and oscillates around the score of 0.992 - 0.995, very close to the score obtained with the handcrafted policy of 0.995. This oscillation is natural, as the agents continue learning by trying other actions (through the ϵ-greedy policy). The performance of Q-Learning and SARSA are very similar, with a slightly lower initial score for SARSA (as it optimises Q^{π} instead of Q^{*}, which is somewhat slower), but with a marginally higher score than Q-Learning afterwards. This sort of behaviour is also highlighted in [25], pointing out that on certain domains the online performance of Q-Learning may

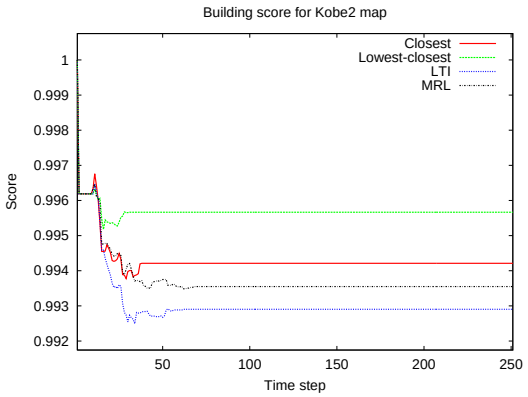

Figure 4.17: Building score associated with the performance of the BaGA-based fire brigade macro behaviour using different utility functions on the Kobe2 map.

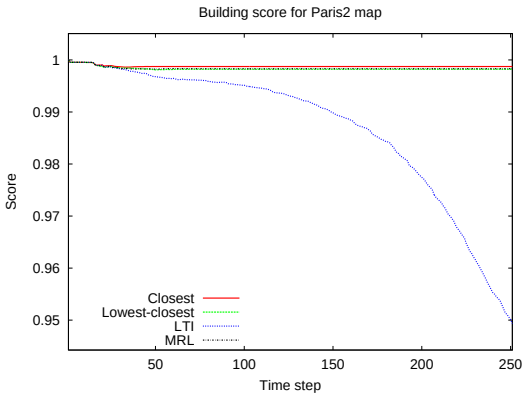

Figure 4.18: Building score associated with the performance of the BaGA-based fire brigade macro behaviour using different utility functions on the Paris2 map.

be lower than that of SARSA, despite a higher convergence time.

4.2 Ambulance Team macro behaviour

4.2.1 Experimental setup

In order to asses the performance of the ambulance teams' macro behaviour, a different setup has been considered due to the fact that fires can affect the civilians and other agents, but not the other way around. As such, the experiments run in this section use a modified version of the Kobe1 map where no fires occur and only ambulance teams are present. Furthermore, the number of agents was reduced to just 6 ambulances to better emphasise the performance of their behaviour. For the same reason, a total of 3 versions of the maps were considered, including 4, 8 and all the victims from the original configuration, respectively. Please note that in the latter configuration not all the civilians are buried and can move by themselves towards the refuges. However, these unburied agents are not taken into consideration in the evaluation. The names of the configurations previously described are **Kobe-civ4**, **Kobe-civ8** and **Kobe-civ-full**, respectively.

The metric used in these experiments is the time to victim discovery. As such, at each time step the total number of found victims is assessed. For each configuration, the results are averaged over 15 runs.

An observation is required with respect to the "Support requests" macro behaviour.

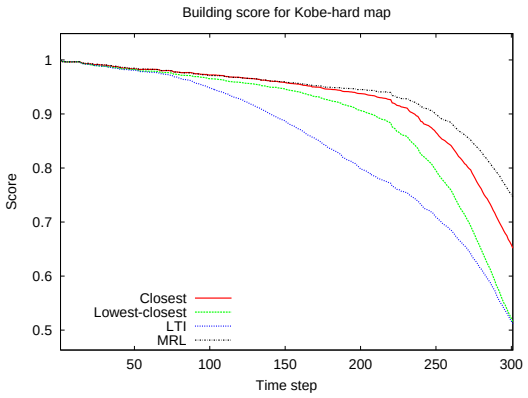

Figure 4.19: Building score associated with the performance of the BaGA-based fire brigade macro behaviour using different utility functions on the Kobe-hard map.

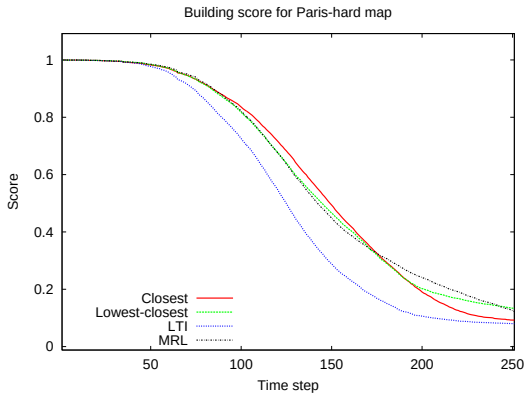

Figure 4.20: Building score associated with the performance of the BaGA-based fire brigade macro behaviour using different utility functions on the Paris-hard map.

Since no agents other than ambulance teams are present in the considered maps, the actual support requests will be very unlikely (issued only if an ambulance already carrying a victim sees another unattended buried victim while on its way to the refuge). As such, the performance of this approach better reflects the exploration of each individual agent through its assigned cluster of buildings.

The behaviours tested in this experiment are "Support requests" and "GMM-based".

4.2.2 Results and discussion

The results obtained for the setup previously described are shown in figures 4.24 - 4.26.

While the performance of the approaches is overall similar for both maps, the GMM-based approach yields better results in the later stages of the simulation (more pronounced in the case of Kobe-civ4). This is to be expected, as the GMM-based approach drives the exploration towards the more promising parts of the map, which become more focused as the simulation progresses. This behaviour becomes more obvious in the situations where finding the victims is harder, i.e. when there are fewer overall victims. Therefore, in the case of Kobe-civ-full the performance difference is not as obvious towards the end of the simulation, although an increase in the slope of performance associated with "GMM-based" approach is visible between timesteps 240 - 300. This suggests a better exploitation of the previously unexplored space.

