Decision-Theoretic Robotic Surveillance

ILLC Dissertation Series 2002-01

Decision-Theoretic Robotic Surveillance

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.mr. P.F. van der Heijden
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Aula der Universiteit
op vrijdag 25 januari 2002, te 10.00 uur

door

Nikolaos Alexiou Massios

geboren te Athene, Griekenland.

Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam

Cover design by the author.
Typeset in LaTeX.
Printed and bound by Print Partners Ipskamp.

ISBN: 90–5776–078–9

Στους γονείς μου γιατί
ήταν πάντα μαζί μου

Contents

Acknowledgments

First and foremost I would like to express my gratitude to Leo Dorst and Frans Voorbraak. This thesis is mainly the result of the weekly interaction I had with them and the expert and wise guidance they offered to me over the last 4 years. I am greatly indebted to them.

Many thanks should also go to Michiel van Lambalgen and Frans Groen for their valuable help. They were always eager to listen to what I had to say and their questions often guided me through my research. Moreover, their final reading and comments greatly improved the readability of this thesis.

My officemates Stephan ten Hagen and Joris Portegies-Zwart helped me with some latex related problems and with the translation of the Dutch summary. Marco Vervoort supplied the excellent ILLC dissertation series style-file for latex and Cesar Garita gave me a CorelDRAW template which helped in making the cover of this thesis. I thank them, therefore, for their kind contributions.

Finally, there are a lot more people that should be thanked for moral support of varying degrees and in various periods throughout my work. I find it hard to thank everyone at the right amount and still be fair but I hope that my every day attitude should make my feelings clear. An exception will be made for my parents and sister who always stood by me, σας ευχαριστώ μεγάλοι και μικρούλα!

Amsterdam   Nikos Massios
December 2001.

Chapter 1

Introduction

Surveillance can be informally defined as “a close watch kept over something or someone with the purpose of detecting the occurrence of some relevant events”. The person or thing to be watched should be understood to include humans, animals, areas, places, parts of aerospace, et cetera. What events are considered relevant depends on the type of surveillance and the reason why we do it.

Most surveillance tasks are very mundane and automating them is highly desirable. Automated surveillance typically concerns itself with detection and recognition of relevant events. A system of CCTV cameras, microphones or smoke detectors can be used to detect interesting events or to analyse patterns of behaviour. Most of the existing work on robotic surveillance is on recognition of relevant events.

However, event recognition is only one aspect of surveillance. Another aspect, and the concern of this thesis, is that of selecting the right places to focus the surveillance attention on. A solution to this is essential in situations where using fixed surveillance sensors is either not possible, or not practical, or where mobile platforms are more socially acceptable since people then know they are being observed. Further, a solution to the problem of selecting the optimal places to focus attention on can be important not only to robots but also to human security guards.

Commonly, the reason for focusing attention on one area is that relevant events are more likely to be detected there either because they are intrinsically more likely to occur, or because the area has not been observed for a while - increasing the chance that something is there. Because the robot is not omnipresent, this becomes a problem of limited resource allocation. With enough knowledge of the structure and the parameters affecting the decisions, this allocation problem can be automated.

1.1 Surveillance as a planning task

By reviewing surveillance in section 2.1, three main capacities in which an artificial or human agent can be involved in a surveillance task are identified. They are those of:

coordinator An agent that determines or constrains which resources can be used for the surveillance task. The coordinator of a surveillance task may decide on the available budget, on the types of sensors to be used, on its relative importance compared with other tasks, et cetera.

planner An agent that decides how to use the available resources (sensors and actuators) to perform the surveillance task. Typically, a planner has a clear goal like exploration or cost minimisation. It should be seen as something much more task-specific than a coordinator.

sensor An agent that gathers information and transmits it to the planner. A security camera is an example of sensor.

One way a robot can be used in surveillance is in the capacity of a (flexible) sensor. In that case, the planning of the surveillance task would be typically done by a human operator. An example is a security guard teleoperating a robot that inspects a hazardous environment.

However, it is the next level that concerns us here. The subject of this thesis is *the investigation of autonomous surveillance planning for an office-like environment*. The main concern is to develop strategies or algorithms that move the robot in a way that the expected cost of relevant events remaining undetected is minimised. To make the discussion simpler we will focus on one type of relevant events, namely that of fires. A detailed justification of these choices is given in the next chapter.

1.2 Approach in this thesis

Humans can perform surveillance tasks quite well most of the time. However, the process that makes this possible is not immediately clear, which makes it hard to replicate it and hard to assess its optimality. In the case where the probabilities and costs are known, we would like to get a structured understanding based on an analysis of surveillance as a probabilistic decision process. We are eventually interested in an algorithmic implementation of such a decision process.

The detailed exposition of the surveillance problem in the rest of the thesis will make it clear that instances of it for simple environments and simple events are still computationally very hard. So it is not clear how a robot should compute solutions to such instances if results are to be found in a reasonable time.

Approximation is a possible solution to the computational issues. However, approximations, in turn, have the problem of departing from optimal. A difficult balancing act exists between the need to approximate and the computational need to get an approximation that is within some reasonable bound to the optimal.

The approximations proposed in this thesis are based on abstracted representations of the environment. Eventually, decisions are taken among different routes of the abstract environment nodes. A mix of theoretic and simulation experiments are presented to support our discussion.

1.3 Thesis overview

This thesis is divided into the following chapters:

Chapter 2 - Issues in Robotic Surveillance. An overview of current and past research in robotic surveillance is given to provide some context. It should become clear from the discussion that most existing work on robotic surveillance addresses event recognition and that not much is on planning. On the basis of this review the decision was taken to concentrate on surveillance planning. In the rest of this chapter some surveillance planning-related issues which affect our work are mentioned.

Chapter 3 - Problem Formalisation and Complexity. The specific problem of fire surveillance is set using several formalisms like (PO)MDPs or decision theory. These formalisms are shown to be equivalent and conveying the exponential nature of surveillance planning viewed as an optimal search problem with the aim of minimising the expected cost of fires. Because of its exponential nature, simple n-step strategies cannot solve this problem optimally, and consequently approximation methods for the surveillance planning problem become a priority.

Chapter 4 - Hierarchy and Strategies. A “cost barrier” problem defined on a specific office-like building is presented as an example of what a simple n-step look-ahead strategy cannot solve. Then a hierarchical abstraction of this environment is given along with a first *ad-hoc* expected cost approximation, which does not solve the problem in all circumstances, but demonstrates the promise of abstraction.

Chapter 5 - Path-based Clustering. This chapter deals with abstracting in a more principled manner. A much better assignment of the abstracted expected costs can be produced if the geometry of the environment is considered in more detail. After discussing some general desiderata for clustering an office building, we concentrate on the specific case of the corridor-based office building containing a “cost barrier”. A better method for assigning expected costs is produced which differentiates between different types of routes that visit abstract nodes.

Chapter 6 - Path-based Decisions. Given the abstraction, a decision procedure for choosing between clusters is necessary. Several decisions concerning the look-ahead and the type of decisions needed are discussed. Then the fixed cluster route strategy which works on the route-generated abstraction of the environment is presented. It is shown to work well in most cases and to have some robustness with reference to abstraction choices.

Chapter 7 - Conclusions. This chapter briefly draws general conclusions from the specifics of the other chapters.

Chapter 2

Issues in Surveillance

Several robotic surveillance projects exist worldwide and a multitude of different aspects have been and are currently being researched. Despite some overlap in the scope of some projects, there are features that are quite unique in others. In this chapter, we give an overview of current and past research in robotic surveillance to provide the context for our work. It should become clear from the discussion that only few projects have been concerned with surveillance planning and that most existing work addresses event recognition.

After mentioning the research and assumptions of others, we proceed to make explicit what kind of surveillance we are considering. Then some key issues in surveillance planning, which have been identified from the literature review, are listed together with the design choices made in relation to each of these issues.

2.1 Robotic surveillance projects

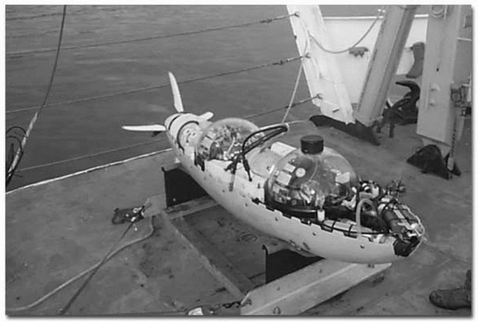
Some robotic surveillance research projects are first listed and then classified. The aim here is to give an idea of the current status of research, rather than completely describe everything that has been done. The classification of the listed projects at the end of the section (see table 2.1) aims to help the reader assess their merits and weaknesses.

