A hierarchical model for decentralized fighting of large scale urban fires

Frans Oliehoek
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ
Amsterdam, The Netherlands
faolieho@science.uva.nl

Arnoud Visser
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ
Amsterdam, The Netherlands
arnoud@science.uva.nl

ABSTRACT

In this article we present a hierarchical model for planning and coordinating firefighting in situations such as RoboCup Rescue, where an urban earthquake is simulated. We show that a hierarchical approach brings leverage to the planning process. By using formal decision theoretic models, we also present a more formal analysis of the RoboCup Rescue domain. Finally we discuss how our model could be applied and treat further abstractions that may be necessary.

1. INTRODUCTION

Coordination of emergency services during disasters is a topic receiving a lot of attention lately: even specialized workshops and conferences are being organized. During an emergency different civil services have multiple teams patrolling the site, each with some degree of autonomy. The different emergency services may also have different information about different locations within the disaster site. However, communicating *all* potential information is typically not possible. As a consequence, coordination might fail.

In the RoboCup Rescue [7, 13] a prototype of a large scale disaster — an earthquake in an urban environment — is simulated. Because of the disaster, communication infrastructure is failing, buildings catch fire and collapse causing roads to get blocked by and people to get trapped in the debris. In this chaotic setting, teams of firefighters, police officers and ambulances have to make decisions, with the goal to gain control of the situation as quickly as possible and with minimal casualties and damage.

When considered from the perspective of artificial intelligence (AI), RoboCup Rescue is a cooperative multi-agent system in a partially observable stochastic world. Recently the Dec-POMDP [2] framework has gained in popularity for modeling such systems. However, the complexity of optimally solving these models (NEXP-complete [2]) has limited the application to the smallest problems.

Hierarchical decomposition is one way of reducing the complexity of problems [1]. In this paper we propose an hierarchical framework for fire fighting based upon the RoboCup Rescue simulation environment. We show how such an approach can bring a large reduction in complexity, especially when combined with state aggregation and abstraction. At the same time we present the RoboCup Rescue world using formal models (Dec-POMDPs, POMDPs). This provides a potential starting point for a more formal approach to RoboCup Rescue. Moreover, it sets a goal for approximating methods for Dec-POMDPs and POMDPs in order to be applied in such a complex real-world problem.

In section 2 we will first introduce the Dec-POMDP model. Next, in section 3, we will model RoboCup Rescue as a 'flat' Dec-POMDP. After that, we present our proposed hierarchical model in section 4. Section 5 threats how the parameters for the hierarchical model should be estimated and section 6 concludes with a discussion.

2. DEC-POMDPS

The *Decentralized partially observable Markov decision process (Dec-POMDP)* [2] is a model for multi-agent planning under uncertainty. Formally, a n-agent Dec-POMDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, O, h \rangle$ where:

- $\mathcal{S}$ is a finite set of states.
- The set $\mathcal{A} = \times_i \mathcal{A}_i$ is the set of *joint actions*, where$\mathcal{A}_i$ is the set of actions available to agent i. Every time-step, one joint action $\mathbf{a} = \langle a_1, ..., a_n \rangle$ is taken. Agents do not observe each other's actions.
- T is the transition function, specifying $P(s'|s, \mathbf{a})$.
- R is the reward function, $R(s, \mathbf{a}, s')$ gives the reward for a transition from s to s' under $\mathbf{a}$.
- $\mathcal{O}_i$ is a finite set of observations available to agent i. A joint observation $\mathbf{o} = \langle o_1, ..., o_n \rangle$ is chosen from the set of joint observations $\mathcal{O} = \times_i \mathcal{O}_i$.
- O is the observation function, specifying $P(\mathbf{o}|\mathbf{a}, s')$.
- h is the horizon of the problem.

The goal of the agents is to maximize the expected (discounted) future reward.[1] Therefore the planning problem is to find a conditional plan or *policy* for each agent as to maximize the expected (discounted) future reward.

When the Dec-POMDP consists of 1 agent, the model reduces to a regular POMDP, for which many results are known [6]. When such an agent can also deduce the state from the observation it receives, the problem is *fully observable* and the model reduces to a MDP, which has als been studied extensively [15].

[1] When the planning horizon is infinite, the expected future reward is discounted by a factor γ to make this sum finite.

There are also extensions of the Dec-POMDP model to include explicit communication [5, 16]. For now, we will assume that communication is part of the regular actions and observations. I.e., an individual action $a_i = \langle a_i^d, a_i^c \rangle$ consists of a domain level action a_i^d and a communication action a_i^c. This communication action influences the observations received in the next time-step in a straightforward manner through the transition model.

3. ROBOCUP RESCUE AS A DEC-POMDP

Here we will give a description of how one can model fire fighting based upon the RoboCup Rescue world as a regular ('flat') Dec-POMDP, before we extend this description to a hierarchical model. Although this is mainly a theoretical exercise as this model is highly intractable, it is useful as it reveals the complexity of the problem. At the same time we also present a more elaborate description of RoboCup Rescue.