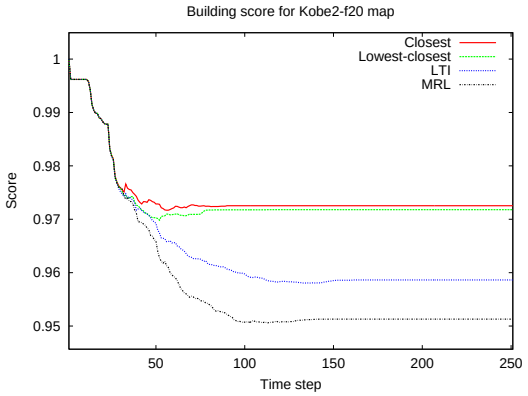

Figure 4.21: Building score associated with the performance of the BaGA-based fire brigade macro behaviour using different utility functions on the Kobe2-f20 map.

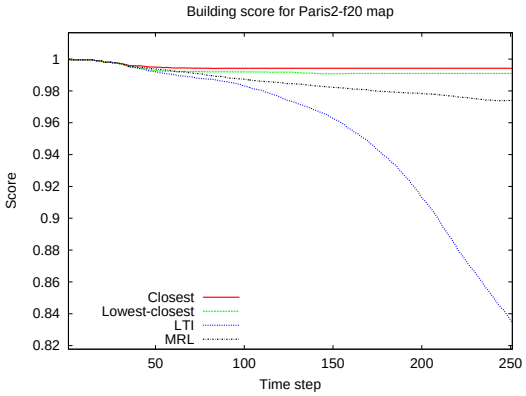

Figure 4.22: Building score associated with the performance of the BaGA-based fire brigade macro behaviour using different utility functions on the Paris2-f20 map.

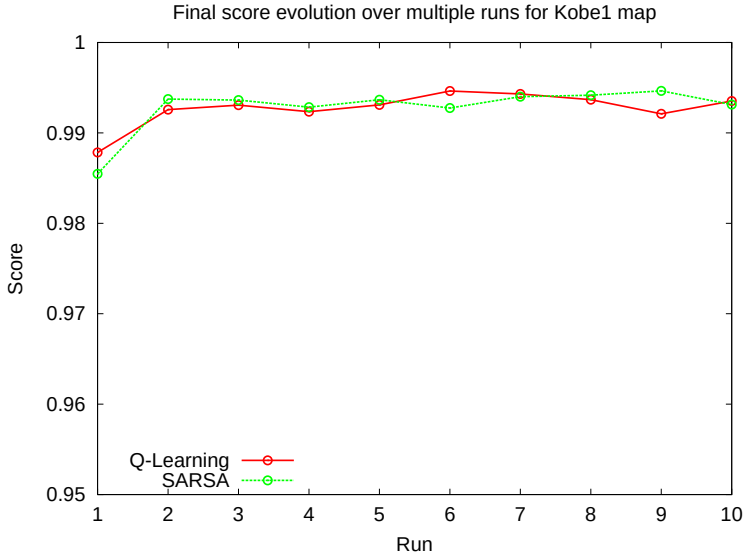

Figure 4.23: Mean of the final score evolution over 10 series of 10 runs each, reflecting the micro level behaviour learning on the Kobe1 map.

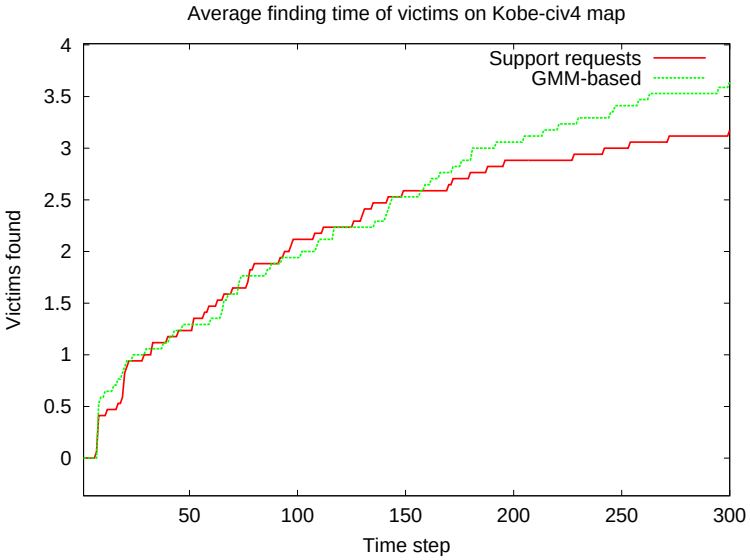

Figure 4.24: Average number of victims found over time for the Kobe-civ4 map.

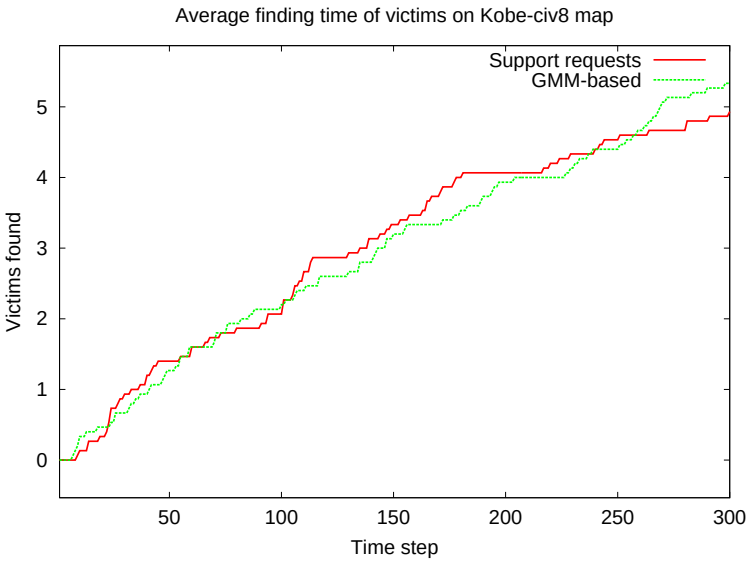

Figure 4.25: Average number of victims found over time for the Kobe-civ8 map.

Figure 4.26: Average number of victims found over time for the Kobe-civ-full map.

Chapter 5

Conclusions

In this thesis I have presented the theoretical foundation and evaluated a novel approach for modelling the behaviour of agents in the RoboCup Rescue Agent Simulation competition. The approach follows a hierarchical structure, with a macro-level behaviour focusing on the higher strategic decisions and agent collaboration and a micro-level behaviour managing the local, tactical behaviour, and rapidly responding to changes in the environment.