2.1.1 List of surveillance projects

AUVs. The goal there is to produce a fleet of small, inexpensive, autonomous underwater vehicles (AUVs) that will roam the earth's oceans [Fri94]. Oceanographers often want simultaneous measurements in many places. These measurements are necessary in building and testing models of the dynamics of the earth's oceans. Currently, such measurements have to be taken on-board ships and it is normally prohibitively expensive to use enough ships to collect data at the required density. A great number of inexpensive AUVs can reduce the cost of collecting such data [Bel97].

However, certain planning problems arise from the fact that these robots have to be capable of performing autonomous navigation inside the sea. For instance, the robot has to be able to avoid underwater obstacles, yet collect the required measurements. An easy way of avoiding underwater obstacles is to get the AUV to rise to the surface. However, rough seas can also be a threat and some sort of compromise has to be made. Furthermore, since the energy of the robot is limited, an optimal survey resolution and time constraint parameters have to be found. This must be done so that the survey takes the minimum amount of energy possible [BW96, WYBM96].

Moreover, there are plans to use "distributed intelligence" by making the robots exchange information about their position and status [TT98]. To do this, underwater communications and robot control mechanisms have to be used [KML+96]. Apart from collecting measurements, other applications suggested include surveillance of fisheries, detection of toxic/nuclear dumping and coast guard support.


(a) Inside view

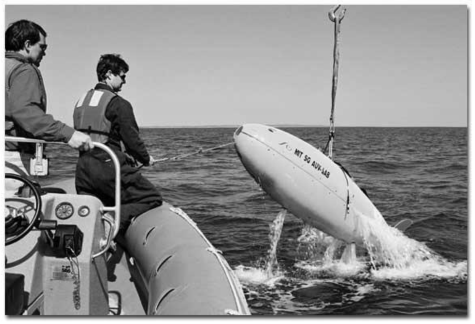
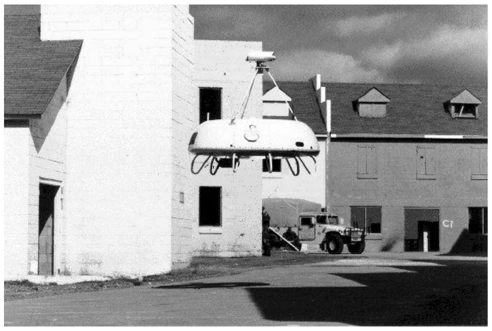

(b) Relative size

Figure 2.1: An AUV (Autonomous Underwater Vehicle).

NRaD, MSSMP. This is a project of the US Navy Research and Development agency (NRaD). The MSSMP [MBB+97] acronym stands for Multipurpose Security and Surveillance Mission Platform. It involves using Cypher, a VTOL (Vertical Take-Off and Landing) robotic vehicle developed by Sikorsky, to perform airborne surveillance. Essentially, Cypher is used as a flying tele-operated robot. It carries a visible and an infrared light camera, a laser range finder and some audio sensors. The robot is controlled by an operator, who can send the robot inside a war zone and use it for locating enemy positions or convoys. Moreover, it can be used to control the perimeter of bases and to inspect difficult terrains (e.g. villages and rooftops) for enemy troops. The US Army calls this system autonomous because there is no tether connecting the operator with the robot. However, we believe that a truly autonomous system is one that does not need an operator. There is some on-board processing performed on the robot to reduce the amount of information needed to be sent over the radio link between the robot and the operator. This on-board processing is limited, however, to the detection of events of interest.

It should also be mentioned here that there is a number of related army projects which aim at producing UAVs (Unmanned Airborne Vehicles). Those vehicles also do not require a tether and their navigation relies on tele-operation and GPS waypoints. Their range is much larger, up to a little over 900 km for some of them. These vehicles communicate with a control centre using satellite links and they are capable of providing their operator with real-time video links. UAVs were used in Bosnia, Iraq and Afghanistan for reconnaissance missions. They have also been used as airborne sensors for fire detection and even surveillance. Predator, Dark Star, and Pioneer are the names of some of the more well-known models. The differences amongst them are mainly related to the type of sensor payload they carry and the altitude they fly.


(a) Cypher

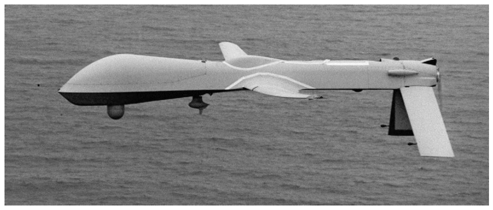

(b) Predator

Figure 2.2: Two UAVs (Unmanned Airborne Vehicles).

Cyberguard. Cyberguard is a mobile security robot marketed by a company called Cybermotion. The aim of Cyberguard is to patrol indoor building areas for intruders, fires, and chemical threats. The system consists of an autonomous robot, an autocharger docking station and an array of software that allows the user to control the robot over a secure digital radio link using a PC. The robot is equipped with a great variety of sensors that allow it to navigate and detect events of interest.

The robot navigation relies partly on sonar sensors and partly on odometry. The robot localises itself using walls, halls and other existing building features and so no artificial landmarks are needed. The path to be followed is predefined by its operator using a CAD-based package and a building floor plan. The path representation consists of path lines and specific locations where the robot has to perform specific actions. There is a separate path-checking program that checks all paths and interacts with the programmer to make sure that all programs are acceptable. Fuzzy logic is used to assess the degree of certainty of detected hazards and to assure appropriate response. The robot can operate continuously for over 12 hours between 3-hour battery charges.

Cyberguards have been sold to academic, industry, and military customers. NRaD used a Cyberguard in a joint army-navy effort to provide automated intrusion detection and inventory assessment capabilities for US DoD warehouses and storage sites. For the inventory assessment task NRaD uses RF transponder tags that are attached to items of interest and hence used to track the location/presence of those items [RLGIG99].

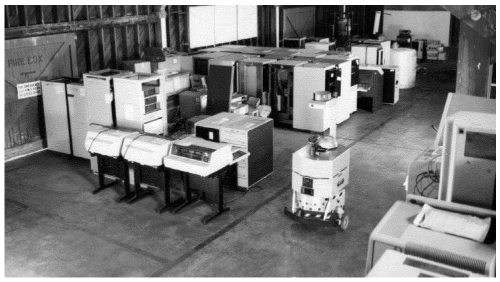


Figure 2.3: Cyberguard patrolling a warehouse.

Visual Surveillance. While a lot of work has been done in visual surveillance covering different aspects and tasks, the topic is still open since none of the systems proposed is completely reliable. The problem can be divided into recognition of objects, tracking of objects/persons and analysis of behaviours.

We mention a particularly representative project on visual surveillance. VSAM (Video Surveillance And Monitoring) is a consortium project sponsored by DARPA. The VSAM consortium consisted of about 15 cooperating US universities and its efforts focus on fusion of information from a large number of surveillance devices to provide surveillance in areas where using humans is too dangerous, costly or impractical.

In demonstrations, cameras distributed over a campus were used to track cars or humans [CLK$^+$00]. The fusion problem was quite hard because many cameras of different types (e.g colour and black and white) and possibly also moving (e.g. with a pan and tilt mechanism) were used. Further, fusion of information coming from UAVs together with information collected on the ground has been discussed [KCL$^+$97, CLK$^+$00]. One of the proposals of this project is that the cameras should rearrange their orientation so that a maximal area of the environment in which they operate is covered. This re-orientation issue seems to be related to the Art Gallery theorems and algorithms [O'R87]. In the Art Gallery problem the position and number of cameras in a gallery is to be decided to ensure that no area of the physical space is left uncovered. In VSAM, movable cameras or many static ones can be used to survey large areas of a campus.

Several vision-related issues have been discussed within this framework like: how calibration of the data coming from different sources should be done [LRS00], how objects should be tracked in the calibrated data [CD00, HHD00], and how sensors on moving robots could be considered [BK01].

Apart from the simple lower-level tracking, higher-level skills like identification of activities of humans or of cars were an issue. The target behaviours were things like "person traversing parking lot", "person meeting other person and walking away", "car traversing parking lot" etc. For the identification of this kind of activities, more commonly, static cameras (often a single one) were used, instead of many moving ones, but the assumption was made that in the future the low-level tracking would be sufficiently well solved to make this irrelevant. The models and techniques used to describe the types of possible human behaviours varied; [ORP00] used Coupled Hidden Markov Models (CHMMs), [BK00] used Hidden Markov Models (HMMs) and Entropy Maximisation, while [IB00] used probabilistic grammar-based descriptions of possible behaviours. In fact, more work has been done in other projects on this kind of behaviour identification and in [Tes97] Petri-nets were used while in [BG95] Bayesian Networks were the model of choice.