3.1 State description

Here we will give a description of the proposed state space by identifying all relevant state variables or *factors*. This state space is based on the dynamic factors in the RoboCup Rescue world, restricted to the factors relevant for firefighting. Static factors, i.e. factors that do not change throughout the simulation, will not have to be incorporated in the state space. Their influence is incorporated through the transition model. In RoboCup Rescue, the world is given by a map, consisting of buildings, roads and nodes.[2] A typical map consists of approximately 1000 buildings and the same number of roads and nodes.

This world is populated by agents, broadly dividable in three categories: the mobile civilians and rescue agents (platoons) and the immobile rescue centers. The rescue agents (both mobile and center agents) can in turn be divided in the fire, police and ambulance agents. The center agents function as communication centers. In a typical simulation there are 70–90 civilians, 20–40 mobile rescue agents and 3 centers.

The mobile agents can be located on roads, nodes or in buildings. So the number of valid positions on a map is the sum of these elements (i.e., typically around 3000). Also these mobile agents have a particular health, expressed in health points (HPs). Fire-brigade agents have a particular amount of water left. Ambulance agents can have a civilian loaded or not.

Because of the earthquake that takes place at the beginning of the simulation, buildings collapse and catch fire. These collapses can cause roads to get blocked (in either direction)[3] and civilians to get trapped in debris. Although there are ideas to incorporate the effects of aftershocks (causing new collapses and fires), this currently is not implemented. Therefore we will not consider this.

The fire simulator is based on heat energy as primary concept. Fire propagation's main component is heat radiation, but the latest fire simulator also incorporates the influence of the wind [9] by dividing the open (non-building) area of the map in (approx. 20000) cells for which the air temperature is simulated. Other factors that determine the spread of fire are properties of buildings. The dynamic factors are the heat of a building, whether it is burning, how much fuel is left and how much water is left in the building (and thus how damaged it is). Static properties like the size and composition of a particular building (wood, steel or reinforced concrete) and how close it is to surrounding buildings also influence the spreading of fire. However, because these properties are static, they do not need to be incorporated in the state description. Instead the probability of a particular building i catching fire given that a neighboring building j is on fire is modeled through the transition model.

factor	values
for all fire agents in the world (±15)	
current position	valid positions
health	0–9999HPs
amount of water	0–15000l
for all roads (±1000)	
blocked? (2 directions)	blocked/free
for all buildings (±1000)	
heat energy	0–10^6 GJ
state	(not) burning
fuel left	percent(%)
amount of water	0–150000l
for all air cells (±20000)	
temperature	20-10000°C
wind speed and direction	n/a

Table 1: State variables or *factors* for a particular RoboCup Rescue world (map). The dark entries are ignored, as we focus on fire-fighting.

Table 1 summarizes the state factors. From the available descriptions is is not always entirely clear in how factors are represented (especially regarding the wind details are lacking), therefore these 'value' entries are based on judgement of the authors. Also note that, as this paper focuses on the firefighting aspect, certain factors (e.g. the position, health and other properties of other agents) are ignored.

3.2 Actions

The actions for an agent i in RoboCup Rescue can be divided in domain level actions $\mathcal{A}_i^d$ and communication actions $\mathcal{A}_i^c$. A mobile agent can perform both a domain level action as communication within one time-step (e.g. a fire-brigade agent can move/extinguish and communicate). This means that the set of actions $\mathcal{A}_i$ for a mobile agent i is the Cartesian product of all domain level and communication actions $\mathcal{A}_i = \mathcal{A}_i^d \times \mathcal{A}_i^c$. In this section we will discuss the domain level actions. Section 3.4 will deal with communication actions.

All mobile agents can perform the *move* action. The argument of this *move* action is a path along which the agent should move. Clearly the *move* actions are dependent on the current position of the agent. Also, there is a maximum distance that an agent can travel in one time-step (333m). This means that two paths that deviate only after this point lead to the same action.

Fire-brigades have 2 specialized actions: *extinguish* and

[2] Nodes act as the glue between roads/roads and roads/buildings.

[3] In RoboCup Rescue the blockage of roads is simulated with more detail (e.g. the measure of block and the cost to clear the blockage are incorporated). For the fire-fighting task, knowing whether a road is blocked or not (in both ways), is sufficient.

refill. The *extinguish* action takes is a compound action consisting of multiple *'nozzle'* actions. Each *nozzle* action specifies a building and the amount of water (in liters) to direct to that building.[4] The total amount of water can't exceed the maximum amount of water emitted per time-step (1000l), also a fire-brigade cannot use more water than it has left. Until 2005 there has been no use for the possibility to divide the amount of water over different buildings, but this has changed with the new fire-simulator that allows for preemptive extinguishing of buildings. The *refill* action restores the water supply of the brigade and can only be performed at 'refuges', these are special buildings were agents can find shelter. Refilling occurs at the same rate as extinguishing (1000l per time-step). The other agents also have specialized actions: police-brigades can clear the road current road and ambulances can rescue civilians and load and unload them into the vehicle. As we focus on fire-brigades we will not treat these in further detail.

