On Applicability of Automated Planning for Incident Management

Lukáš Chrpa
PARK Research group
School of Computing and Engineering
University of Huddersfield

Kristinn R. Thórisson
Icelandic Institute of Intelligent Machines, Reykjavik &
School of Computer Science, Reykjavik University

Abstract

Incident management aims to save human lives, mitigate the effect of accidents, prevent damages, to mention a few of their benefits. Efficient coordination of rescue team members, allocation of available resources, and appropriate responses to the realtime unfolding of events is critical for managing incidents successfully. Coordination involves a series of decisions and event monitoring, usually made by human coordinators, for instance task definition, task assignment, risk assessment, etc. Each elementary decision can be described by a named action (e.g. boarding an ambulance, assigning a task). Taken as a whole, the team coordinating an incident response can be seen as a decision-making system. In this paper, we discuss how invaluable assistance can be brought to such a system using automated planning. In consultation with experts we have derived a set of requirements from which we provide a formal specification of the domain. Following the specification, we have developed a prototype domain model and evaluated it empirically. Here we present the results of this evaluation, along with several challenges (e.g uncertainty) that we have identified.

Introduction

Automated planning is a research discipline that addresses the problem of generating a totally- or partially-ordered sequence of actions that transform the environment from some initial state to a desired goal state. Automated planning has been successfully applied for decades in several areas, including space exploration (Ai-Chang et al. 2004) or machine tool calibration (Parkinson, Longstaff, and Fletcher 2014), to mention a couple.

Incident management consists of time- and resource-critical operations that aim to save human lives, mitigate the effect of traffic or industrial accidents, prevent excessive damages, to mention a few of their benefits. In other words, incident management involves groups of people who need to achieve close coordination to carry out one or more time-critical tasks. Carrying out such tasks may be very stressful even for well trained professionals. Efficient coordination of rescue team members, as well as efficient allocation of available resources, are key determinants of success. Incident management is typically planned, coordinated, and monitored from rescue centres by human coordinators. Rescue operation planning involves a series of decisions such as which team member will do what task, whether a team should reach an incident location by car or by helicopter, and so on. Making these decision "manually" can be inefficient and error prone, even for well experienced professionals. Automated rescue operation planning can both be used to assist people during an actual live incident and to provide simulation for training exercises. The technological requirements for either case are more or less the same; the system must be able to represent objects with various static and temporal properties, i.e., anything from an unchanging building or a vehicle whose only change is from position x to y (and associated energy consumption) to an accident victim whose vitals are continuously changing via highly complex processes and a tornado causing vast changes to the environment and conditions in a short amount of time. Planning thus involves making decisions about events over which rescue teams have control, while being constrained by those out of their control.

Existing approaches for rescue operation planning (and execution) consist of coordination of (heterogeneous) autonomous vehicles to perform given tasks (e.g. surveillance of some area) (Doherty and Heintz 2011; George, Sujit, and Sousa 2011). Plans are often sequences of waypoints that are passed to the autonomous vehicles, and emphasis is given on "low-level" planning, i.e., planning of elementary vehicles' manoeuvres. For incident management planning, "high-level" planning (e.g. releasing a trapped victim) is sufficient. The most related "high-level" planning work is about road traffic incident management (Shah et al. 2013), which also inspired our work. Incident management is more general, although abstract concepts are similar to those in traffic incident management.

In this paper, we present our ongoing work involving automated planning in incident management. Typically, incidents are managed by coordinators whose decisions are based on experience and are made in a semi-reactive way. Automated planning can provide a complete overview on the incident management task, i.e., from the initial situation when incident(s) are reported to the goal situation when incidents are successfully dealt with. This is especially useful when the coordinator has to deal with situations that are unusual or when training of some specific skills

of rescue teams is designed. Automated planning can be straightforwardly used in incident management since each elementary decision the incident coordinator must take (e.g. a *paramedic1* boards the *ambulance2*) can be formalised as an action. Plans generated by planning engines provide the coordinator (partially ordered) sequences of elementary decisions that must be taken during the rescue operations. However, these plans must be provided quickly (a few seconds latency at most) and in a good quality (nearly optimal), otherwise it might have a negative impact on success of these operations. We have derived a set of requirements, in consultation with experts, from which we provide a formal specification of the domain. Following the specification we have developed a prototype domain model using which we are able to generate plans quickly and in reasonable quality as we empirically verified. Of course, incident management control requires a complex system involving planning and execution episodes overseen by human coordinators. We have designed an architecture of such a system and have identified several challenges, mostly due to uncertainty, that we also discuss in this paper.