While micro level behaviour was restricted to a simpler MDP-based approach due to computation time limitations of the simulators, several methods have been investigated and tested for developing the macro level behaviour of fire brigades and ambulance teams. Following the DTP 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 maps tested. Since the quality of the solutions obtained by this algorithm depends on the utility function used, a separate experiment investigated the influence of 4 such functions. Another method, popular in the scholarly works studying the RoboCup Rescue domain, involved formulating the macro level behaviour 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 [5]. Another heuristic method was evaluated obtaining good results in circumstances where its core assumption held.

A fourth approach, favouring a more elaborate micro-level behaviour, yielded poorer results for the fire brigades, but proved suitable for the ambulance teams. For the victim finding task another approach proved slightly better, making use of a more elaborate model. This model, based on a GMM, keeps track of the most probable victim positions,

constantly updating them.

Lastly, the methods developed and evaluated in this thesis followed the rules and constraints of the Rescue Agents competition setting, making them suitable not only in benchmarks or artificial experiments, but also for competitions. As such, participating in the 2014 RoboCup in Brazil and the 2014 Iranian Open, on certain maps our team outperformed several competitors of the main competition and the highest total score in a special multi-agent challenge. Further details on these results are given in Appendix A

5.1 Future work

Although the design, implementation and evaluation of the approaches studied in this thesis provided great insight into the challenges and particularities of the RoboCup Rescue Agent Simulation, there are multiple potential directions for future research.

One decision with a potentially significant impact on the performance of the agents concerns the placement of the boundary between the micro and the macro level behaviour. Throughout this thesis, the decisions taken in this regard are founded on the analysis of the problem constraints and of the approaches employed by other teams. Naturally, a different interpretation of these aspects can lead to a different set of design decisions. Even by not modifying the methods employed, the structure of the state and action spaces for the micro and macro level behaviours could reflect a shift in the boundary between the two. For example, incorporating some multi-agent coordination at the micro-level behaviour would allow some degree of local strategic planning. Similarly, a more elaborate macro level behaviour could improve the tactical decision of the agents through global knowledge. Ultimately, designing a solution which could seamlessly shift the boundary between the two, either statically or dynamically, would prove ideal.

A crucial step in developing a robust method is proper evaluation. However, even in the case of the official competition, defining properly balanced scenarios proves to be very challenging. Often, maps are either too easy or too difficult, yielding minute differences between the scores obtained by different teams or methods. To some degree this was also the case with some of the official maps used for the experiments presented in chapter 4. This led to the need of artificially creating the harder maps Kobe-hard, Paris-hard, Kobe2-f20 and Paris2-f20. Although the results of the experiments were largely consistent, suggesting a good performance of the more elaboarate approaches, BaGA and DCOP, some degree of dependence on the map particularities was apparent. As such, finding other maps with various other particularities can provide further insight on the strengths and weaknesses of the proposed methods. Also, the score metric

measured, while being a legitimate measure of the overall performance, is somewhat restricted. More precisely, it does not reflect the individual performance of the micro and macro level behaviours. As such, finding a more informative metric can lead to a more thorough evaluation.

Another aspect not addressed in this thesis is the behaviour of the police agents, which, in competition circumstances, can prove to have a crucial impact on the performance of all other agents. As mentioned in chapter 1, their behaviour may not be best suited for the methods specific to the DTP. Nonetheless, their abilities and presence can be coordinated with the behaviour of other agents, e.g., through the formation of heterogeneous teams, including multiple types of agents. As such, their support role can either be exploited at the micro level behaviour, aiding agents based on their immediate surroundings, or at the macro level behaviour, ensuring an overall accessibility to important parts of the map.

Lastly, throughout this thesis, the limitations with regard to inter-agent communication and memory consumption were not thoroughly addressed. While this allowed a good focus on the theoretical aspects of the studied approaches, they can have a serious impact on the overall performance, as highlighted in the official competition results and some of the presented experiments. As such, these aspects need to be taken into consideration in further research.

Appendices

Appendix A

2014 RoboCup results

Abiding to the official RoboCup rules and constraints, the approach described in this thesis formed the University of Amsterdam submission to the 2014 RoboCup championship in João Pessoa, Brazil [34]. The focus of the thesis was on modelling the behaviour of fire brigades and ambulance teams. Thus, other aspects with a significant impact on the competition score, such as the behaviour of police teams or the communication model were largely responsible for an overall low ranking. Nonetheless, on maps with relatively few blockades and sufficient communication capabilities, such as NY1, Paris1 and Mexico1, our team outperformed other competitors. The initial conditions and score evolution for these maps are presented in figures A.1 - A.6. One characteristic of the score graphs which does not occur in the graphs reported in chapter 4 is the stepped shape, apparent for example in the Paris1 map at timesteps 75, 170 and 190. This is due to the fact that the scores presented in this appendix also incorporate the civilian health component, which is sensitive to punctual events such as aftershocks or new fire ignitions. The complete results of the first 2 days of competition are presented in tables A.1 - A.2, along with a rough estimate of level of blockades and communication availability for each map. Due to some technical difficulties, the agents did not run properly on the Kobe1 and VC1 maps. Further details, such as replays, can be obtained at http://roborescue.sourceforge.net/2014/results/.

In the case of the Paris1 map, the agents successfully contain and extinguish 2 of the initial fires. A third one is discovered later and the closest fire brigades are blocked by debris. By the time other agents arrive to attend to it, at around timestep 65, it becomes too large to be completely extinguished. On the NY1 map a large number of agents were blocked in debris from the start of the simulation. As such, the exploration of the map was impaired and two of the initial fires positioned along the edges of the map were discovered relatively late. A third one, more central, was discovered earlier, but could not be completely extinguished completely due to the relatively few agents

available. However, they did manage to limit its spread to some degree. Finally, in the case of the Mexico1 map, the agents completely extinguished one of the initial fires before it spread to more than 2 buildings. The second initial fire grew larger and was almost completely extinguished. However a single burning building was missed by the agents, as most of them were refilling, which lead to an uncontrollable extension of the fire starting at around timestep 110.

Apart from the main competition, a special "Multi-agent challenge" was held [35]. This challenge focused only on the performance of the fire brigade agents and, as such, all other agent types were eliminated, along with road blockades and civilians. Furthermore, the agents were aware of all the fires on the map, with the goal of the challenge revolving around making optimal task assignments. This was successfully achieved by the macro-level behaviour and, in the absence of blockades and communication restrictions, our team obtained the highest overall score. The results for this challenge are detailed in table A.3.