3.3 Observations

Now we turn our discussion to observations. Similar as to actions we specify the set of observations for agent i as the Cartesian product of domain and communication observations $\mathcal{O}_i = \mathcal{O}_i^d \times \mathcal{O}_i^c$. Here we treat the domain observations, communication observations are treated in section 3.4.

At the beginning of the simulation all agents receive the full state of the world before the earthquake. This models the fact that rescue agents know their city. After this initial observing of the city only regular observations are received: at each time-step, only objects within a range of 10m are seen, except for fiercely burning buildings, which can be observed from a larger distance. When an agent executed a *move* action, only observations of the new position are received (i.e, no observations are made 'en route'). On average 4-6 static objects (building and roads) can be visually observed during a time-step.[13]

Observing an object means that the agent receives the object ID, its type and properties. For a road this property is whether or not it is blocked (in both ways), for a building, the so-called 'fieriness', listed in table 2, is observed. This fieriness factor is a direct function of the amount of fuel and water left in the building and determines the part of the area counted as damaged. As far as we were able to tell, the properties of the wind are not observed at all.

3.4 Communication

Communication consists of both an action (by the sender) as an observation (for the receiver). In RoboCup Rescue there are two forms of communication actions: *say* and *tell*. The *say* messages are directly transferred (i.e., shouted), the latter transmitted by radio. Both types of communication are broad-casted: *say* messages can be picked up by agents of all types within 30m, *tell* messages can be received by all agents of the same type regardless of the distance.[5] The restrictions that are posed on communication vary per competition. In 2005, platoon agents could send and receive 4 *tell* messages. For center agents this was $2n$ *tell* messages. Platoons additionally can also send and receive one *say* message.

[4]In fact a 'nozzle' action also requires (x,y) coordinates and a direction. We will assume that these can be simply derived from the building to be extinguished.

[5]An exception here is that center agents, can receive *tell* messages from other types of centers.

F.	description	ign.	ext.	dmg.
0	not burnt, no water damage	$\surd$	$\surd$	0
1	burning, slightly damaged	$\times$	$\surd$	1/3
2	burning, more damaged	$\times$	$\surd$	2/3
3	burning, severely damaged	$\times$	$\surd$	1
4	not burnt, watered-damaged	$\surd$	$\surd$	1/3
5	extinguished, slightly dam.	$\surd$	$\surd$	1/3
6	extinguished, more damaged	$\surd$	$\surd$	2/3
7	extinguished, severely dam.	$\surd$	$\surd$	1
8	completely burnt down	$\times$	$\times$	1

Table 2: Fieriness (F.) values for buildings on fire. Also shown is whether the building is ignitable, extinguishable and how much of the area is counted as damaged.

The above is difficult to model as a Dec-POMDP, because of the restrictions on the incoming messages: in RoboCup Rescue the agents are assumed to select the messages they want to hear, by selecting 4 messages based on only the sender's ID. This effectively introduces a sub-stage where an action has to be selected before the observation is received. To model this we introduce an additional state variable, *phase*, that alternatingly takes on the value *act* and *rc* (receive communication). Also, we introduce a variable *action* that will 'remember' the joint action taken.

When the *phase* is *act*, agents take an domain level and communication action as normal, the state than changes in the following way: *phase* is set to *rc* and *action* is set to the taken joint action, $\mathbf{a} = \left\langle \left(a_1^d, a_1^c\right), ..., \left(a_n^d, a_n^c\right) \right\rangle$. Next, the agents receive a special 'receive communication' observation. For agent i we will denote this observation by o_i^{rc}. This observation consists of all the message IDs and the corresponding sender IDs for all messages agent i is able to receive (all *tell* messages and the *say* messages of agents within 30m). Next, the agents have to select a 'receive communication' action, a_i^{rc}. This action corresponds to the 4 messages the agent wants to receive. Next the state changes again: the state stochastically changes, however, now the transition is dependent on the *action* variable, instead of the joint 'receive communication' action $\mathbf{a}^{rc}$. I.e., let a state in this new representation be denoted $\bar{s} = \langle s, \mathbf{a}, rc \rangle$, we than have $P(s'|\mathbf{a}, s) = P(\langle s', \mathbf{a}_\emptyset, act \rangle \,|\, \langle s, \mathbf{a}, rc \rangle) = P(\bar{s}'|\bar{s})$, where $\mathbf{a}_\emptyset$ is an empty joint action (in the *act* phase no remembered actions are needed). As is implied during this transition *phase* is set to *act* and *action* is set to $\mathbf{a}_\emptyset$. Finally, the communication observations are received at the beginning of the new *act* phase. Here the 'receive communication' action $\mathbf{a}^{rc}$ is relevant (as it determines what messages are received by what agent). We assume that the selected messages are received without noise, i.e., we have $P(\mathbf{o}^c | \langle s, \mathbf{a}, rc \rangle, \mathbf{a}^{rc}) = 1$ for exactly one $\mathbf{o}^c$. Note that this observation is dependent on the previous rc state. In contrast, the domain observation $\mathbf{o}^d$ depends on the new state: $P(\mathbf{o}^d|\mathbf{a}, s') = P(\mathbf{o}^d|\mathbf{a}, \langle s', \mathbf{a}_\emptyset, act \rangle)$. Therefore, for a combined joint observation $\mathbf{o} = \left\langle \mathbf{o}^d, \mathbf{o}^c \right\rangle$, this can be written as:

$$P(\left\langle \mathbf{o}^d, \mathbf{o}^c \right\rangle | \bar{s}, \mathbf{a}^{rc}, \bar{s}') = P(\mathbf{o}^c | \bar{s}, \mathbf{a}^{rc}) \cdot P(\mathbf{o}^d | \mathbf{a}, \bar{s}')$$

In a pure planning framework, messages have no a priori semantics. Instead the planning process should embed the 'optimal meaning' in each communication action [5]. In RoboCup Rescue all messages are 256 bytes. When an agent can send 4 *tell* and 1 *say* message, this means has $8 \cdot 256 \cdot 5 = 10240$ bits to encode its communication action and thus that $|\mathcal{A}_i^c| = 2^{10240}$. This means that the number of joint communication actions $|\mathcal{A}^c| = 2^{10240n}$. Clearly this way of treating communication is an intractable factor.

3.5 Transition, observation and reward model

The transition model of the flat MDP as described so far, would describe the probabilities of all next states given the previous states and actions. Because the state description is the same as used by the simulator components, these probabilities could theoretically be found by analyzing the code of the simulation system.

The (domain) observation model is even simpler, given a successor state, the observations an agent receives are deterministic, i.e., given s' there is no uncertainty regarding $\mathbf{o}^d$. Also $\mathbf{o}^d$ is independent of the actions taken: $P(\mathbf{o}^d|\mathbf{a}, s') = P(\mathbf{o}^d|s')$, because agents receive no observations 'en route' as discussed.

The reward function is easily derived from the scoring function. The scoring function used for the 2005 competition was:

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

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

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

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

Note that the initial score $Score(s_0)$ is the highest score possible and that 0 is the lowest score possible. This means that cumulative reward is bounded by $[-Score(s_0), 0]$. As the focus is on firefighting, only the second term of eq. 1 will be used.

Although the horizon is finite in the RoboCup Rescue competition (300 time-steps) we typically want to plan for a varying horizon (until all fire is extinguished and all trapped people are either rescued or dead). We allow this by treating the problem as infinite horizon. Because we know that the reward is a finite, we don't need to introduce a discount factor.

3.6 Complexity

Here we will give a brief argument of the complexity of a flat Dec-POMDP representation. By ignoring some factors of the state-space, we effectively performed a first abstraction to reduce the state space. However, the state space as presented in table 1 is still huge. When there are $n = 15$ fire-brigade agents and 3000 valid positions this already leads to 3000^{15} different configurations. When considering only the first 4 factors, we already get a state space of

$$\begin{aligned} |nr\_pos|^{15} \cdot |HPs|^{15} \cdot |water|^{15} \cdot 2^{|2 \cdot nr\_roads|} &= \\ 3000^{15} \cdot 10000^{15} \cdot 15000^{15} \cdot 2^{2000} &\approx \\ 10^{52} \cdot 10^{60} \cdot 10^{62} \cdot 10^{602} &= 10^{776} \end{aligned}$$

and this is not even including the state of each of the 1000 buildings. With the number of atoms in the universe being estimated around 10^{85}, clearly it is not possible to represent the full state space. We already saw that the number of joint communication actions is prohibitively large and the same holds for domain actions and observations. Therefore this is clearly an intractable problem.

However, in the actual world a fireman who's extinguishing some building, typically won't care about properties of a building beyond the zone of potential fire spreading. Nor will he consider what the exact position of a colleague at the other side of town is. Effectively, in order for a firefighter to perform his job he needs focus on those state variables that are relevant for his current task. States that do not differ on relevant variables can be grouped or *aggregated*. We use these insights to provide a hierarchical decomposition of tasks, bringing leverage to the decision process, by constraining the number of possible policies.

4. A HIERARCHICAL FRAMEWORK

In RoboCup Rescue hierarchical techniques have implicitly been applied. Especially the division of the map in regions or sectors and the use of roles have been frequently employed by teams in the competition [13, 10]. The inclusion of roles within a model similar to the Dec-POMDP and its potential application to domains RoboCup Rescue like has also been considered [8]. We will first discuss some theory of hierarchical approaches. The appliance in our proposed hierarchical framework for RoboCup Rescue is discussed after that.