Related Work

Search and Rescue (SAR), which can be understood as a subset of incident management, is the search for and provision of aid to people in need (often after a disaster). SAR is a well established research topic. The research in this area is promoted by the Robocup-Rescue project[1] that was motivated by earthquake in Kobe in 1995. This led to the RoboCup Rescue Robot and Simulation competitions that have been held since 2000 (Akin et al. 2013). Most of the SAR approaches consist of (semi)-autonomous robotic systems that are especially useful in "high-risk" areas such as Fukushima after the 2011 incident (Sato, Muraoka, and Hozumi 2014). Besides systems coordinating multiple heterogeneous vehicles (Doherty and Heintz 2011), a system supporting human-robot teams in disaster management scenarios has been recently developed and deployed (Kruijff et al. 2014). SAR often takes place in military operations. Comirem (Smith, Hildum, and Crimm 2005), a system incorporating mixed-initiative planning and scheduling in order to allocate resources more efficiently and with strong emphasis to the user interface, has been applied to Special Operational Forces planning.

Traffic accident management, which deals with situations after traffic accidents in order to provide aid to involved victims as well as to restore the situation into normal, can be also understood as a subset of incident management. Here, the need is to coordinate a team of human agents (paramedics, policemen etc.) rather than robots, so there might not be a need for "low level" control. Applying AI planning in Traffic accident management has been studied from the perspective of stochastic integer programming (Ozbay et al. 2013) as well as of "classical" domain-independent planning (Shah et al. 2013).

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

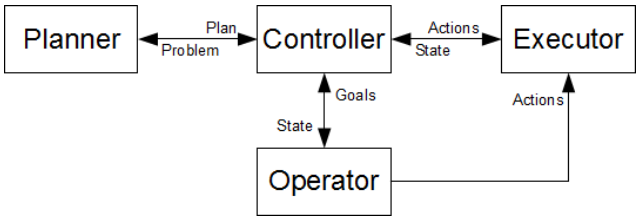

Figure 1: An architecture of an incident management system

Incident Management and Control

A system for incident management and control requires a planning/execution platform that is overseen by a human coordinator. Similar existing platforms such as T-REX (Rajan and Py 2012), which is run on-board of autonomous vehicles, or NEPTUS (Dias et al. 2006), which is a command and control system for autonomous vehicles, provide a framework for planning and executing operations for these vehicles. In our case, we do not need a "low-level" planning and control for coordinating autonomous vehicles, since human agents (rescue team members) are able to autonomously execute "high-level" actions (e.g. drive from one place to another). Our system architecture, depicted in Figure 1 is inspired by the architecture of T-REX.

The *Planner* component consists of one or more planning engines and one or more domain models. It receives the planning problem descriptions from the Controller component and provides solution plans back to it.

The *Executor* component consists of rescue team members who execute actions provided by the Controller component or the human coordinator (the Operator component). During the operation (executing the actions) rescue team members inform the Controller about their progress (e.g. whether the action has been successfully completed) as well as about the state of the environment (e.g. whether an incident victim is seriously injured).

The *Operator* component refers to a human expert who monitors the operations, setting up goals and, eventually, gives orders to the rescue team members (the Executor component). The latest gives the human coordinator to overrule the system in case of some unexpected event, emergency or error in the system.

The *Controller* component is the 'brain' of the system. It receives information about the operation progress from the Executor component and stores information about the current state of the environment. It accepts mission descriptions (goals) from the Operator component from which it generates planning problems that are sent to the Planner component. Plans retrieved from the Planner component are dispatched to the Operator and Executor components. The Controller component also monitors whether execution of plans matches what has been planned. In case of discrepancies, or intervention from the Operator component, re-planning is triggered.