Highways. Traffic surveillance has also been considered. A type of visual traffic surveillance is that of analysing behaviours of cars, like "car overtaking other car" or "car exiting roundabout", just by static cameras being placed around a roundabout [BG95]

Another type of traffic surveillance is that of trying to determine the origin/destination counts of cars travelling on the highway system [HR98, PROR99]. For this kind of work cameras are placed in several places of the network (mainly entrances and exits). Various features of the cars, like colour parameters, approximate size, velocity of travel, lane occupied etc.,

Project name	Event type	Agent	Sensor type	Sensor range	Structured environ.	Surveillance plan	Surveillance level
AUVs	simple	multi	multi	variable	no	no	sensor
MSSMP	interm	multi	multi	variable	no	no	sensor
Cyberguard	simple	single	multi	variable	yes	no	sensor
Tracking	interm	multi	multi	variable	no	some	sensor
Behaviour	hard	single	single	static	yes	no	sensor
Traffic	interm	single	multi	static	yes	no	sensor
Rescue	simple	multi	multi	variable	yes/no	yes	coord.

Table 2.1: Classification of surveillance approaches

can be identified by the cameras. Then a probabilistic system can be used to make the association between vehicles currently observed and vehicles previously observed by other cameras. A solution to this problem can lead to better highway design and to automatic identification of accidents and breakdowns in parts of the network not covered by the cameras.

Search and Rescue. RoboCup-Rescue deals with the search and rescue operations after an earthquake [TKT$^+$00]. A RoboCup competition was initiated in order to start an international cooperation between universities in applying robotics technology to the search and rescue problem. The idea is how to provide means for up to 1000 rescue agents to cooperate with each other. Within this framework many heterogeneous types of agents were used: fire fighting agents, police agents, road clearing agents, ambulance agents etc. All these agents are given very incomplete information on what the current state of affairs is and so cooperation and communication are of main concern. The behaviour planning problem is extremely complex and has widely time-varying objectives. The goal is not to find optimal but rather semi-optimal strategies. Time for decisions is limited and apart from trying to rescue people, prevention of a secondary disaster is a goal. The eventual goal is to be able to use the system to develop and to coordinate real rescue agents and to simulate scenarios of potential disasters. So the planning involved is more of coordinator level surveillance as it was identified in section 1.1. Perhaps it should be mentioned that this project is in a relatively early stage.

2.1.2 Classification of surveillance approaches

Here, the projects mentioned before are classified (Table 2.1) in terms of some of their characteristic properties. In the table presented, each project corresponds to a different row while each property corresponds to a different column. The property values assigned to each project are discrete. It should be mentioned that there are some borderline cases where it is not clear what such a discrete property value assignment should be. In such cases we try to make the most accurate assignment possible based on the information presented in the project descriptions available.

In our table, the "event type" column states the difficulty involved in detecting the basic surveillance events considered by each project. Events whose detection is based directly on sensor data are considered to be simple events, e.g. discrete changes of thermometer readings. Intermediate difficulty events are events detected after some analysis of the sensor data is performed, e.g. moving object detection after computation of optic flow. The triggerings of cliche scenarios, e.g. "car overtaking other car" or "car parking", are considered to be hard events since many simpler events have to be detected before a cliche can be triggered.

The "surveillance plan" column is related to the type of planning the agent performs. In order for a project to qualify for our use of the term "surveillance planning", it has to be able to autonomously select the areas of interest within its environment and move its focus of attention to those areas.

The "agent" column refers to the number of agents used. A project is multi-agent if many agents are used and the information obtained by them is fused to provide a single view of the world. The "sensor type" column is related to the data fusion problem. A project is "multi-sensor" if each of the agents used is equipped with *multiple* types of sensors. For example, a project that uses agents equipped with a sonar and a camera is considered to be a multi-sensor project.

The "sensor range" column essentially refers to whether the sensor range is controllable or not. The sensor range is considered to be "variable" if the sensor can automatically move or adjust itself so that a different view of the environment is obtained. An "environment" is considered to be structured if it is not a "naturally" occurring environment. For example, the AUV robots work inside the sea, so they work in an unstructured environment. On the other hand, Cyberguard operates in a structured environment since its environment is man-made. Finally, by level of surveillance we mean a classification into "sensor", "planner", and "coordinator" along the lines of our definition in section 1.1.

From this table, it can be observed that there has been a lot of work on the fusion of information derived from many agents and/or sensors. A system deployed in the real world has first and foremost to solve its sensor interpretation problem. This is perhaps the reason why so much work exists on this aspect of surveillance. Although sensor interpretation and sensor fusion are hard problems, a lot of progress has been made on these, especially in the last decade. Also, the great variation in most properties indicates that different types of difficulties have been encountered in each project.

Further analysis of research in surveillance suggests that although a lot of work has been done at sensor level surveillance, little progress has been made in the planner and coordinator levels which were identified in section 1.1. This indicates that there is potential for research in the area of surveillance planning.

2.2 Surveillance in this thesis

The main concern of this thesis is to develop strategies which use the available sensors in such a way that interesting events are detected. A mechanism for generating such strategies is an example of planner level surveillance. Therefore, this project fills in a gap left open by previous research into robotic surveillance.

More precisely, this thesis deals with the problem of robot fire surveillance in an office-like building. The fires are assumed to start with a certain probability and to have a certain cost when present. The robot is supplied with a simple virtual sensor that can detect a fire within its sensor range with complete reliability. However, the robot has no way of knowing if a fire is present at a location outside its sensor range. So our problem then becomes one of detecting fires as soon as possible with preference given to expensive fires.

The fires we are considering will be modelled to be independent of each other, i.e. we assume that a fire starting at a specific location cannot increase the probability of a fire starting at another location. Further, our fires are modelled to produce the same cost per time unit. In contrast, real fires usually cause more damage at their beginning before they eventually burn everything and extinguish themselves. These real fires do not produce the same cost per time unit as our fires do.

The geometrical and topological structure of the environment is assumed to be perfectly known. The robot has perfect localisation and a perfect map of the environment to begin with. Also the actions for moving from one location to the next never fail and always take the same amount of time.

Although we are discussing the problem of surveillance for fires, the main aim is not to make a system that looks for fires. The aim is to understand how costs, probabilities, and the topology of an office building interact in order to specify how decisions should be taken. We are interested in how a decision process that can make decisions about probabilities and costs in such a situation can be constructed.

Even though the specifics of the problem are not very realistic, this does not mean that it is not relevant to robotics since several similar problems exist. Leaky pipes in a nuclear power plant, or dirty rooms that need cleaning could replace fire as an event to be detected. It should be seen as a general detection problem. In the next chapter the problem will be formalised to a great level of detail. Further, many of the simplifications made in this thesis can be removed in a straight forward way in future work.

A lot of the results presented will be based on simulations of our situation. There is no real robot moving in a real office building that can be on fire. There are several reasons for this decision. One is that working with a real robot is expensive, especially if the building can catch fire. Another is that working with a real robot is time-consuming since a lot of mechanical problems can delay work. Further, and perhaps more importantly, although some assumptions about the sensors and the localisation are made, it does not mean that they are 100% realistic in the current state of robotics. The aim is to develop decision strategies that could command a robot once these capabilities are developed to a sufficiently high level.

The most important decisions can be summarised as:

Sensor interpretation Perfect sensing is assumed in this thesis. This is rather realistic in that we focus on simple events namely fires.

Sensor fusion The decision was taken to assume that the robot uses a single sensor. Normally robots have multiple sensors; however, we overcome this using a virtual fire sensor.

Indoor structured environment The robot works in a simple indoor environment. Further it has a map of that environment.

2.3 Key issues in surveillance planning

This section discusses some key issues that we have considered in order to achieve the goal of our project, and we indicate on which of them we concentrated our efforts. Several of these issues refer to dimensions along which the difficulty of the surveillance planning task can vary, and we will indicate our preferences concerning choices that can make the surveillance problem manageable. Some simplifying choices have to be made, since one cannot expect to solve the complicated problem of automating surveillance planning in its full generality, especially since this problem had not been extensively studied before. The aim is to clarify what type of surveillance planning we are considering in this thesis.

2.3.1 Comparing surveillance strategies

Since the virtual fire sensor of the robot cannot span the entire area of interest, the robot must somehow decide where to use it first. In all but the most trivial situations, the robot will not be absolutely certain of the sensor location that is going to yield the best chances of the relevant events being detected and classified. A probabilistic surveillance planner is one that explicitly reasons about the uncertain outcomes of sensor-location choices and attempts to make the best possible decisions. Of course, this type of decision problems can be masked by forcing the robot to explore its environment either randomly or systematically. It is also possible to use heuristic-based methods that make decisions on actions which are not necessarily based on probabilities. So the important observation here is that there is a number of possible strategies for surveillance.