4.1 Hierarchical approaches to planning

The goal of hierarchical approaches is to reduce the complexity of a problem, eloquently put by Barto and Mahadevan [1] as:

> "Recent attempts to combat the curse of dimensionality have turned to principled ways of exploiting temporal abstraction, where decisions are not required at each step, but rather invoke the execution of temporally-extended activities [...] this leads naturally to hierarchical control architectures..."

The mentioned temporally-extended activities are at the heart of hierarchical approaches and are also referred to as partial policies, options, skills, behaviors, modes or activities. The last of these is suggested for use in more general context, while the preceding terms are used for specific formalisms. Activities or behaviors [3], the term we will adopt for reasons that will become apparent, typically are employed to accomplish some task or sub-goal. In this paper we present a framework of behaviors that can be used for the RoboCup Rescue world.

Most current work on hierarchical decision making is based on the framework of *semi-Markov decision processes (SMDPs)*. These are like regular MDPs, but the state transitions take a variable amount of time. As a consequence the probability of a transition is now written as:

$$P(s', \Delta t \mid s, a).$$

In particular, most approaches are based upon *discrete-time* SMDPs, where the delay Δt is restricted to be an integer multiple of underlying time-steps t.

In hierarchical approaches, this generally is applied in the following way. A behavior, denoted β, can either be a *primitive action a* or a partial policy: A partial policy is defined over a subset of states $\mathcal{S}$ and finishes (possibly stochastically) upon reaching a state in a set of sub-goal states. The transition function $P(s', \Delta t | s, \beta)$ gives the probability that behavior β started from state s finishes after Δt time-steps and that the resulting state is s'. Here s' is a state in which the sub-goal of behavior β is accomplished.

We propose a different way to model the uncertainty regarding the completion of a sub-task that remains closer to way that regular MDPs specify the transition probabilities. Instead of considering how long a particular behavior will take to terminate, we consider how far the task has progressed after a fixed number of time-steps. I.e. we fix Δt to some integer T and consider how the state has changed after these T time-steps:

$$P^T(s' \mid s, \beta),$$

where s' is not necessarily a state where the sub-goal of β is accomplished, but can be any ('intermediate') state. We refer to such a model as an T-MDP. This also motivates our choice for the term 'behavior': the agent will behave in a certain way for a fixed number of time-steps.

Clearly, it is not possible to consider one-step actions in the same T-MDP as the behaviors: an T-MDP with $T > 1$ is only meaningful when the behaviors β are actual policies and not primitive actions. In fact, it is very likely that different behaviors cannot be considered at the same time-scale T either. Therefore we will make use of a hierarchy of models for different time-scales, similar to how MAXQ [4] uses a hierarchy of SMDPs.

An additional benefit of our approach is that it is possible to switch from behavior before a sub-goal is completed without having to resort to special techniques to guarantee that sub-tasks are finished or behaviors aborted within a specific time. Especially within highly dynamic and partially observable environments this is an important asset.

Of course there is also a catch: it might be possible that an agent finishes the task for which the behavior is designed within T time-steps. There are two ways to reduce the effects of this threat. The first is to select T (which is experimentally determined) as a small number, such that a sub-task typically needs several T periods to finish. This will reduce the relative delay. A second option is to define the behaviors in such a way that, when the primary task of a behavior is fulfilled, it specifies other actions that are beneficial to perform. E.g., when a fire is extinguished, it can't hurt to sweep the neighborhood for trapped civilians.

4.2 An hierarchy of (Dec-)POMDPs

Here we will describe a possible hierarchy of tasks and behaviors and discuss how this reduces the complexity of the overall problem by distributing the reasoning over different hierarchical levels. Also, to overcome to inherent complexity of planning over communication (as to embed to optimal meaning of messages as described in section 3.4) we will assume a fixed communication policy. This also eliminates the need to introduce a separate sub-stage for making decisions about communication.

First we give description of the hierarchy as a whole. The highest level considers the city as a whole. This city consists of districts which form the intermediate level. The lowest level we consider is the 'fire zone' level which corresponds to a block of buildings on fire. At each of these levels, the fire-agents can select different behaviors, these are summarized in table 3. As planning at the highest level is performed the least frequent (the suggested T is the highest) we assume that information reaches all agents and planning is performed using a centralized model. At lower levels, planning has to be performed more often and is assumed to be decentralized.

level	behaviors	T
city	perform duty in district 1,2,...	20
district	fight fire zone 1,2,..., refill, patrol	5
fire zone	move, extinguish	1

Table 3: Levels of hierarchy and the behaviors and suggested T at those levels.