Domain Specification

Given the requirements from experts, we provide a formal conceptualisation of the incident management domain. The way how the conceptualisation is made was inspired by work of Shah et al. (2013). The ontology has three main categories: object types, relation and function types, and action types. The incident management domain is specified on an abstraction and expressiveness level that is relatively independent of specific scenarios and planning engines. We will consider time in our specification, so hereinafter let T be a set of time-stamps that refer to a given scenario (a planning episode). Hereinafter, the constant $\perp$ will refer to an 'undefined value'.

The main abstract *object types* in the incident management domain are as follows:

- *Assets* $X = X_s \cup X_m$ are divided into two categories, *static assets* X_s (e.g. buildings) and *mobile assets* X_m (e.g. vehicles). Assets can accommodate *artefacts* or *agents* (see below).
- *Agents* A are intelligent entities that can interact with assets or artefacts (e.g rescue team members).
- *Artefacts* Y are various objects that cannot act on their own (e.g. incident victims, first aid kits). Artefact can be loaded to assets or carried by agents.
- *Locations* L are 'points of interest' (e.g towns, incident locations).

Notice that agents were not considered in work of Shah et al., since it was assumed that agents are "connected" to assets (e.g. a paramedic is always in an ambulance). Here, we distinguish between agents and assets, since an asset can accommodate agents of various capabilities (e.g. a helicopter can take paramedics, policemen or rescuers).

The main abstract *relation and function types* in the incident management domain are as follows:

We define a relation $conn \subseteq (X_m \cup A) \times L \times L$ for determining whether a mobile asset or agent can move between locations. We define a function $at : (X \cup A \cup Y) \times T \rightarrow L \cup \{\perp\}$ referring to a position of an object in a given time-stamp. Clearly for a static asset the function at is constant (does not change its value with time). We define a function $in : Y \times T \rightarrow (X \cup A \cup \{\perp\})$ referring to situations when an artefact is loaded to an asset or carried by an agent (or $\perp$ if not). Similarly, we define a function $inside : A \times T \rightarrow X \cup \{\perp\}$ referring to whether an agent is inside some asset or not ($\perp$). Each asset or agent may have a limited *capacity* which, informally said, stands for a maximum number (or weight) of loaded or carried artefacts (or agents) in the same time. Formally, a capacity is defined as a function $cap : X \cup A \rightarrow \mathbb{N}$, *volume* of agents or artefacts is represented by a function $vol : A \cup Y \rightarrow \mathbb{N}$, and, finally, *fullness* of agents or assets in a given time-stamp is defined as a function $full : (X \cup A) \times T \rightarrow \mathbb{N}$.

We also define *properties* as functions characterising a state of an object in a given time-stamp. Properties can reach values that are specific for a given class of objects. For example, a 'status' of an incident victim might have one of the following values: GREEN (no injury), YELLOW (minor injuries), RED (severe injuries), BLACK (dead). Agents have *knowledge* about the environment that can be obtained by observation or communication with other agents. Let a finite set K be an *universal knowledge base*. We define a function $k : A \times T \rightarrow 2^K$ referring to a knowledge of agents.

The environment specified by object, relation and function types can be modified by actions, specified via preconditions (what must be met in order to apply the action) and effects (what is changed in the environment after applying the action). We define the following *action types* which modify the environment of the incident management domain. We assume that the action is applied in a time-stamp t and lasts for Δt time.

move(e, l_1, l_2) moves a mobile asset or agent $e \in X_m \cup A$ from a location l_1 to a location l_2 ($l_1, l_2 \in L$). As a precondition it must hold that $(e, l_1, l_2) \in conn$ and $at(e, t) = l_1$. The effect of applying the action is that $at(e, t + \Delta t) = l_2$ and $\forall t' \in (t, t + \Delta t) : at(e, t') = \perp$.

load(z, x, l) loads an artefact or agent $z \in Y \cup A$ to an asset $x \in X$ in a location $l \in L$. As a precondition it must hold that $at(z, t) = l, \forall t' \in [t, t+\Delta t] : at(x, t') = l$ and $\forall t' \in [t, t + \Delta t) : full(x, t') \leq cap(x) - vol(z)$. The effect of applying the action is that $in(z, t + \Delta t) = x$ (resp. $inside(z, t + \Delta t) = x$), $\forall t' \in (t, t + \Delta t) : in(z, t') = \perp$ (resp. $inside(z, t') = \perp$), $\forall t' \in (t, t+\Delta t] : at(z, t') = \perp$ and $full(x, t + \Delta t) = full(x, t) + vol(z)$.