Our team also participated in a secondary competition, the 2014 Iranian Open [36]. The results, shown in table A.4, were similar to the ones of the main competition in Brazil, usually higher for the maps with few blockades and good communication.

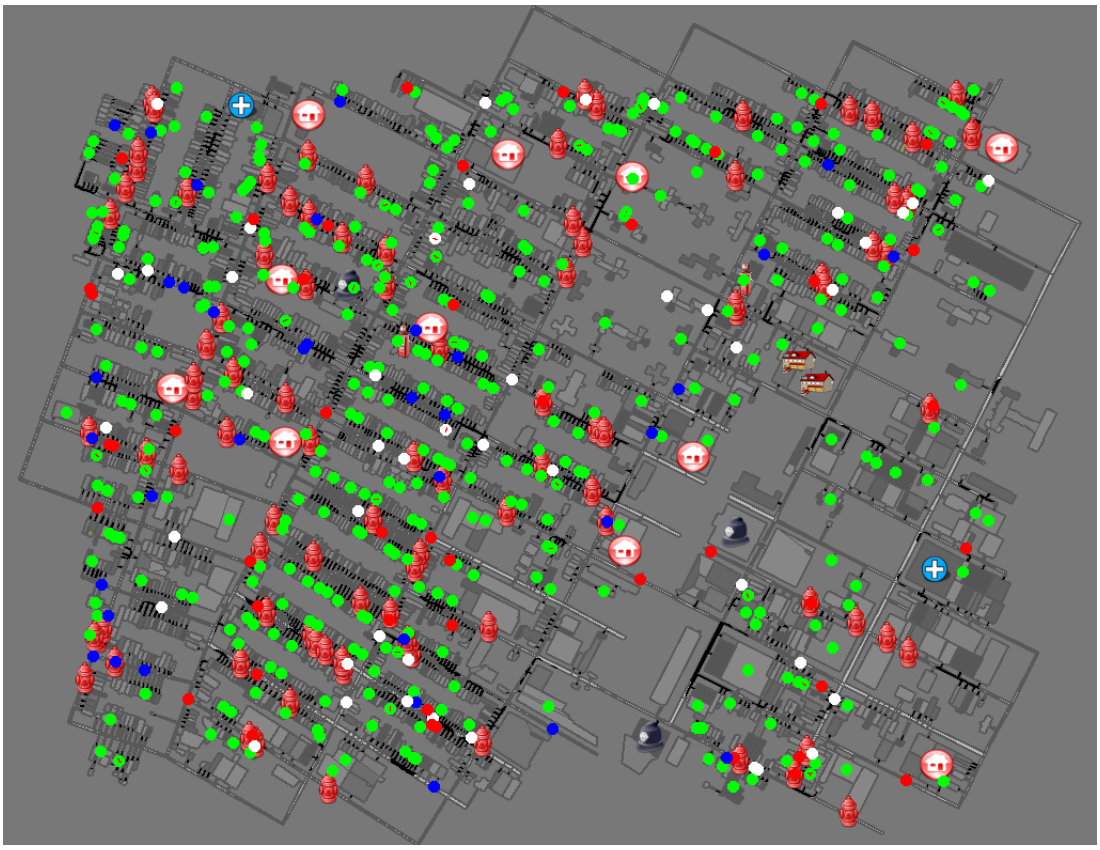

Figure A.1: Initial conditions of the NY1 map in the 2014 RoboCup Agent Simulation competition.

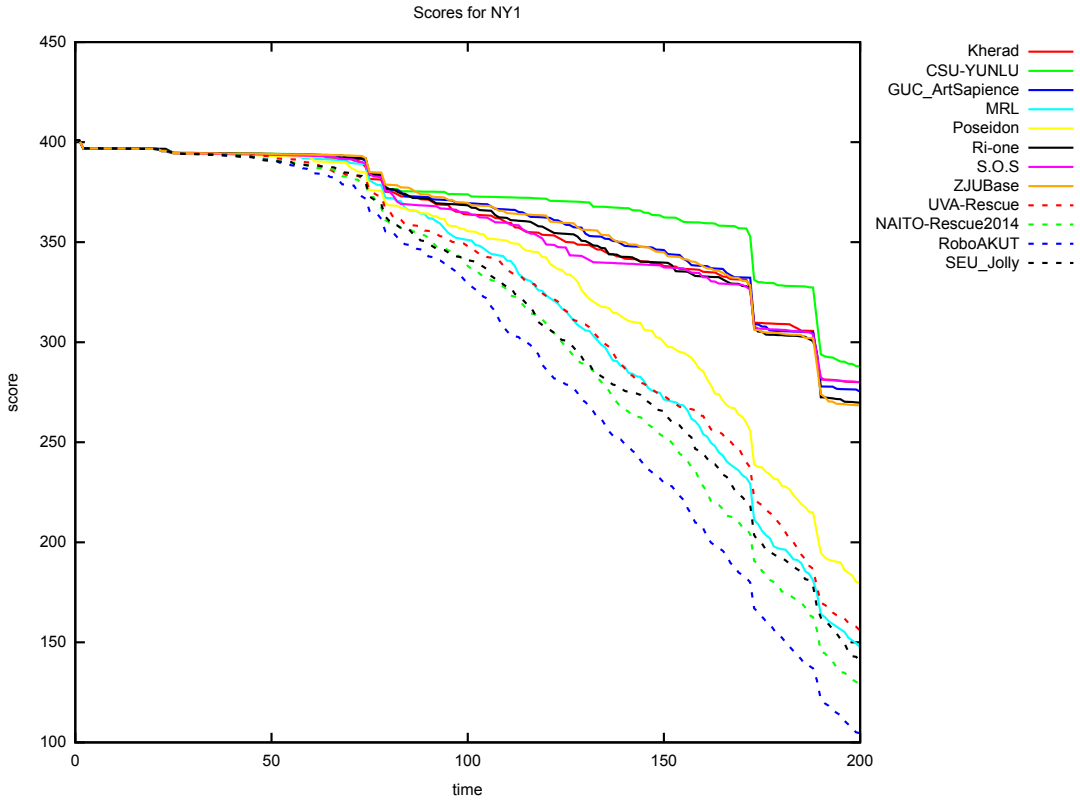

Figure A.2: Overall score for the NY1 map in the 2014 RoboCup Agent Simulation competition.