This multitude of possible surveillance strategies raises the question of which strategy is best for minimising the expected cost. A comparison between different high-level sensing strategies is, therefore, important. There are many ways in which such a comparison can be made. The first way is to start comparing different strategies at a theoretical level. For instance, it might be possible to prove that in a specific scenario one strategy will be better than another. Such proofs can be extremely useful because their results are more general than the ones obtained through experimentation.

However, being able to prove something largely depends on the assumptions and the scenario under consideration. For most realistic situations theoretical proofs are not possible. In such cases the comparison should be an experimental one. One could simulate an environment and try to decide which strategy performs best in simulation. For this type of experimental comparison, there is still the question of what the criteria are that should be used to compare two strategies.

Our decision on how to compare the strategies was to use a mix of theoretical proofs and experimental simulation-based techniques. For the simulation-based comparison the metric used in comparing is that of the actual strategy cost as it is produced by simulating the strategy.

2.3.2 A priori information

The availability and reliability of pre-supplied (*a priori*) information may vary. For instance, the robot can be supplied with a map of the environment which describes the locations of walls, doors, and other possible landmarks that the robot can use to determine its location in the environment. Or it can be supplied with (approximate) probabilities of interesting events occurring at various locations. One can also imagine a robot *learning* this kind of information. This raises the question of how much prior information is necessary and how much information can or should be learned.

Prior information is usually available only in the form derived from a human expert. Converting the knowledge of a human to a format suitable for automated manipulation is usually very hard. Researchers in the area of expert systems have tried in the past to convert human knowledge into facts and control rules that can be processed by computer. There is experience within Knowledge Based Engineering on how such an undertaking can be handled. However, as workers in this area report, converting this knowledge to facts and rules is non-trivial.

At first sight, having a robot supplied with a map of the environment seems simpler than having a robot which first has to learn such a map. However, as soon as both the prior and sensory information stop being completely reliable, it is not clear what should be done when they disagree. This difficult problem is avoided in the case of a learning robot.

The main advantage of a learning robot is clear: It is capable of operating even if pre-supplied information is unavailable. However, for some applications, it may be impractical to include an extensive learning stage before the robot can be used reliably. Also, some of the necessary information may not be learnable at all within a reasonable amount of time. For example, it is a serious problem to learn the probabilities of relatively exceptional events, such as fire, at different locations from past experiences alone.

We have not dealt with the problem of when to choose learning robots. Only the case where a great deal of reliable prior information is available to the robot was considered, and the aim was to develop strategies for using this information together with sensory information to optimise behaviour. Another case not considered is that where the available prior information is not sufficient or is too unreliable to exactly determine optimal behaviour. In such situations, one would be looking for behaviour that is robust, in the sense that small changes in the available information would result in small changes in the behaviour.

2.3.3 Level of detail

The supplied environment map may have various levels of detail. Some authors [Fab96] use a very coarse cell-like representation. Others, use finer-grained ones such as occupancy grids [Kon96].

A well-built surveillance system should probably dynamically adjust the detail of its world representation to suit the task at hand. For instance, on the one hand, route planning inside a building might require only a very coarse representation in terms of rooms and connections between rooms. A path that connects two rooms in that representation can easily be found using classical graph searching algorithms. On the other hand, trajectory planning and localisation within a room probably needs a much more detailed description so that individual obstacles can be cleared with only a few centimetres to spare. Moreover, even experts use different detail levels when describing their field. So the right environment description simplifies planning and also makes the planning approaches more natural.

Maintaining a correct representation of the environment at various resolution levels might be too hard. So an issue that arises within surveillance and, in fact, within almost any AI application is how much of the world needs to be represented and how this should be done. There has been a lot of work on this by both general AI researchers and roboticists. However, there is still a lot of room for improvement within this area.

We have not attempted to solve this problem in its full generality. However, abstraction-based techniques were developed for handling the interaction between representations at different levels of detail. This was unavoidable and we believe there is little hope of applying a probabilistic surveillance planner to realistic surveillance tasks without this kind of interaction between different levels of detail.

2.3.4 Intelligent and non-intelligent opponents

Another issue in surveillance is that of the potential differences between the methods needed to treat intelligent and non-intelligent opponents. Non-intelligent opponents can be thought to be natural phenomena or accidental events. For example, accidental fires, leaky pipes and slippery floors can be classified as non-intelligent opponents. Intelligent opponents are people and events caused by them. For instance, a fire can be manipulated by an arsonist to make it especially dangerous and deceiving. This type of preplanned fires would be classified in the intelligent opponent category.

One can imagine that the detection methods and surveillance strategies should be affected by the sort of opponents the system is supposed to be dealing with. The system should protect itself against attempts by opponents to modify its characteristics. So opponents should not be capable of changing detection rates, false alarm rates or the outcome of actions to alarms by some sort of intelligent manipulation. Therefore, the mode of operation of an autonomous robot should not be completely transparent to make the system resilient to this type of manipulation attacks by intelligent opponents. If the robot succeeds in hiding its strategy from the intelligent opponent, it is more or less justified to treat the events caused by this intelligent opponent similarly to the events caused by nature.

We have not considered the case of intelligent opponents. Instead, we concentrated our efforts on surveillance of natural phenomena. To be more specific, in the rest of this thesis we are considering stochastic events of known probability that can only occur at specific locations in our environment. We took “fire” as an example of such an event. Although this makes our problem quite specific, different types of natural events can replace that of “fire”. In fact, the “fires” we are considering are not totally realistic since, for example, they do not spread. In section, 3.1 we list the assumptions we are making about the problem, and some of them concern the types of events considered. We believe that the rest of this thesis is sufficiently general to deal with several types of non-intelligent opponents, provided that the opponent events have similar characteristics to the ones listed in section 3.1. For example, one could use the techniques developed here to check for leaky pipes in a nuclear reactor.

2.3.5 Confidence

Finally, an important open problem is how to deal with information about the environment that is acquired at different times. The fact that there was nothing interesting in a room at the point in time when sensor measurements in that room were taken does not exclude the fact that something interesting might happen there after some time.

Temporal aspects like the decrease in confidence in the relevance of sensor measurements, given the dynamic nature of the world, are very interesting. For example, an agent should deduce that there is no need to re-examine a recently examined room but wait instead until some time has passed.

For realistic applications it is not obvious what these rates should be, and they may even vary over time. For example, an aerial robot should be capable of deducing that confidence in the non-presence of bush fires during the summer should decrease more rapidly than it does during the winter. This is because during the summer the woods are dry and, as a result, fires can start and spread more easily.

The decision was to use probabilities to model the probabilistic belief that a fire is present some time after a sensor measurement is taken. This decision will be justified further in section 3.2.

2.3.6 Problem complexity

The issues of this section are mentioned because they can determine the style and complexity of surveillance planning approaches. Therefore, the comparison of these approaches depends on the choices made in these issues. The simplest possible scenario is that of a system with a lot of *a priori* information and with a single scale world representation which ignores confidence aspects and deals with non-intelligent opponents. The hardest scenario is probably that of a system that does not make any assumptions about the *a priori* information available, has a true multi-scale map, and at the same time, takes into account confidence aspects in a sophisticated manner and deals with intelligent opponents.

2.4 Other work on surveillance planning

As we have mentioned there has been little work on surveillance planning. There is only one notable exception we are aware of. It is Fabiani's PhD thesis which deals with several issues of uncertain decision making but surveillance is taken as an application area for these methods. Fabiani's application is the selection of areas to be inspected for possible bush-fires, using an autonomous flying robot [Fab96]. This project takes some temporal reasoning aspects into consideration.

His environment consists of a finite set of cells and it is assumed that at each moment the sensor range coincides with a single cell. All locations are thought to be immediately accessible from each other. The dynamics at each cell is modelled by a Markov process. The probabilities of fires starting are assumed to be independent of the state of the neighbouring cells.

The main theoretical improvement is that the confidence with which something is believed in his system decreases over time. Such beliefs originate from sensor measurements. The decrease in confidence of belief can be explained by taking into account the fact that changes occur in the environment while it is not observed. A strategy for surveillance based on this notion of confidence, as well as several others were proposed in his thesis and in the next chapter we will be discussing some of them.

Fabiani has also worked on tracking a partially predictable target using a moving, tracking robot [FGBLL01]. The target has some movement capabilities that set some constraints on how far it can move. Target visibility and position uncertainty are concurrently considered. Both the uncertainty in the target and tracker position are modelled. The tracker can reduce the uncertainty in its position by observing predetermined landmarks in the environment. A game-theoretic framework is applied to decide what actions the tracker should take to guarantee that it keeps the target within its field of view, while managing its position uncertainty. This is a different kind of surveillance problem because the target robot is moving and is evading its tracker. The events considered in this thesis are simpler than those in the target-tracker problem.