unload(z, x, l) unloads an artefact or agent $z \in Y \cup A$ from an asset $x \in X$ in a location $l \in L$. As a precondition it must hold that $\forall t' \in [t, t + \Delta t] : at(x, t') = l$ and $in(z, t) = x$ (resp. $inside(z, t) = x$). The effect of applying the action is that $at(z, t + \Delta t) = l$, $\forall t' \in (t, t+\Delta t] : in(z, t') = \perp$ (resp. $inside(z, t') = \perp$) and $full(x, t + \Delta t) = full(x, t) - vol(z)$.

fetch(y, a, l) allows an agent $a \in A$ to fetch an artefact $y \in Y$ in a location $l \in L$. As a precondition it must hold that $at(y, t) = l$, $\forall t' \in [t, t + \Delta t] : at(a, t') = l$ and $\forall t' \in [t, t + \Delta t) : full(a, t') \leq cap(a) - vol(y)$. The effect of applying the action is that $in(y, t + \Delta t) = a$, $\forall t' \in (t, t + \Delta t) : in(y, t') = \perp$ and $\forall t' \in (t, t + \Delta t] : at(y, t') = \perp$.

drop(y, a, l) drops an artefact $y \in Y$ carried by an agent $a \in A$ in a location $l \in L$. As a precondition it must hold that $\forall t' \in [t, t + \Delta t] : at(a, t') = l$ and $in(y, t) = a$. The effect of applying the action is that $at(y, t+\Delta t) = l$, $\forall t' \in (t, t + \Delta t] : in(y, t') = \perp$ and $full(a, t + \Delta t) = full(a, t) - vol(y)$.

communicate(a_1, a_2, m) sends a message $m \in K$ from an agent a_1 to an agent a_2 ($a_1, a_2 \in A$). As a precondition it must hold that $m \in k(a_1, t)$ and communication must be possible between a_1 and a_2 in a time interval $[t, t + \Delta t]$. The effect of applying this action is that $k(a_2, t + \Delta t) = k(a_2, t) \cup \{m\}$.

observe(a, m) allows an agent $a \in A$ to get an elementary knowledge $m \in K$ by observing the environment. The effect of applying this action is that $k(a, t + \Delta t) = k(a, t) \cup \{m\}$.

```
(:durative-action firstaid
 :parameters (?p - paramedic
              ?f - firstaidkit
              ?v - victim
              ?l - location)
 :duration (= ?duration 20)
 :condition (and
    (over all (at ?p ?l))
    (over all (at ?v ?l))
    (over all (in ?f ?p))
    (at start (available ?p))
    (at start (injured ?v)))
 :effect (and
    (at start (not (available ?p)))
    (at end (available ?p))
    (at end (not (injured ?v)))
    (at end (aided ?v)))
)
```

Figure 2: The First-aid action in PDDL.

interact$(a, l, \dots)$ changes a property (or properties) of an object (or objects). At least one agent ($a \in A$) must be involved and be present in a location l ($l \in L$), i.e., $\forall t' \in [t, t + \Delta t] : at(a, t') = l$. Also, a cannot be simultaneously involved in another interact action.

From the description of the action types we can derive the following invariants: $\forall y \in Y, \forall t \in T : at(y, t) = \perp \Leftrightarrow in(y, t) \neq \perp$, $\forall a \in A, \forall t \in T : inside(a, t) \neq \perp \Rightarrow at(a, t) = \perp$ (the opposite implication does not hold because the agent might be 'on the way'), $\forall z \in A \cup X, \forall t \in T : full(z, t) \leq cap(z)$, and $\forall z \in A \cup X, \forall t \in T : full(z, t) = \sum_{\{x|in(z,t)=x \vee inside(z,t)=x\}} vol(x)$.

Clearly, the above specification does not include all possible constraints – further constraints relate to specific assets, agents or artefacts. For example, incident victims (as a subclass of artifacts) can be loaded to ambulances and cannot be loaded to fire trucks (both are subclasses of mobile assets). The interact action type is specified in a very general way here, generalising a heterogeneous set of actions (e.g. giving first aid, extinguishing fire etc.)