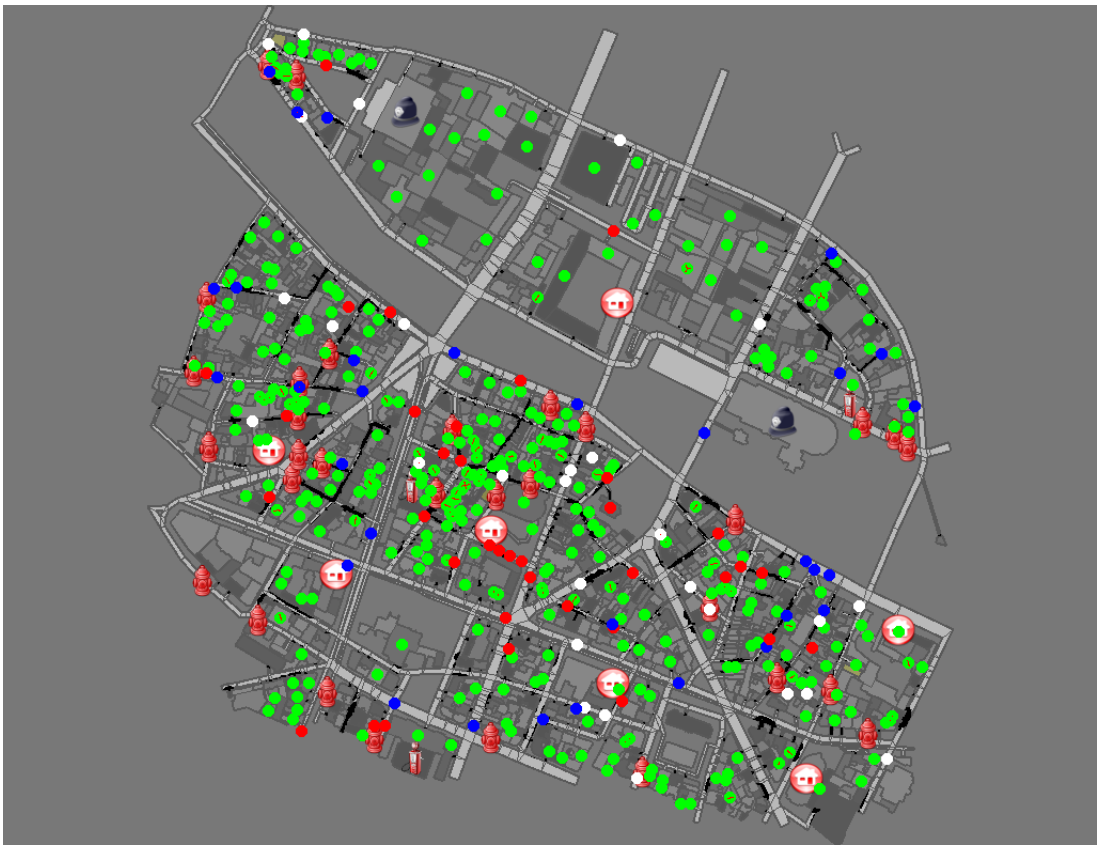

Figure A.3: Initial conditions of the Paris1 map in the 2014 RoboCup Agent Simulation competition.

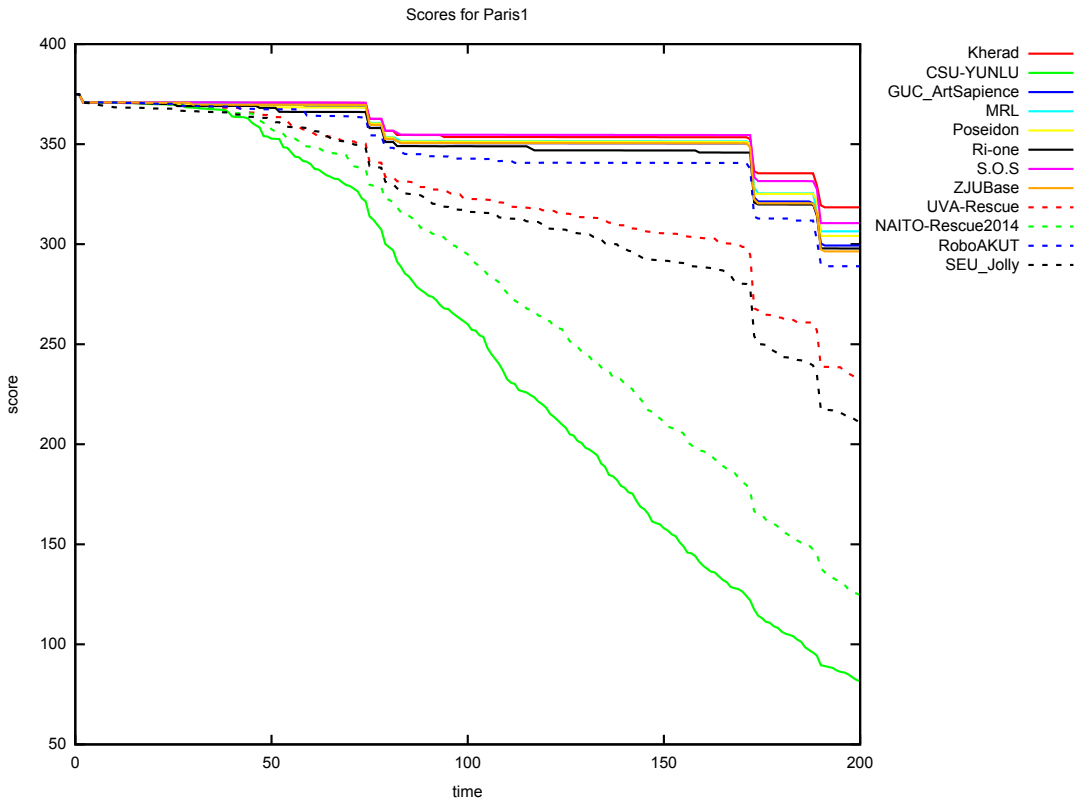

Figure A.4: Overall score for the Paris1 map in the 2014 RoboCup Agent Simulation competition.