2.5 Summary

Although robots seem to be excellently suited for automating the process of moving surveillance sensors, most of the existing projects on robotic surveillance use robots as a kind of flexible sensor and let human operators control the high-level decision-making on where to go to next. So far, relatively little attention has been paid to the problem of automating the planning of available resources to optimally use the sensors for the surveillance task.

As a result of this observation, we decided to develop and evaluate surveillance strategies that can be used in an autonomous surveillance robot. The goal is a strategy producing surveillance plans of approximately minimum cost. The type of problem solved in this thesis would be classified according to the scheme of table 2.1 as: simple event, single agent, single sensor, variable sensor range, structured environment, planning-based and aiming at the planning level. Further, the emphasis is on stochastic independent events, and we shall take ‘fire’ as an example of such an event.

Chapter 3

Problem Formalisation and Complexity

As we have indicated in the previous chapter, there exist various robot surveillance problems of several different difficulties. Our thesis is focused on the specific problem of a robot surveying an office-like building so that the expected cost of fires is minimised. The aims of this chapter are: (a) to formalise this problem to a degree of specificity that allows for it to be tackled, (b) to demonstrate that the various ways in which it is formalised in this chapter are equivalent, and (c) to show that approximations are necessary and that existing approximate methods cannot be used to solve it.

Problems involving expected costs and probabilities are commonly set in a decision-theoretic way. In such a setting the various costs and probabilities, as well as the goal of the surveillance, have to be specified. In robotics decision-theoretic problems are often described in a Markov Decision Process (MDP) setting or in a Partially Observable Markov Decision Process (POMDP) setting. (PO)MDPs are a general framework in which decision-theoretic problems can be specified. Their use has become popular in robotics and other areas because of the existence of several standard solution methods for problems set in this way.

We set our surveillance problem both in the general decision-theoretic way and in several (PO)MDP-based ways. Setting the problem in several ways helps us understand it better by examining its various representations. We shall demonstrate that all these settings are equivalent.

The goal in all alternative settings of our problem is that of minimising the expected cost of fire, but in general, a combination of costs (such as fire damage to the building and fire damage to robot) could be minimised. We believe that the decision-theoretic view on surveillance (including the (PO)MDP settings) is sufficiently general to incorporate, at least theoretically, many of the possible refinements of the problem presented in this chapter.

Although standard methods exist for solving problems set as (PO)MDPs, we shall demonstrate that our problem is too hard for conventional (PO)MDP-solving methods. In fact, any method attempting to solve our problem exactly would fail because its state space grows exponentially in the number of rooms present in our environment. The (PO)MDPs settings will not be pursued in subsequent chapters. Yet, this thesis would not be complete without discussing the possibility of using (PO)MDP-solving methods, and the reason why we think no extra benefit can be gained in this representation.

3.1 Problem assumptions

As in most problems, some assumptions are necessarily made in order to formalise robot fire-surveillance. They describe what the capabilities of our robot are, how time evolves, what the nature of the fires we are trying to extinguish is, etc. Sections 3.2 and 3.3 elaborate on our assumptions by setting the problem and discussing some of the issues related to them. The fact that our state space grows exponentially (as it will be shown in section 3.4) indicates that our problem is not oversimplified by the assumptions made here.

- **Environment structure**. The environment is discretised by assuming that there exists a finite partition of it into rooms. The robot is assumed to know the structure of this partitioning.
- **Deterministic localisation**. The robot always knows where it is.
- **Deterministic movement**. The robot always succeeds in going to a new location.
- **Deterministic sensing**. The robot always knows if its current location is on fire or not.
- **Sensor range**. The sensor range is limited to the current robot location.
- **Sensor type**. It is assumed that only one virtual sensor is present that can be the result of data fusion.
- **Time and actions**. We assume time to be discrete, and we assume that at any time, all the robot's possible actions considered are those that may change the sensor range.
- **Time to move**. The time to move between any two locations is the same and corresponds to one time-step.
- **Stopping of fires**. Fires do not stop spontaneously (burn out).
- **Independence of locations**. Fires do not spread, which implies that the probability of fire at a specific location is independent of other locations.

- **Probabilities**. The robot is assumed to know the exact fire starting probabilities.
- **Costs**. The robot is to know the exact cost a fire causes per time-step. It is assumed that this cost per time-step is the same for the duration of the fire.
- **Start**. For simplicity, we assume that the surveillance starts in a fire-free environment. The robot starts from a specific known location.

3.2 Decision-theoretic setting

In this section we first set the problem in a decision-theoretic way. We then discuss some simple, abstract examples of surveillance problems and show that in these examples the minimum expected cost criterion leads to reasonable surveillance strategies.

3.2.1 Formal environment model

The formal model of the environment that is used for describing the surveillance task is:

3.2.1. Definition. An environment E is a tuple $\langle X, A_0, A, F, C, P_0, P\rangle$, where:

- X is a set $\{X_i : 1 \leq i \leq n\}$ of mutually disjoint spatial areas, or locations,
- $A_0 \in X$ is the start location of the robot,
- $A \subseteq X \times X$ represents the relation of immediate accessibility (for the robot) between locations,
- $F = \{f_i : 1 \leq i \leq n\}$ is a set of Boolean variables indicating if a fire is present at each location X_i. If a fire is present, we write $f_i = 1$ or $\mathbf{f_i}$,
- C is a function assigning to each location $X_i \in X$ the cost $C(\mathbf{f_i})$ associated with not detecting a fire present at X_i,
- P_0 is a function assigning to each location $X_i \in X$ the probability $P_0(\mathbf{f_i})$ that at time 0 an event occurs at X_i,
- and P is a set of transition probabilities. That is, for every $X_i \in X$ and time $t \geq 0$, P contains $P(f_i \rightarrow 1)$ denoting the (prior) probability that a fire starts at X_i during a time-step, and $P(f_i \rightarrow 0)$ denoting the (prior) probability that the fire at X_i stops during a time-step.

In section 3.1 we stated that the robot starts working in a fire-free environment. This means that $P_0(\mathbf{f_i}) = 0$ for every $X_i \in X$. As a consequence, we can simplify the description of an environment $\langle X, A_0, A, F, C, P_0, P\rangle$ to $\langle X, A_0, A, F, C, P\rangle$. Further, we have assumed that $P(f_i \rightarrow 1)$ and $P(f_i \rightarrow 0)$ do not depend on the time t. We also assume that the environment is *connected*, in the sense that for every $X_i, X_j \in X$ there is a path $X_i = Y_1, \ldots, Y_m = X_j$ such that $\langle Y_k, Y_{k+1}\rangle \in A$.

As mentioned in section 3.1, we assume that the sensor range is one environment location and we write r_t to denote the sensor range of the robot at time t. We assume that the sensor range at time $t = 0$ coincides with the robot's start location, so $r_0 = A_0$. The set of reachable states is affected by the immediate accessibility relation so for every $t \geq 1$, $r_t \in \{X_i : \langle r_{t-1}, X_i\rangle \in A\}$. If $r_t = X_i$, then we say that X_i is visited at time t. The decision strategies should decide which immediately accessible location to visit next. For the moment, we do not take recognition uncertainty into account and assume a fire to be detected whenever it is in the sensor range of the robot. However, this type of uncertainty can be easily introduced.

The above definition provides an abstract model of the decision problem. For realistic applications, we have to take into account that a robot can have several sensors, each with its own sensor range, that the sensor range is not necessarily an element of X, that the actions of the robot may include changing its location, its orientation, and possibly manipulating aspects of the environment, such as opening a door. We also have to take into account the exact state and dynamics of the environment, the exact position of the robot in the environment, the uncertainty in the recognition of fires, et cetera. We could have captured more realistic surveillance problems by dropping some of our assumptions, but we preferred a simple model so that we could concentrate on the surveillance strategies. In spite of the many simplifying assumptions, the notions formalised above are sufficiently general to capture the abstract environment used in [Fab96] in order to experimentally compare different surveillance strategies.

Of course, not all applications of surveillance are meant to trigger intervening responses to the observed relevant event. For example, observations made in the context of a scientific study are primarily aimed at information gathering, not at intervening. However, when interventions do play a role, their effects should be incorporated in the model of the surveillance problem. Since the particular actions triggered by a detection are not themselves, strictly speaking, part of the surveillance behaviour of the agent, we will leave them out of our considerations.

For simplicity, we have assumed that fires do not stop spontaneously, but immediately after being detected by the surveillance agent. Formally, $P(f_i \rightarrow 0) = 0$, and $P_{t+1}(\mathbf{f_i}) = P(f_i \rightarrow 1)$, if $r_t = X_i$. It would, of course, have been possible to introduce some time delay for the countermeasures to take effect, but this would have raised the problem of deciding how important it is to monitor areas where fires are known to be present or have been observed to occur. It would also have been possible to allow $P(f_i \rightarrow 0) > 0$ and to model the effect of the actions triggered by observing a fire as an increase in $P(f_i \rightarrow 0)$. Our simplifying assumption can be viewed as an extreme instance of this possibility.