Because we want to avoid the complexity of planning over communication, we assume that there is a fixed communication policy. It is intuitively clear that the agents will be able to make better decisions if they have better information regarding the world. Therefore we assume that this fixed policy is tailored to communicating the observations agents receive. Of course it is possible also to allow other types of communications (i.e. requests, orders, intentions etc.). However in a Dec-POMDP it is optimal to communicate all observations if communication if free [16]. Of course it is most likely not possible to send and receive all observations, but we assume the fixed communication policy tries to share the most relevant observations.

4.2.1 City level

The highest level we consider covers the entire city in which the disaster is assumed to take place. The state representation for this level is given by table 4 and illustrated in figure 1. Although this state space is still large, it is much smaller than the state space of the flat Dec-POMDP.

for all fire-agents:	
current position	district
for all districts:	
total heat energy	0-10^8GJ
damage	destroyed area

Table 4: State representation at city level.

Basically, the decision process that is performed at this level is that of how to distribute the fire agents over districts. Clearly fire agents should be assigned to districts where large and dangerous fires are burning. However, it is also important to assign brigades to districts from which there is no information, as there might be fires there that are still to small to observe.

As we assumed that the fixed communication policy communicates observations, we will treat observations received directly the same at those received through communication. That is, at this level we ignore the delay in observations received through communication. The actual observations considered at this level are all fieriness observations and the positions of each agent accumulated over the last 20 time-steps.

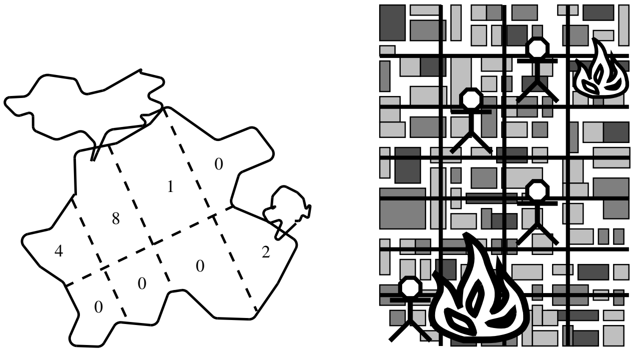

Figure 1: Left: city-level assignment of fire agents. The numbers show the number of fire agents in each district. Right: illustration of assignment to fires within a district. See text for description.

At this level there is only one type of behavior defined: selecting each district is a behavior: *district(i)*. After this selection reasoning is resumed at district level for the selected district.

We assume that at the highest level, control is 'centralized'. Because of the larger T (a suggested value of 20), we assume that total heat information (observations) of all districts has reached all agents and that therefore all these agents can solve the centralized POMDP and execute their component of the joint behavior.

4.2.2 District-level

The district level is the intermediate level in our proposed hierarchy. It is modeled as a Dec-POMDP with the behaviors *extinguish_fz, patrol* and *refill*. The agents participating in the Dec-POMDP for district i are those agents that selected behavior *district(i)* at the city level. However, as each agent individually calculates the policy for the city level, there might be discrepancies: i.e., agent i might select a policy based on the assumption that agent j is also assigned to this district, but agent j might have selected a different district. To counter this problem, the agents send their computed behaviors to the center, which distributes them to all agents as described, as part of the fixed communication policy. If necessary, agents can re-plan using the correct *district(i)* behaviors.

for all fire agents in district:	
position	pos. in district
amount of water	0–15000l
for fire-zone 1...#FZ:	
position	'neighborhood'
total heat energy	0-10^7GJ
damage	destroyed area

Table 5: State representation at district level.

The state space of the Dec-POMDP at this level is given in table 5. The position of the fire agents in the district are restricted to actual positions in the district and a special *outside_district* value. Because fire zones can have varying sizes, we need a more 'fuzzy' way to encode their location. We do this be dividing the district in 'neighborhoods'. We than define the position of a fire zone to be the neighborhood in which the most of its burning houses are located. The state representation of the district level is also illustrated in figure 1.

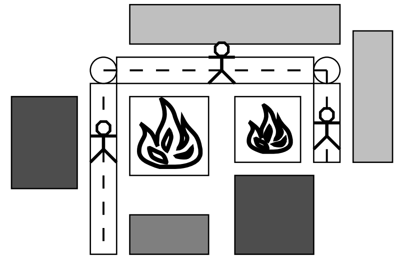

Figure 2: Planning at the fire zone level. The number of agents and buildings to be considered is much lower.

Similar to the city level, the observations considered at the district level are the fieriness properties of buildings, but now restricted the buildings within the district, and the position of agents, but restricted to agents assigned to this district.

At this level there are multiple types of behaviors. The *fz_extinguish(fz)* is similar to the *district(i)* behavior and selects a fire zone to extinguish. When the agent is inside the selected fire zone, reasoning is resumed at fire zone level, otherwise he navigates to the fire zone and start reasoning at fire zone level after arrival. The *refill* behavior specifies to navigate to a refuge (if necessary) and than to refill. The behavior *patrol* makes the agent patrol the district, searching for fires.