A fundamental challenge in incident management is dealing with the situation immediately after an incident (or incidents) has been reported. The goal of a incident management team or coordinator is to restore the situation to the normal order (e.g. providing first aid to incident victims and transporting them to hospitals).

Prototype Domain Model

Following the domain specification given above we have developed a prototype domain model. The domain is developed in PDDL 2.1 (Fox and Long 2003), following the same requirements as in the Temporal Track on the International Planning Competitions, as these requirements are widely supported by temporal planning engines.

In our domain model, we consider the following types of *static assets*: Hospital, Fire station, Rescue station, Police station and Building; *mobile assets*: Ambulance, Police car, Fire truck, Rescue car and Helicopter; *agents*: Paramedic, Fireman, Policeman, Rescuer; and, finally, *artefacts*: Victim, First-aid-kit, Extinguisher, Rescue equipment.

```
...
0.09: (move ambulance0 reykjavik hella) [50]
1.10: (board policeman0 helicopter0 reykjavik) [1]
1.11: (fetch paramedic0 firstaidkit2 reykjavik) [1]
1.12: (fetch paramedic1 firstaidkit0 reykjavik) [1]
1.13: (fetch rescuer0 rope0 reykjavik) [1]
1.14: (fetch rescuer1 rope1 hella) [1]
2.15: (move helicopter0 reykjavik landmanalaguar) [15]
2.16: (firstaid paramedic0 firstaidkit2 victim4 reykjavik) [20]
2.17: (firstaid paramedic1 firstaidkit0 victim1 reykjavik) [20]
2.18: (rescue rescuer0 rope0 victim0 reykjavik) [20]
2.19: (rescue rescuer1 rope1 victim2 hella) [20]
17.20: (debark policeman0 helicopter0 landmanalaguar) [1]
18.21: (move helicopter0 landmanalaguar hekla) [5]
18.22: (secure policeman0 landmanalaguar) [10]
...
```

Figure 3: A fragment of a sample plan.

We have also defined the following properties that come on top of the relations and function defined in the previous section. Assets can be *onfire* or *extinguished*. Locations can be *unsecured* or *secured*. Finally, victims can be *injured* or *aided*, and *trapped* or *released*.

Actions are derived from the action types introduced in the previous section. In case of action types *move, unload, fetch* and *drop*, actions in our domain model are encoded straightforwardly according to descriptions of the action types. For the action type *load* we have implemented two variants of actions. One is specific for victims, i.e., victim can be loaded to a mobile asset only if the victim is *released* and *aided*; the other is general and encoded straightforwardly from the *load* action type. We have implemented four actions that extends the *interact* action type:

secure(p, l) allows a policeman p to secure a location l, i.e., the property of l changes from *unsecured* to *secured*.

extinguish(f, e, x, l) allows a fireman f to extinguish an asset x by an extinguisher e in a location l. It must hold that $\forall t' \in [t, t + \Delta t] : at(x, t') = l \wedge in(e, t') = f$. The effect is that the property of x changes from *onfire* to *extinguished*.

firstaid(p, fa, v, l) allows a paramedic or rescuer p to give a first aid to a victim v using a first-aid-kit fa in a location l. It must hold that $\forall t' \in [t, t + \Delta t] : at(v, t') = l \wedge in(fa, t') = p$. The effect is that the property of v changes from *injured* to *aided*. The PDDL encoding of this action is depicted in Figure 2.

release(r, re, v, l) allows a rescuer r to release a victim v using a rescue equipment re in a location l. It must hold that $\forall t' \in [t, t + \Delta t] : at(v, t') = l \wedge in(re, t') = r$. The effect is that the property of v changes from *trapped* to *released*.

Notice that action types *sense* and *communicate* are not implemented. This is due to the requirements for a deterministic and fully observable environment. Clearly, sensing and communication are needed for environments that are partially observable. Moreover, it creates contingency (non-deterministic action effects). Such issues are discussed later.

	LPG-td		Yahsp3-MT	
	T	Q	T	Q
Small	0.62±0.42	116±24	0.16±0.18	143±27
Medium	3.22±1.82	178±63	0.11±0.10	249±63
Large	5.45±1.79	419±83	0.39±0.39	384±86

Table 1: Results showing runtime of the planners in seconds (T) and quality (make-span) of the firstly generated plans (Q) with respect to different size of problems.