As in [Fab96], we assume that the cells are independent, and that the probability of $f_i \rightarrow 1$ is constant over time. It is then possible to express $P_t(\mathbf{f_i})$ in terms of $P(f_i \rightarrow 1)$ and the amount of time that has passed since the last visit to X_i.

3.2.2. Proposition. *Let $E = \langle X, A_0, A, F, C, P\rangle$ be an environment where $P(f_i \rightarrow 0) = 0$, and $r_t = X_i$ implies that $P_{t+1}(\mathbf{f_i}) = P(f_i \rightarrow 1)$. Then $P_t(\mathbf{f_i}) = 1 - (1 - P(f_i \rightarrow 1))^{t-t'}$, where t' is the largest time point $\leq t$ such that $r_{t'} = X_i$.*

Proof.
Prove for $t = t' + 1$. For $t = t' + 1$ using the formula in proposition 3.2.2, we get $P_{t'+1}(\mathbf{f_i}) = 1 - (1 - P(f_i \rightarrow 1))^{t'+1-t'} = P(f_i \rightarrow 1)$. This is correct since $P(f_i \rightarrow 1)$ is the probability of a fire starting in one time-step.
Assume for $t = t' + k$. Assume $P_{t'+k}(\mathbf{f_i}) = 1 - (1 - P(f_i \rightarrow 1))^k$.
Prove for $t = t' + k + 1$. We first compute $P_{t'+k+1}(f_i = 0)$ as the probability that no fire existed at $t = t' + k$ times the probability that a fire did not start during $t = t' + k + 1$. We have $P_{t'+k+1}(f_i = 0) = (1 - P_{t'+k}(f_i = 1))(1 - P(f_i \rightarrow 1)) = (1 - P(f_i \rightarrow 1))^k(1 - P(f_i \rightarrow 1)) = (1 - P(f_i \rightarrow 1))^{k+1}$. We know $P_{t'+k+1}(f_i = 1) = 1 - P_{t'+k+1}(f_i = 0)$. So, our formula also holds for $t = t' + k + 1$ because $P_{t'+k+1}(f_i = 1) = 1 - (1 - P(f_i \rightarrow 1))^{k+1}$. ∎

3.2.2 Surveillance strategies

In [Fab96] a surveillance strategy is proposed based on the newly introduced notion of confidence, which can be viewed as a second-order uncertainty measure. Whenever sensory information about the state of a location becomes available, the probability of an event occurring at that location at that time is updated, and one is assumed to be very confident about this assessment of the state of the location. This confidence then drops gradually over time during a period in which no fresh sensory information concerning this particular location is obtained. The rate by which the confidence decreases depends on the transition probabilities: the more likely the changes, the higher the decrease rate.

Specifically, the factor λ^p is used as confidence decrease rate, where p is the transition probability leaving from the observed state and λ is some unspecified parameter. The actually used computation of confidence is slightly more complicated, due to the fact that some time after the observation it is no longer clear which transition probability ($P(f_i \rightarrow 1)$ or $P(f_i \rightarrow 0)$) should be used in the computation of the decrease rate.

In our model, the situation is simpler, since we assumed that when visiting a location X_i at time t, it either observes that no fire is present at X_i or it immediately extinguishes the fire. In both cases, the robot can be confident that no fire is present at X_i after t. This confidence can decrease over time due to the possibility that a fire starts after t. The rate of this decrease depends, of course, on $P(f_i \to 1)$. The transition probability $P(f_i \to 0)$ does not play any role.

Since the factor $\lambda^{P(f_i \to 1)}$ is meant to be a decrease rate, one can infer that $0 < \lambda < 1$. Every time a location X_i is not visited, the degree of confidence of the robot that no fire is present at X_i is multiplied by $\lambda^{P(f_i \to 1)}$.

3.2.1. Observation. *Let* $0 < \lambda < 1$. *Then* $(\lambda^x)^n > (\lambda^y)^m$ *iff* $nx < my$.

When visiting X_i, the robot can either see that no fire is present, or the robot will immediately extinguish it. In either case, the confidence of the robot that no fire is present at X_i is maximal, say 1. It thus follows from the above observation that the location with the lowest confidence at time t is the location X_i such that $P(f_i \to 1)(t - t')$ is maximal, where t' is the time of the last visit to X_i.

The strategy proposed in [Fab96] can be described as follows.

maximum confidence Choose the action that changes the sensor range to the neighbouring location which has the lowest degree of confidence attached to it.

In [Fab96], this strategy is experimentally compared to the following strategies.

random exploration Randomly choose a location as the next sensor range.

methodical exploration Choose all the locations, one after the other, and always in the same order, as the sensor range at the next moment.

maximum likelihood Choose the action that changes the sensor range to the neighbouring location with maximal uncertainty, where the uncertainty at location X_i is measured by $\min(P(\mathbf{f_i}), 1 - P(\mathbf{f_i}))$

Notice that both random and methodical exploration, as described above, allow choosing non-neighbouring locations. Actually, in the experiments of [Fab96] it is assumed that all locations are directly accessible from each other ($A = X \times X$). This is only realistic in the case when changing attention to a far removed location involves no or only a negligible amount of cost or time and this is not the case in robotic surveillance. Of course, it is not difficult to restrict random exploration to choosing randomly between neighbouring locations only, but it is not clear how to put a similar restriction on methodical exploration.

One possible strategy that can be considered to be a local variant of methodical exploration is the following.

minimax interval Minimise the maximum time interval between visits of locations by choosing the action that changes the sensor range to the neighbouring location which has not been visited for the longest time.

We propose to use this minimax interval strategy as a kind of reference strategy. Since this strategy does not use information about the uncertainties, it can be used to clarify how much other strategies which do use uncertainty information gain in efficiency.

It should be mentioned that in the case of the maximum likelihood strategy many uncertainty measures, including, for example, entropy (which is defined as $\sum_{X_i \in X} P(\mathbf{f_i}) \log(P(\mathbf{f_i})))$, give rise to the same preferences as $\min(P(\mathbf{f_i}), 1-P(\mathbf{f_i}))$. We use this definition of maximum likelihood rather than entropy, because we want to be able to compare our strategies with those of Fabiani [Fab96] and that is the definition used there. The uncertainty is maximal whenever the probability of fire is closest to 0.5. This is because $\min(0.5, 1-0.5) = 0.5$ while for example $\min(0.7, 1-0.7) = 0.3$. As we will see in section 3.2.3, the maximum likelihood strategy seems more appropriate for *symmetrical* surveillance understood as maintaining a maximally correct model of the state of the environment, with respect to both the presence and the absence of relevant events, than for *asymmetrical* surveillance aimed at detecting relevant events.

In [Fab96], no explicit choice is made between such a symmetrical view on surveillance and the asymmetrical view we take. Several criteria are used to evaluate the performance of the strategies in the experiments, including the symmetrical criterion of the percentage of erroneous estimations of the state of each location and the asymmetrical criterion of the percentage of undetected relevant events.

We propose a surveillance strategy based on decision-theoretic considerations. By decision-theoretic surveillance we understand the kind of behaviour guided by the following decision strategy.

minimum expected cost Choose the action that minimises the expected cost.

This decision strategy can be interpreted both globally and locally. Under the global interpretation, the action that has to be chosen corresponds to the behaviour of the surveillance agent from the start to the end of the surveillance task. There is not an inherent end to a surveillance task, but in practice each particular task has a limited duration (say, until the next morning when the employees return to the office building, or until the batteries of the robot have to be recharged).

The (global) expected cost EC_T until time T can be computed by the following formula.

$$EC_T = \sum_{1 \leq t \leq T, X_i \neq r_t} P_t(\mathbf{f_i}) C(\mathbf{f_i}).$$

Notice that a choice to visit location X_i at t not only removes the term $P_t(\mathbf{f_i})C(\mathbf{f_i})$ from the above sum, but also has some indirect benefits, due to the fact that it reduces $P_{t'}(\mathbf{f_i})$ for $t' > t$ and it makes some neighbouring locations of X_i available to be visited at time $t+1$.

The behaviour of the surveillance agent from the start to the end of the surveillance task can also be viewed as consisting of a sequence of simpler actions. One can apply the above decision strategy locally to choose at each time between the possible simple actions by comparing the consequences of these simple actions, or perhaps by comparing the (expected) consequences of small sequences of simple actions. Let us say that an n-step strategy compares the (expected) consequences of sequences of n (simple) actions. Of course, the strategy is more easily implemented for small n, whereas, in general, it better approximates the global strategy for large n.