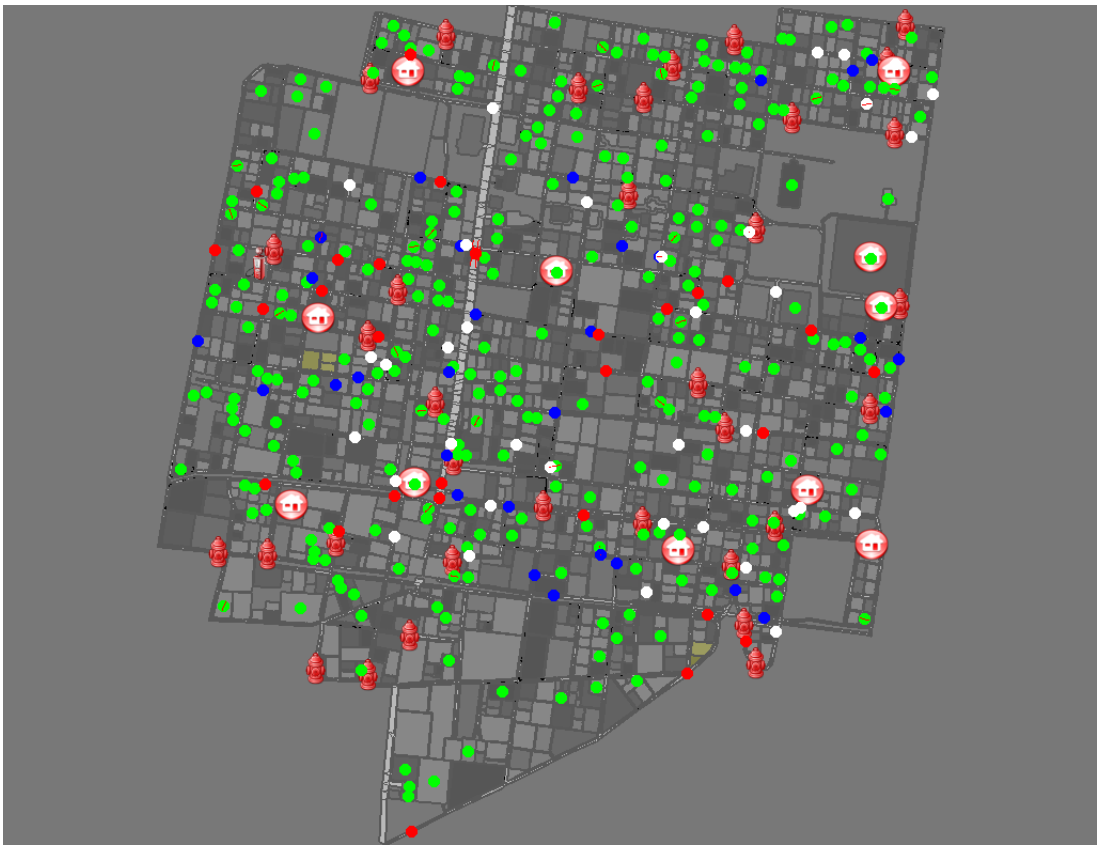

Figure A.5: Initial conditions of the Mexico1 map in the 2014 RoboCup Agent Simulation competition.

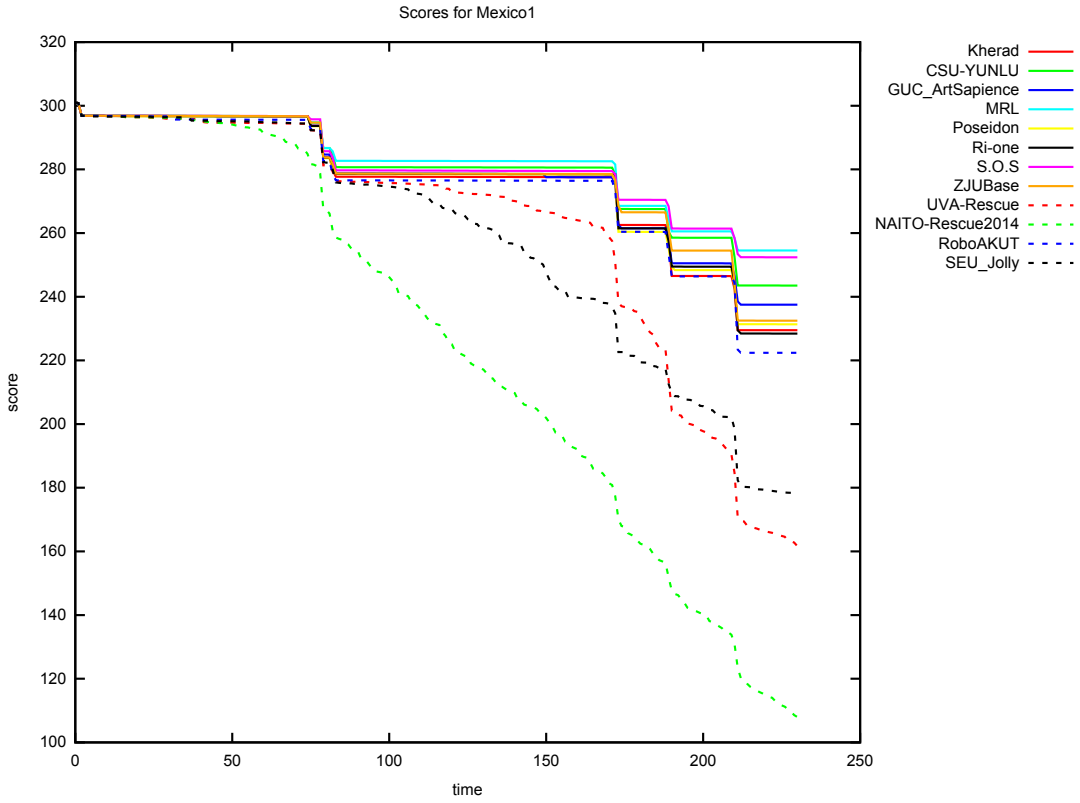

Figure A.6: Overall score for the Mexico1 map in the 2014 RoboCup Agent Simulation competition.