	Small	Medium	Large
Time	1.35±2.10	4.25±3.38	3.60±2.54
Quality	109±20	197±41	313±47

Table 2: Results showing runtime of Yahsp3-MT in seconds and quality (make-span) of the highest quality generated plans with respect to different size of problems.

Our prototype domain model deals with the following incidents: an asset in fire, a location to be secured from public, and injured and/or trapped victims. The goal is to manage these incidents such that: the asset in fire is extinguished, the location is secured, and the victim is rescued and eventually transported to the hospital (if his/her injury is serious). A fragment of a sample plan is depicted in Figure 3.

Experimental Evaluation

For the experimental evaluation of our approach we used a scenario consisting of: 5 locations, 1 hospital, 1 ambulance, 1 helicopter, 2 of each of the remaining types of assets, 2 of each agents, 3 first-aid-kits, 2 extinguishers and 2 units of rescue equipment. Each agent is 'loaded' to "its" asset (e.g. a police man is in the Police station) and mobile assets are in the same locations as "their" static assets (e.g. an ambulance is in the same location as the hospital). Then, we randomly select some locations where we set their property to 'unsecured', generate n victims and distribute them randomly along the locations with randomly assigned properties to *injured* and/or *trapped*. Finally, we generate k buildings randomly distributed along the locations with properties set to *onfire*. The goal is to change all unsecured locations to secured, buildings being on-fire to extinguished, victims being trapped to released, and finally, victim changed from being injured to aided and some of them to be delivered to the hospital.

We defined 3 classes of problems: "Small" ($n = 5$, $k = 2$), "Medium" ($n = 10$, $k = 3$) and "Large" ($n = 20$, $k = 5$). We generated 5 problems per each class and for solving them we used four state-of-the-art planning engines: Yahsp3-MT (Vidal 2014), LPG (Gerevini, Saetti, and Serina 2003), Popf2 (Coles et al. 2010) and Optic (Benton, Coles, and Coles 2012). We have observed that Popf2 as well as Optic do not scale well in our domain model, so the time needed to find solutions considerably increases with problem size. On the other hand, LPG as well as Yahsp scales well and time needed to extract the first solution is within a few seconds even for the large instances as shown in Table 1. LPG and Yahsp thus comply with one of the required criteria – obtain solutions in at most a few seconds. However, quality of first solutions is often not very good. On the other hand, both LPG and Yahsp can incrementally improve solutions, so it is possible to obtain solutions of better quality. We took a closer look on Yahsp, where we have observed that solutions can improve considerably, as depicted in Table 2, even while keeping quite strict cutoff of 10 seconds.

Our domain model does not require concurrency, so it is possible to solve problems as classical ones and then schedule actions from the plans in order to minimise make-span. This seems to be the main reason that extended classical planners (LPG and Yahsp) performed much better than "purely" temporal planners (Popf2 and Optic).

Given the performance of Yahsp, we can obtain solutions in reasonable quality (however, sub-optimal) in a very little time even for relatively large problems. Generated plans to large extent comply with expectations of the domain experts. This gave us a promising outlook for applying AI planning in incident management. Clearly, it applies for "standard" situations where it is not necessary to involve a large number of entities and/or consider complex cooperative actions (e.g. releasing the victim while giving him/her first aid). For disaster management, where thousands of entities are involved, we might use a different domain model, which is more abstract. For considering cooperative actions, a modified domain model is also required.

Challenges

We have identified several challenges related to application of AI planning in incident management. These challenges can be divided into three categories, namely Task Complexity, Uncertainty and Goal prioritisation. Most of these challenges can possibly be overcome without necessarily changing the domain model we have presented, or introducing more expressiveness (e.g. conformant planning). Testing this belief is planned in our future work.

Task Complexity

In real world scenarios, it is common to have a huge amount of objects being possibly in a plenty of different relations. Hence, we might have an excessive number of possible actions do deal with. AI planning is generally intractable, so it is impossible to handle very large models. However, we do not have to represent everything in very detail, we can either abstract or relax. For example, the firstaid action is a good example of abstraction. Although there are various ways how to give the first aid to incident victims, which mostly depends on what sort of injuries they have, professional paramedics know how to give the first aid (i.e. execute the firstaid action) without need to provide details. Also, we do not have to, for example, consider road traffic, so we can relax it. In normal conditions, it can marginally affect the driving time of rescue vehicles. If traffic is very heavy, we might consider, in the worst case, the road being blocked, and if there is no alternative that we might assume that relevant locations are not connected for the rescue vehicles (we apply abstraction).