Notice that none of the strategies considered in [Fab96] takes into account a notion of cost which, for example, allows one to express the opinion that for some areas it is more important to observe relevant events than it is for other areas. Another use of the notion of cost is to express the opinion that early detection is important as it decreases the cost over time after the start of an event.

3.2.3 Examples and first results

We have defined six decision strategies for surveillance, three of which make use of some kind of uncertainty information, namely maximum confidence, maximum likelihood, and minimum expected cost. The following proposition shows that these strategies essentially agree if there is no (relevant) uncertainty information to be used.

3.2.3. Proposition. *Let $E = \langle X, A_0, A, F, C, P\rangle$, and assume that for all locations $X_i, X_j \in X$, $P(f_i \rightarrow 1) = P(f_j \rightarrow 1)$ and $C(\mathbf{f_i}) = C(\mathbf{f_j})$. Then the maximum confidence strategy and the 1-step minimum expected cost strategy both reduce to the minimax interval strategy. Also, for sufficiently small transition probabilities $P(f_i \rightarrow 1)$, the maximum likelihood strategy will agree with the minimax interval strategy.*

Proof. Examining observation 3.2.1 in conjunction with the assumption that $P(f_i \rightarrow 1) = P(f_j \rightarrow 1)$ for locations $X_i, X_j \in X$, we can conclude that an agent choosing the location with the smallest degree of confidence essentially chooses the location that has not been visited for the longest period of time in part of the environment that is reachable in one time-step.

Similarly, under the same assumption, a 1-step minimum expected cost strategy chooses the room with the largest expected cost but since the fire costs are also uniform among rooms, this always corresponds to the room that has not been visited for more time-steps.

In the case of the maximum likelihood strategy for sufficiently small transition probabilities $P(f_i \rightarrow 1)$ and relatively small environment, we have $\min(P(\mathbf{f_i}), 1 - P(\mathbf{f_i})) = P(\mathbf{f_i})$. If that is the case, choosing the action with maximal uncertainty corresponds to using the 1-step minimum expected cost strategy and we have

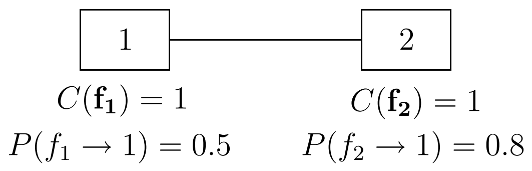


Figure 3.1: The environment of example 3.2.4.

already shown that, under our assumption, this corresponds to the minimax interval strategy. This result supports the choice of using minimax interval as a reference strategy. ∎

It follows that we can expect a difference between the strategies mentioned only in the case of varying probabilities or costs. Our first example illustrates the difference between the various surveillance strategies introduced so far.

3.2.4. Example. Consider an environment $E = \langle X, A_0, A, F, C, P \rangle$, where X is a set $\{X_1, X_2\}$ consisting of two rooms, $A = X \times X$ (fig. 3.1). Assume $C(\mathbf{f_1})$ $= C(\mathbf{f_2}) = 1$, $P(f_1 \rightarrow 1) = 0.5$ and $P(f_2 \rightarrow 1) = 0.8$ (Remember that we also assume that $P(f_i \rightarrow 0) = 0$, and that $P_{t+1}(\mathbf{f_i}) = P(f_i \rightarrow 1)$, if $X_i \subseteq r_t$). In other words, we have two equally important rooms, each accessible from the other, but the probability of a fire starting at room X_2 is higher than the corresponding probability at room X_1.

The strategy based on maximum likelihood will always look at room X_1 (where the uncertainty is maximal), and will never take a look at room X_2. It is maximal for 0.5 because $\min(0.5, 1-0.5) = 0.5$ while for example $\min(0.7, 1-0.7) = 0.3$. The strategies based on methodical exploration and minimax interval go back and forth between both rooms, just as the maximum confidence strategy does, at least if one assumes that the confidence in an observation made at the immediately preceding moment is higher than the confidence in an observation made before that time. This seems to follow from the way the decrease rate of confidence is computed in [Fab96].

The strategy based on a 1-step minimisation of expected cost is slightly more complicated. At time 1, room X_2 is chosen because $P(\mathbf{f_2}) = P(f_2 \rightarrow 1) = 0.8 > 0.5 = P(f_1 \rightarrow 1) = P(\mathbf{f_1})$. At time 2, $P(\mathbf{f_2})$ is again 0.8, since this room was visited at the immediately preceding time-step. However, $P(\mathbf{f_1})$ has only increased to 0.75, which is not enough to get chosen. Only at time 3, $P(\mathbf{f_1}) = 0.875$ has increased above the 0.8 probability that a fire occurs in room 2. We thus obtain a sequence where room X_1 is only chosen every third time-step. See table 3.1, where for the first six time-steps the expected costs of not visiting a room are displayed.

time	$EC(\mathbf{f_1})$	$EC(\mathbf{f_2})$
1	0.5	**0.8**
2	0.75	**0.8**
3	**0.875**	0.8
4	0.5	**0.96**
5	0.75	**0.8**
6	**0.875**	0.8

Table 3.1: The room expected costs of the first six time-steps of example 3.2.4. The minimum expected cost strategy chooses the room with the maximal expected cost (printed in boldface).

In this example, the maximum likelihood strategy does not result in an exhaustive exploration of the environment. Both maximum confidence and minimum expected cost behave better in this respect if we assume that exhaustive exploration is desirable. The problem with the maximum likelihood criterion is that $P(\mathbf{f_i}) > P(\mathbf{f_j})$ is not guaranteed to result in a preference to visit X_i rather than X_j whenever $P(\mathbf{f_i}) > 0.5$. In fact, if $P(\mathbf{f_i}) > P(\mathbf{f_j}) \geq 0.5$, then the criterion prefers X_j. Notice that not only in the artificial example above, but also in practical applications, with sufficiently many locations and sufficiently high transition probabilities, it can happen that $P(\mathbf{f_i}) > 0.5$. However, it should be mentioned that typically the fire starting probabilities are very small and that the environment is not large enough for the probability to grow to 0.5. This means that in those far more normal cases maximum likelihood does correspond to minimising the probability of fire.

Still we believe that there is a theoretic point to be made. Since the maximum likelihood criterion does not prefer locations where the chance of detecting a fire is high, but is more interested in locations where the occurrence of a fire is highly unknown, we conclude that the criterion is more appropriate for symmetrical than for asymmetrical surveillance.

In example 3.2.4, the 1-step minimum expected cost strategy results in a behaviour which seems intuitively appealing, since it clearly reflects the fact that $P(f_1 \rightarrow 1)$ is substantially lower than $P(f_2 \rightarrow 1)$, whereas maximum confidence, just as methodical exploration, treats both rooms the same. However, we will see below that this intuitive appeal may be somewhat misleading.

The maximum confidence strategy does also take the probabilities $P(f_1 \rightarrow 1)$ and $P(f_2 \rightarrow 1)$ into account, since the rate of confidence decrease is a function of these probabilities. However, the decrease rate proposed in [Fab96] does not result in a different treatment of both rooms in the example. Thus one can view the example as an indication that in the minimum expected cost strategy the probabilistic information is used more directly and taken more seriously than in the maximum confidence strategy.

The maximum confidence strategy does not consider at all the possibility that for some areas it may be relatively more important to detect events. This is easily implemented in the minimum expected cost strategy by letting the cost $C(\mathbf{f_i})$ of not detecting a fire depend on the area X_i. Such varying costs may cause a problem, since they may prevent the 1-step minimum expected cost strategy from achieving an exhaustive exploration of the environment.

It can be shown that if the cost of not detecting a fire is constant over the different areas X_i, then, in the long run, the 1-step minimum expected cost strategy will result in an exhaustive exploration of the environment. More precisely, one can show the following:

3.2.5. PROPOSITION. *Let $E = \langle X, A_0, A, F, C, P\rangle$, and assume that for all locations $X_i, X_j \in X$, $C(\mathbf{f_i}) = C(\mathbf{f_j})$. Then, in the long run, every $X_i \in X$ is visited when applying the 1-step minimum expected cost strategy, and there is a finite upper bound N_i on the length of the time interval between visits of X_i.*

Proof. We prove the slightly more general proposition that says that for an environment $E = \langle X, A_0, A, F, C, P\rangle$, with $C(\mathbf{f_i}) = C(\mathbf{f_j})$, $P_0(\mathbf{f_i}) < 1$, $P(f_i \to 1) < 1$ $\forall X_i, X_j \in X$. Then $\forall X_i \in X$ there exists a maximal time N_i between visits of X_i. The proof is by induction to the size of $X = \{X_i : 1 \leq i \leq n\}$:
Prove for size 1. Trivial. The robot is always present in that room.
Assume for size $< n$.
Prove for size n. Let X_k be a location such that the infimum of time between visits is at least as large as for the other locations.