In the above behaviors, we assume that there is a procedure for navigating to a particular goal and that there is a policy for the behavior *patrol*. This could also be modeled as a (Dec-)POMDP, but we restrict our hierarchy to firefighting tasks.

4.2.3 Fire-zone level

The lowest level at which reasoning is performed is the 'fire zone' level. A fire zone is a consecutive block of buildings on fire together with the adjacent buildings and roads as illustrated in figure 2.

Similar to the district level, the fire zone level is modeled as a Dec-POMDP, where the participating agents are those that selected the *extinguish_fz(i)* behavior. The state representation used at this level is shown in table 5.

Because the extent of the fire zone changes as the fire spreads, it is necessary to let the state description follow this change. This means that the state description has to be adaptive. As soon as the fire spreads, the building(s) that now become adjacent to the fire (and thus part of the fire-zone) are included in the state description. This is also the level of the hierarchy that would incorporate the influence of the wind. However, as it appears that agents are not able to observe the wind at all, it is not profitable to use this for planning and the concerning factors have been omitted.

Clearly, this level is the most complex in the hierarchy; although the number of considered agents is the smallest, the number of factors considered for each building is larger and the number of considered buildings and roads tends to outweigh the former for all but the smallest, most isolated

for all fire-agents that extinguish this fire-zone:	
position	pos. in FZ
health	0–9999
water	0–15000l
for all buildings in fire-zone:	
heat	0-10^6GJ
state	(not) burning
amount of fuel left	percent(%)
amount of water	0–150000l
for all roads in fire-zone:	
blocked	yes/no — 2 direct.

Table 6: State representation at fire zone level.

fires.

The observations considered at this level are the fieriness properties of buildings within the fire zone, the exact positions of agents that extinguish this fire zone and the state of roads within fire zone.

At this lowest level the agent plan regarding their primitive actions *move* and *extinguish*. However, when an agent runs out of water, this ends the current extinguish fire zone behavior end immediately executes the refill behavior from the district level.

4.3 State abstraction and aggregation

Above we presented a three level hierarchy to replace the flat Dec-POMDP as presented in section 3. At each of these levels the number of state factors is a lot less than in this flat Dec-POMDP. As the size of the state space is exponential in the number of factors it is described by, this means that we have reduced to complexity a lot. However, the complexity of the hierarchical model as described is still prohibitive. We will discuss how state abstraction and aggregation can be applied to further counter this without going into too much detail.

The most straightforward measure is to make a coarser discretization of all factors with many values. This is a kind of state aggregation, as we are in fact grouping similar states. In fact we already applied this kind of aggregation at the city level, where we specified the agents' positions in districts instead of precision locations. This can also be applied to other state factors, e.g. the amount of water could be expressed in kilo-liters (kl), the health on a scale of 1-5 etc.

State abstraction (abstracting away certain factors) is also possible. For instance, the amount of water an agent has might be ignored at the fire zone level, as it is also considered at the district level. The fact that we ignored factor for the influence of the wind at the fire zone level is also a form of abstraction.

Similar techniques could be applied to the observations, especially at the higher levels: Instead of considering each change in fieriness for each individual building, it may be possible to merge these observations into a more coarse observation indicating the change in fieriness for each neighborhood or district. I.e., to apply a filtering or aggregation on the observations.

One can question the usefulness of the hierarchical model as presented here, as apparently there is still a need to perform aggregation and abstraction. In the authors' opinions this is not the best perspective. A better perspective is to recognize, that hierarchical decomposition in fact facilitates making better abstractions. I.e, when considering the flat Dec-POMDP model as presented in section 3 it is much harder to perform aggregation and abstraction while preserving most of the dynamics of the system.

5. PARAMETER ESTIMATION AND SOLUTION METHODS

In the previous section we presented a hierarchical model for fire fighting and described the states, observations and actions (behaviors) at the different levels. What we did not discuss yet are the transition and observation function. The estimation of transition probabilities has become less straightforward, because the abstractions made make it more difficult to relate these transitions to the algorithms implemented by the simulator. Moreover, the transition model at the higher levels is dependent on the quality of the policies executed at the lower levels. E.g., the probability that the total heat energy of a specific fire zone i becomes less when 3 agents select the behavior *extinguish_fz(i)*, depends on how well these 3 agents perform.

This suggests that the transition model should be learned in a bottom-up fashion. I.e., first the transition model for the fire zone level should be estimated. Because this level is still represented by the same entities as the simulator uses (primitive actions, same factors and observations), it should be possible to derive a fairly accurate transition model by examining the code. Next, the transition model for the district level can be estimated (learned) by repeated simulations, given the policy used for the fire zone level. I.e., by writing the district level $(s, \mathbf{a}, s')$ tuples that are encountered during the simulation runs to a log file (the true states are accessible from within the simulator), $P(s'|s, \mathbf{a})$ can be determined by analysis. The city level transition model can be determined with a similar procedure.