Our prototype domain model, therefore, does not have to deal with an excessive number of objects. Problems are

thus relatively easy to solve. Possible issues are related to accuracy of the model and understanding the correct representation of the objects. In our case, most of actions are restricted to a single agent. However, in some case having more agents to perform an action might be more appropriate. For instance, if a victim has a serious injury, then more paramedics might give the victim the first aid. Also, an actual meaning of "rescue equipment" might vary regarding the situation in which it is used. Rescuers might use a different equipment for rescuing a victim in the mountains than in the forest. While the latter issue can be tackled by some sort of meta-reasoning, the former needs an enhanced domain model.

Unsolvable Problems

Solvability of the problem cannot be guaranteed. The problem might become unsolvable if, for example, some location is unreachable, or some kind of artefact is not present (e.g. fire extinguisher). In these cases the system can easily find the reason and notify the mission coordinator, so s/he can remove problematic goals.

Similarly, we cannot guarantee that the solution will be retrieved in a given time limit. The reason might be that the problem is too large (too many goals), or there are some specific constraints that makes the problem too hard for a planner. Although we believe that this is an unlikely scenario, if it occurs, the coordinator has to take over the control. In particular, the coordinator might decide to control the mission manually, or remove some (less important) goals in order to make the problem easier.

Uncertainty

In real world scenarios, unexpected situations may arise just before, or during, the operation being carried out. While there is always a human coordinator who can overrule the system, the system must be able to appropriately react on such situations.

Bad weather is often an issue for incident management. Although weather might be rather unpredictable in long term (days), it can be well predictable for short-term (hours). Since incident management is usually of short-term, weather be considered in the problem description by providing some restrictions (e.g. a helicopter cannot fly to locations affected by T-storms).

It might not be known how long some actions will take. For example, giving first aid to a victim with minor injuries will take much less time than first aid to a victim with severe injuries. However, severity of victim's injuries might not be very well known, since incidents are often reported with many inaccuracies. Therefore, several plans considering different action durations (e.g. optimistic, realistic and pessimistic estimation) might be generated. If these plans vary widely the human coordinator may decide which plan will be executed.

Often, complete information about the environment may not be available and thus we need sensing actions (e.g. "observe location X"). Sensing actions lead to non-deterministic contingencies. To make it deterministic we might consider the most likely outcome of the sensing action. If the actual outcome is different we might re-plan. Alternatively, we might provide plans for every outcome of the sensing actions (unless there are too many possible outcomes).

Goal Prioritisation

Another problem might be prioritising of some goals. It might be the case, for example, that life of one of the victim is in danger, so the victim has to be rescued as quickly as possible. If there are more victims in different locations the planner might not consider priorities while planning when it optimises for 'make-span'. Of course, there is a possibility to put priorities into the problem definition and force the planner to optimise for them. However, such a feature is not widely supported by planners. A possible solution to the prioritisation issue is to isolate goals with a very high priority and generate plans achieving only these goals. The remaining goals can be achieved in a separate plan, where, of course, objects (e.g. agents) that are involved in the former plan will not be used.

Conclusion

In this paper, we have derived a set of requirements from which we have conceptualised a formal specification of the incident management domain. Following the specification we have developed a prototype domain model and evaluated it by using state-of-the-art planning engines. We have shown that it is possible to solve "common-size" problems in a reasonable time (up to a few seconds) and in decent quality (short 'make-span'), so the preliminary results indicated a promising direction of our work. We have identified several challenges, most of them related to uncertainty issues, and showed how to deal with most of them without the necessity to extend our model as it stands to support more expressive features. On the other hand, in situation with higher level of uncertainty it might be useful to exploit probabilistic planning (state-of-the-art probabilistic planning engines accept domain and problem specification in RDDL (Sanner 2011)). Developing an RDDL domain model will be our future work.