Team	Kobe1		VC1		Eindhoven1		Berlin1		Paris1		Total		Rank
Road blocks	++++		+++		+		++		+				
Communication	++++		++		++		+++		++++				
	Score	Points	Score	Points	Score	Points	Score	Points	Score	Points	Score	Points	
GUC_ArtSapience	15.58	11	81.93	22	55.41	10	141.81	19	299.32	19	594.06	81	3
MRL	38.57	24	74.26	20	128.73	23	159.72	24	306.38	21	707.66	112	1
S.O.S	34.34	21	84.47	23	0.00	1	153.87	22	310.49	22	583.17	89	2
ZJUBase	15.86	12	50.66	12	89.46	16	97.10	9	296.36	17	549.43	66	5
NAITO-Rescue2014	7.58	5	27.02	4	30.54	5	91.73	7	124.62	1	281.49	22	11
Poseidon	15.05	10	46.59	10	36.97	6	123.93	15	304.12	20	526.67	61	7
Ri-one	13.54	8	0.00	1	42.42	7	115.28	13	297.84	18	469.09	47	8
RoboAKUT	10.05	6	31.81	6	90.79	17	127.63	16	288.97	16	549.24	61	6
Kherad	14.21	9	36.74	7	129.44	24	66.77	2	318.39	24	565.55	66	4
UVA-Rescue	6.01	4	26.89	3	29.21	4	49.92	1	233.23	7	345.25	19	12
SEU_Jolly	0.00	3	47.05	11	46.02	8	134.83	18	210.99	3	438.90	43	9
CSU-YUNLU	11.13	7	85.03	24	55.04	9	43.69	1	81.61	1	276.50	42	10

Table A.1: Results following the first day of the 2014 RoboCup Agent Simulation competition.

Team	Joao1		NY1		Mexico1		Istanbul1		Kobe2		Day1		Total		Rank
Road blocks	+++		+++		+		++		+++						
Communication	++++		+++		+++		+		+						
	Score	Points	Score	Points	Score	Points	Score	Points	Score	Points	Score	Points	Score	Points	
GUC_ArtSapience	79.77	18	275.26	20	237.49	18	153.08	10	96.55	21	594.06	81	1436.21	168	3
MRL	81.82	22	147.59	2	254.53	24	214.96	23	98.61	23	707.66	112	1505.17	206	2
S.O.S	82.46	24	280.06	22	252.39	23	216.17	24	99.68	24	583.17	89	1513.93	206	1
ZJUBase	73.39	5	268.45	18	232.48	17	169.13	14	92.51	18	549.43	66	1385.39	138	6
NAITO-Rescue2014	74.20	7	129.77	1	107.98	1	57.85	1	43.82	1	281.49	22	695.11	33	11
Poseidon	76.84	13	179.77	7	231.35	16	138.82	8	92.50	17	526.67	61	1245.95	122	7
Ri-one	76.11	11	269.81	19	228.43	14	173.67	15	83.38	10	469.09	47	1300.48	116	8
RoboAKUT	73.45	6	104.62	1	222.37	13	158.75	11	92.46	16	549.24	61	1200.89	108	9
Kherad	79.63	17	279.99	21	229.50	15	182.90	17	95.62	20	565.55	66	1433.19	156	4
UVA-Rescue	62.67	1	155.83	3	161.66	1	61.18	1	34.54	1	345.25	19	821.13	26	12
SEU_Jolly	74.62	8	140.62	1	178.03	1	158.87	12	91.44	15	438.90	43	1082.49	80	10
CSU-YUNLU	80.58	20	287.95	24	243.52	20	190.88	18	97.47	22	276.50	42	1176.90	146	5

Table A.2: Results following the second day of the 2014 RoboCup Agent Simulation competition.

	Kobe1				Paris1				Paris2						
	Time	Damage	Score	Rank	Time	Damage	Score	Rank	Time	Damage	Score	Rank	Total Score	Sum Rank	Final Rank
UvA Rescue	141	145	0.7911	3	299	573	0.6443	1	176	108	0.9476	2	2.383	6	2
NAITO-Rescue2014	138	139	0.7932	2	299	1251	0.236	3	180	106	0.9466	3	1.9758	8	3
MRL	129	131	0.8089	1	299	1062	0.357	2	89	54	0.9877	1	2.1536	4	1

Table A.3: Multi-agent challenge results.

Score Table	Kobe-1	Point1	Paris-1	Point2	VC-1	Point3	Istanbul-1	Point4	Berlin-1	Point5	Map6	Point6	Total Point	Total Score
S.O.S	136.586	15	149.736	15	145.38	14	120.563	15	120.28	15	59.841	13	87	732.386
Soshiant	122.494	13	48.277	1	149.15	15	111.642	12	118.83	10	66.849	15	66	617.242
GUC_ArtSapience	105.948	10	148	14	130.556	10	72.26	5	120.275	14	52.422	11	64	629.461
Apollo	107.163	12	135.514	13	144.161	13	106.496	8	113.934	8	41.314	9	63	648.582
ZJU_Base	14.576	2	133.62	12	131.369	11	111.505	11	119.524	11	53.374	12	59	563.968
Kherad-Rescue	96.034	8	130.598	10	115.362	8	103.479	6	117.552	9	61.618	14	55	624.643
MRL	132.754	14	82.13	6	10.608	1	118.503	14	119.573	12	41.159	8	55	504.727
Poseidon	106.901	11	125.571	8	47.409	4	109.512	9	120.068	13	41.134	7	52	550.595
SEU_Jully	59.592	5	84.677	7	132.928	12	106.386	7	103.032	7	45.843	10	48	532.458
R.A.S_Roshd	103.006	9	133.608	11	126.093	9	112.381	13	54.837	1	21.413	1	44	551.338
Sorena	63.44	7	126.582	9	37.76	3	110.464	10	76.976	3	25.795	2	34	441.017
Simorgh	60.404	6	60.392	4	101.543	7	55.411	4	85.564	4	39.731	6	31	403.045
Distribot	22.084	3	50.545	2	73.792	5	38.4	3	100.94	6	31.018	5	24	316.779
UvA_Rescue	11.428	1	58.655	3	94.311	6	18.385	2	93.562	5	27.967	4	21	304.308
CSU_Yunlu	36.915	4	78.615	5	11.515	2	12.254	1	71.134	2	26.646	3	17	237.079

Table A.4: Results following the first day of the 2014 RoboCup Agent Iranian Open.

Bibliography

[1] http://www.robocuprescue.org/

[2] http://roborescue.sourceforge.net/2014/rules2014.pdf

[3] Boutilier, Craig, Thomas Dean, and Steve Hanks. "Decision-theoretic planning: Structural assumptions and computational leverage." arXiv preprint arXiv:1105.5460 (2011).

[4] Papadimitriou, Christos H., and John N. Tsitsiklis. "The complexity of Markov decision processes." Mathematics of operations research 12.3 (1987): 441-450.