For the observation probabilities, similar approaches can be taken. This is slightly simpler, however, as there is no dependence on the quality of lower-level policies (observations probabilities are conditioned on the successor state). For the fire zone level, the observation probabilities is deterministic in the sense that it is certain which observations are received in which states. For the higher two levels, the probability of (aggregated) observations can be determined by simulation as described above.

Alternatively, it would be possible to not to try and estimate the models, but to use a reinforcement learning approach. However, hierarchical multi-agent reinforcement learning in partially observable worlds is an unexplored topic that most likely will prove to be extremely hard.

Another aspect we did not cover in the preceding section, is how, given we accurately learned or estimated the transition and reward models, the resulting (Dec-)POMDPs should be solved. Even though the size of the models in our hierarchy is much smaller than a flat model, it will typically remain to large to solve exactly. As a consequence approximating methods are needed.

For the top-level POMDP, approximate POMDP solution [17, 14, 12] methods could be used. These algorithms, compute a joint policy for the entire belief space off-line. In the on-line phase every agent simply executes his component of the joint action specified by this joint policy π for the top level of the hierarchy. I.e., every $T(=20)$ time-steps each agent performs a belief update, based on the city-level observations which are assumed equal for all agents, giving a new belief b'. Next, each agent i looks up $\mathbf{a} = \pi(b')$ and executes its own component a_i. Another possibility would be using an on-line algorithm such as presented in [11]. This particular method starts a branch and bound n-step lookahead search starting from the current belief state. The advantage is that there is no planning over parts of the belief space that are never reached and thus no expensive off-line computations. The disadvantage is that special mechanisms are necessary to guarantee that the branch and bound search is performed in the same way, such that the agents will perform the same calculations and remain coordinated.

For the Dec-POMDPS — the lower levels of the hierarchy — similar satisfying approximating methods should be examined. Because of the higher complexity of Dec-POMDPs, it is likely that on-line algorithms, performing a lookahead search as described above, are the most viable option. However, it could be that good off-line methods will developed that could also be used. In this way the lower levels of the hierarchy set a benchmark for real-life application of approximate Dec-POMDP methods.

6. DISCUSSION

We presented a hierarchical framework for fighting large scale urban fires based on the RoboCup Rescue environment. We showed that such a hierarchical approach brings significant leverage when compared to a non-hierarchical approach.

From a theoretic perspective, we proposed an application of Dec-POMDPs for a complicated real-world scenario, thus supporting the relevance of these models as a basis for multi-agent planning under uncertainty. From a more practical perspective, we formally model fire fighting in RoboCup Rescue setting a starting point for a more formal treatment of this problem. We also discussed some abstraction and aggregation methods that may be applied to further reduce the complexity.

In this paper we explicitly specified the behaviors and thus sub-tasks Another, more interesting, approach would be to automatically discover (hierarchies of) sub-tasks and the corresponding behaviors. This might prove to be too costly, however. Another interesting question is whether the proposed method is actually useful for the current RoboCup Rescue worlds. It might turn out that a three level hierarchy in fact is unnecessary currently and that the power of such an approach becomes only clear for larger maps. These questions however will have to be experimentally verified.

7. ACKNOWLEDGMENTS

The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

8. REFERENCES

[1] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcent learning. *Discrete event dynamic systems: Theory and applications*, 13:343–379, 2003.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002.

[3] R. A. Brooks. Achieving artificial intelligence through building robots. Memo 899, MIT AI Lab, May 1986.

[4] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[5] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 137–144, New York, NY, USA, 2003. ACM Press.

[6] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

[7] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjoh, and S. Shimada. Robocup rescue: Searh and rescue in large-scale disasters as a domain for autonomous agents research. In *Proc. 1999 IEEE Intl. Conf. on Systems, Man and Cybernetics*, pages 739–743, October 1999.

[8] R. Nair, M. Tambe, and S. Marsella. Role allocation and reallocation in multiagent teams: towards a practical analysis. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 552–559. ACM Press, 2003.

[9] T. A. Nssle, A. Kleiner, and M. Brenner. Approaching urban disaster reality: The resQ firesimulator. Technical report, Universitt Freiburg, 2004.

[10] S. Paquet, N. Bernier, and B. Chaib-draa. DAMAS-Rescue description paper. Technical report, DAMAS labaratory, Laval University, 2004.

[11] S. Paquet, L. Tobin, and B. Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2005.

[12] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025 – 1032, August 2003.

[13] S. Post and M. Fassaert. A communication and coordination model for 'robocuprescue' agents. Master's thesis, University of Amsterdam, June 2004.

[14] P. Poupart and C. Boutilier. VDCBPI: an approximate scalable algorithm for large POMDPs. In *Advances in Neural Information Processing Systems 17*, pages 1081–1088, 2004.

[15] M. L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

[16] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of AI research (JAIR)*, 16:389–423, 2002.

[17] M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.