This work being intended as part of a larger system, we plan to integrate our domain model (and some planning engines) into a planning and execution simulation framework. Then we plan to test it on some real cases in order to determine how plans retrieved by planning engines using our domain model differ from plans provided by mission coordinators. We believe that it will provide us with further insights that will help us to develop and deploy the system.

Acknowledgements

This work is funded by University Research Fund of University of Huddersfield under contract number URF2014_17.

References

Ai-Chang, M.; Bresina, J. L.; Charest, L.; Chase, A.; Hsu, J. C.; Jónsson, A. K.; Kanefsky, B.; Morris, P. H.; Rajan, K.; Yglesias, J.; Chafin, B. G.; Dias, W. C.; and Maldague, P. F. 2004. MAPGEN: mixed-initiative planning and scheduling

for the mars exploration rover mission. *IEEE Intelligent Systems* 19(1):8–12.

Akin, H. L.; Ito, N.; Jacoff, A.; Kleiner, A.; Pellenz, J.; and Visser, A. 2013. Robocup rescue robot and simulation leagues. *AI Magazine* 34(1):78–86.

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the Twenty-second International Conference on Automated Planning and Scheduling (ICAPS-12)*.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10)*.

Dias, P. S.; Gomes, R. M. F.; Pinto, J.; Gonçalves, G. M.; de Sousa, J. B.; and Pereira, F. L. 2006. Mission planning and specification in the neptus framework. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA*, 3220–3225.

Doherty, P., and Heintz, F. 2011. A delegation-based cooperative robotic framework. In *2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011, Karon Beach, Thailand, December 7-11, 2011*, 2955–2962.

Fox, M., and Long, D. 2003. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *J. Artif. Intell. Res.(JAIR)* 20:61–124.

George, J.; Sujit, P. B.; and Sousa, J. B. 2011. Search strategies for multiple UAV search and destroy missions. *Journal of Intelligent and Robotic Systems* 61(1-4):355–367.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in lpg. *Journal of Artificial Intelligence Research (JAIR)* 20:239–290.

Kruijff, G. M.; Kruijff-Korbayová, I.; Keshavdas, S.; Larochelle, B.; Janícek, M.; Colas, F.; Liu, M.; Pomerleau, F.; Siegwart, R.; Neerincx, M. A.; Looije, R.; Smets, N. J. J. M.; Mioch, T.; van Diggelen, J. V.; Pirri, F.; Gianni, M.; Ferri, F.; Menna, M.; Worst, R.; Linder, T.; Tretyakov, V.; Surmann, H.; Svoboda, T.; Reinstein, M.; Zimmermann, K.; Petrícek, T.; and Hlavác, V. 2014. Designing, developing, and deploying systems to support human-robot teams in disaster response. *Advanced Robotics* 28(23):1547–1570.

Ozbay, K.; Iyigun, C.; Baykal-Gursoy, M.; and Xiao, W. 2013. Probabilistic programming models for traffic incident management operations planning. *Annals OR* 203(1):389–406.

Parkinson, S.; Longstaff, A.; and Fletcher, S. 2014. Automated planning to minimise uncertainty of machine tool calibration. *Engineering Applications of Artificial Intelligence* 30:63–72.

Rajan, K., and Py, F. 2012. T-rex: partitioned inference for auv mission control. *Further advances in unmanned marine vehicles. The Institution of Engineering and Technology (IET)*.

Sanner, S. 2011. Relational dynamic influence diagram language (RDDL): Language description. Technical report, NICTA and the Australian National University.

Sato, M.; Muraoka, K.; and Hozumi, K. 2014. Flight control design and demonstration of unmanned airplane for radiation monitoring system. In *19th World Congress of The International Federation of Automatic Control (IFAC)*.

Shah, M. M.; Chrpa, L.; Kitchin, D.; McCluskey, T. L.; and Vallati, M. 2013. Exploring knowledge engineering strategies in designing and modelling a road traffic accident management domain. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence (IJCAI)*, 2373–2379. AAAI Press.

Smith, S. F.; Hildum, D. W.; and Crimm, D. R. 2005. Comirem: An intelligent form for resource management. *IEEE Intelligent Systems* 20(2):16–24.

Vidal, V. 2014. Yahsp3 and yahsp3-mt in the 8th international planning competition. In *Proceedings of the 8th International Planning Competition (IPC-2014)*.