[5] Ramchurn, Sarvapali D., et al. "Decentralized coordination in robocup rescue." The Computer Journal 53.9 (2010): 1447-1461

[6] Kleiner, Alexander, et al. "RMASBench: benchmarking dynamic multi-agent coordination in urban search and rescue." Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[7] Jie Shen, Zhiwei Liang, Zhiqian Liu and Siyu Zhang "RoboCup 2013 - Rescue Simulation League Team Description Apollo-Rescue (P.R. China)", RoboCup 2013, Eindhoven, July 2013.

[8] Hisayuki Sasaoka "Effectiveness of Agents' Communication using Pheromone in RoboCup Rescue Simulation System", RoboCup 2013, Eindhoven, July 2013.

[9] Fu Jiang, Shang Wang, Zengrong Luo, Qun Yang and Jun Peng "Rescue Simulation League Team Description CSU_YunLu-China", RoboCup 2013, Eindhoven, July 2013.

[10] Alan D. Barroso, Felipe de C. Santana, Victor Lassance, Annibal B. M. da Silva, Luis G. Nardin, Anarosa A. F. Brandão and Jaime S. Sichman "RoboCup Rescue 2013 LTI Agent Rescue Team Description", RoboCup 2013, Eindhoven, July 2013.

[11] James Parker, Ernesto Nunes, Julio Godoy and Maria Gini "MinERS Preliminary Report", RoboCup 2013, Eindhoven, July 2013.

[12] Sho Okazaki, Takahiro Nakagawa, Ko Miyake, Shinya Oguri and Masahiro Takashita "RoboCupRescue 2013 - Rescue Simulation League Team Description Ri-one (Japan)", RoboCup 2013, Eindhoven, July 2013.

[13] Afsoon Afzal, Parand Alizadeh, Mozhdeh Arian Nezhad, Kimia Ghaffari, Ghazal Hasanpour, Kiana Jahedi, Pooria Kaviani and Giti Omidvar "Poseidon Team Description Paper RoboCup 2013, Eindhoven", RoboCup 2013, Eindhoven, July 2013.

[14] Dai Obashi, Toshiyuki Hayashi, Nobuhiro Ito and Kazunori Iwata "Implementation of a communication library among heterogeneous agents: NAITO-Rescue 2013 (Japan)", RoboCup 2013, Eindhoven, July 2013.

[15] Wei Liu, Hongyu Zhao and Jinghao Lei "RoboCupRescue 2013 - Rescue Simulation League Team Description ZJUBase (P.R.China)", RoboCup 2013, Eindhoven, July 2013.

[16] Pooya Deldar Gohardani, Peyman Ardestani, Siavash Mehrabi, Mehdi Taherian, Sam Mirzaee Ramhormozi and Mohammad Amin Yousefi "RoboCup Rescue 2013 - Rescue Simulation League Team Description MRL (Iran)", RoboCup 2013, Eindhoven, July 2013.

[17] Dina Helal, Ahmed Abouraya, Noha Khater, Mina Fahmy, Fadwa Sakr, Salma Osama and Abdullrahman Elhusseni "RoboCup 2013 Rescue Agent Simulation Competition GUC ArtSapience Team Description Paper", RoboCup 2013, Eindhoven, July 2013.

[18] Yoosef Golshahi, Salim Malakouti, Seyed Mohammad Reza Modares Saryazdi, Sina Sheikholeslami, Sina Tagva, Hesam Akbari Morteza Rezayi Khoshdarregi, Navid Haeri, Angeh Aslanian and Aramik Markari "RoboCupRescue 2013 - Rescue Simulation League Team Description AUT S.O.S. (Iran)" , RoboCup 2013, Eindhoven, July 2013.

[19] Kschischang, Frank R., Brendan J. Frey, and H-A. Loeliger. "Factor graphs and the sum-product algorithm." Information Theory, IEEE Transactions on 47.2 (2001): 498-519.

[20] Ferreira Jr, Paulo Roberto, et al. "RoboCup Rescue as multiagent task allocation among teams: experiments with task interdependencies." Autonomous Agents and Multi-Agent Systems 20.3 (2010): 421-443.

[21] Scerri, Paul, et al. "Allocating tasks in extreme teams." Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. ACM, 2005.

[22] Dos Santos, Daniela Scherer, and Ana LC Bazzan. "Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach." Applied Soft Computing 12.8 (2012): 2123-2131.

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

[24] Emery-Montemerlo, Rosemary, et al. "Approximate solutions for partially observable stochastic games with common payoffs." Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on. IEEE, 2004.

[25] Sutton, Richard S., and Andrew G. Barto. Introduction to reinforcement learning. MIT Press, 1998.

[26] Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains." Artificial intelligence 101.1 (1998): 99-134.

[27] Bishop, Christopher M. Pattern recognition and machine learning. Vol. 1. New York: Springer, 2006.

[28] Oliehoek, Frans A., and Arnoud Visser. "A hierarchical model for decentralized fighting of large scale urban fires." (2006): 14-21.

[29] Rabinovich, Zinovi, Claudia V. Goldman, and Jeffrey S. Rosenschein. "The complexity of multiagent systems: The price of silence." Proceedings of the second international joint conference on Autonomous agents and multiagent systems. ACM, 2003.

[30] Bernstein, Daniel S., et al. "The complexity of decentralized control of Markov decision processes." Mathematics of operations research 27.4 (2002): 819-840.

[31] Oliehoek, Frans A., Shimon Whiteson, and Matthijs TJ Spaan. "Lossless clustering of histories in decentralized POMDPs." Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[32] Spaan, Matthijs TJ, Frans A. Oliehoek, and Christopher Amato. "Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion." IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Vol. 22. No. 3. 2011.

[33] Oliehoek, Frans A., Matthijs TJ Spaan, and Nikos A. Vlassis. "Optimal and Approximate Q-value Functions for Decentralized POMDPs." J. Artif. Intell. Res.(JAIR) 32 (2008): 289-353.

[34] Traichioiu, Mircea, and Arnoud Visser. "UvA Rescue-Team Description Paper-Agent competition-Rescue Simulation League RoboCup 2014-João Pessoa-Brazil." (2014).

[35] Traichioiu, Mircea, and Arnoud Visser. "UvA Rescue Team Description Paper Multi-Agent Challenge Rescue Simulation League RoboCup 2014-Joao Pessoa-Brazil." (2014).

[36] Traichioiu, Mircea, and Arnoud Visser. "UvA Rescue-Team Description Paper-Agent competition-Rescue Simulation League-Iran Open 2014." (2014).