Suppose the assumption step is not true for size n, then $\exists X_m$ such that its infimum time between visits of X_m is ∞. Therefore, for X_k the time between visits is ∞. Note also that by the induction hypothesis there is a finite maximum time N_m that the robot can move in $X - \{X_k\}$, without exhaustively exploring $X - \{X_k\}$.

By the assumption step (for n-1 rooms) we have that there is a bound N_i on the time between visits for all rooms in $X - \{X_k\}$ or in other words: $\forall X_i \in X - \{X_k\}, \exists N_i >$ time between visits of X_i. This implies that there is a bound p_i on the probability that there is a fire in any of the rooms in $X - \{X_k\}$ or in other words: $\forall X_i \in X - \{X_k\}, \exists p_i < 1$ such that $\forall t, P_t(\mathbf{f_i}) < p_i$. So the bound on the time between visits N_i of the inductive hypothesis sets a bound p_i on how much the probability of fire can rise. If the time between visits in room X_k is ∞, then $\exists N, \forall X \in X - \{X_k\}, p_i < P(\mathbf{f_k})$, where $P(\mathbf{f_k})$ is the probability of fire at X_k after not visiting it for N times.

Let $X_j \in X - \{X_k\}$ be a neighbouring location of X_k, then between N and $N + N_m$ steps X_j will be visited. Since for that time $\forall X \in X - \{X_k\}, p_i < P(\mathbf{f_k})$, X_k will be visited at the next time-step. So room X_k is visited after the right amount of time and this completes our induction.

Since a bound in the time between visits at every location exists, one can conclude that eventually every location will be visited. ∎

If fires do not stop spontaneously before they are detected, exhaustive exploration implies that all fires will eventually be detected. But even among strategies that are 100% successful, with respect to eventually detecting fires, there may be a difference in performance if, for example, *early* detection is considered to be important.

Before we can continue with the discussion on exhaustive exploration, we need to know the answer to another interesting question namely, that of if and how the indirect benefits of visiting a location can be taken into account. For example, visiting a location at time t decreases the probability of a fire being present at that location after t, but this effect is not considered by simply comparing the expected cost over n time-steps.

3.2.6. DEFINITION. Let t' be the time of the last visit to X_i before t, and T be the time of the next visit after t. Then the indirect benefits of a visit to X_i at t are equal to the following.

$$\sum_{n=t+1}^{T} (1 - P(f_i \to 1))^{n-t} - (1 - P(f_i \to 1))^{n-t'}.$$

This expression computes for every time-step between t and T two probabilities: (a) the probability of a fire given that room X_i was last visited at time t, and (b) the probability of fire given that room X_i was not visited at time t (so its last time of visit was t'). The sum of the differences between these two probabilities is defined as the indirect benefits of a visit at time t.

If $T = t + 1$, then the above expression provides a lower bound of the indirect benefits of a visit to X_i at t instead of later. Incorporating this amount of the indirect benefit into the 1-step minimum expected cost strategy is similar to employing a 2-step minimum expected cost strategy, and it results in the back and forth behaviour in the environment of example 3.2.4.

3.2.7. PROPOSITION. *Let t' be the time of the last visit to X_i before t. Then the indirect benefits of a visit to X_i at t have the following upper bound.*

$$\lim_{T\to\infty} \sum_{n=t+1}^{T} (1 - P(f_i \to 1))^{n-t} - (1 - P(f_i \to 1))^{n-t'}$$

$$= \sum_{n=1}^{t-t'} (1 - P(f_i \to 1))^{n}.$$

Proof. If we expand the series, we get the following sequence:

$$\lim_{T\to\infty}\sum_{n=t+1}^{T}(1-P(f_i\to 1))^{n-t}-(1-P(f_i\to 1))^{n-t'} =$$

$$\begin{array}{lcll}
=(1-P(f_i\to 1)) & - & (1-P(f_i\to 1))^{t+1-t'} & \textbf{(1)}\\
+(1-P(f_i\to 1))^2 & - & (1-P(f_i\to 1))^{t+2-t'} & \textbf{(2)}\\
+(1-P(f_i\to 1))^3 & - & (1-P(f_i\to 1))^{t+3-t'} & \textbf{(3)}\\
\vdots & \vdots & \vdots & \\
+(1-P(f_i\to 1))^{t+1-t'} & - & (1-P(f_i\to 1))^{t+1-2t'} & \textbf{(4)}\\
+(1-P(f_i\to 1))^{t+2-t'} & - & (1-P(f_i\to 1))^{t+2-2t'} & \textbf{(5)}\\
+(1-P(f_i\to 1))^{t+3-t'} & - & (1-P(f_i\to 1))^{t+2-3t'} & \textbf{(6)}\\
\vdots & \vdots & \vdots &
\end{array}$$

The left part of line (4) cancels out the right part of line (1), the left part of line (5) cancels out the right part of line (2) and so forth. As $T\to\infty$ only the left column up to $t-t'$ remains and therefore the limit is $\sum_{n=1}^{t-t'}(1-P(f_i\to 1))^n$. ∎

We now return to the discussion about exhaustive exploration. The next example is a simple modification of example 3.2.4 and in conjunction with this upper bound can be used to show that, in general, proposition 3.2.5 is no longer valid if the costs are allowed to vary. Consequently, the 1-step minimum expected cost strategy is no longer guaranteed to result in an exhaustive exploration of the environment.

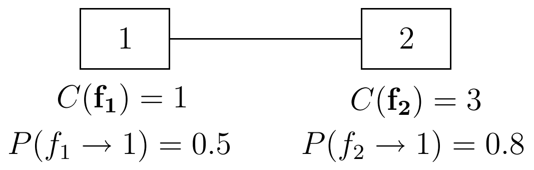


Figure 3.2: The environment of example 3.2.8.

3.2.8. Example. Consider the situation of example 3.2.4, but now assume that $C(\mathbf{f_1})=1$ and $C(\mathbf{f_2})=3$ (fig. 3.2). Then the expected cost of not visiting room X_1 has an upper bound of 1, whereas that of not visiting room X_2 is 2.4, even though it has just been visited. Therefore, by the 1-step minimum expected cost strategy, room X_2 will always be chosen. See table 3.2, where for the first four time-steps the expected costs (of not visiting a room) are displayed.

time	$EC(\mathbf{f_1})$	$EC(\mathbf{f_2})$
1	0.5	**2.4**
2	0.75	**2.4**
3	0.875	**2.4**
4	0.9375	**2.4**

Table 3.2: The room expected costs of the first four time-steps of example 3.2.8. The minimum expected cost strategy chooses the room with the maximal expected cost (printed in boldface).

This effect of ignoring room X_1 can be avoided by allowing the cost of not detecting an event to grow as a function of the time passed since the event started. However, if the mentioned (expected) costs are correct, then this ignoring of room X_1 is defensible. Intuitively, one should not require a surveillance agent to explore irrelevant or unimportant areas of the environment. This is substantiated by the following.

3.2.9. PROPOSITION. *In the environment of example 3.2.8 the 1-step minimum expected cost strategy minimises the global expected cost.*

Proof. To prove this proposition we will show that a visit to room 1 is never justified, so our 1-step minimum expected cost strategy, which only visits room 2, is optimal.

The direct benefit of visiting room 1 is $\lim_{t\to\infty} EC(\mathbf{f_1}) = 1$. By proposition 3.2.7 we know that the upper bound of the indirect benefit of a potential visit to room 1 at time t is $\sum_{n=1}^{t-0}(1-0.5)^n$. The limit of this upper bound, as $t \to \infty$, is also 1. Therefore, a potential visit to room 1 as $t \to \infty$ can bring us a maximal potential benefit of 2 (One util for the direct benefit plus one for the indirect benefit). But this is always less than the 2.4 direct benefit of visiting room 2. So a visit to room 1 is never justified. ∎

Perhaps a more important problem than possibly preventing an exhaustive exploration of the environment is that the varying cost can form an obstacle to obtaining optimal behaviour using the local (1-step) minimum expected cost strategy.

3.2.10. EXAMPLE. Consider $E = \langle X,\ A_0,\ A,\ F,\ C,\ P\rangle$, where X is a set $\{X_1, X_2, X_3\}$ consisting of three rooms (fig. 3.3). The accessibility relation A is given by $\langle X_i, X_j\rangle \in A$ iff i and j differ by at most 1, $C(\mathbf{f_1}) = 10$. $C(\mathbf{f_2}) = 1$, $C(\mathbf{f_3}) = 3$, $P(f_1 \to 1) = 0.1$, $P(f_2 \to 1) = 0.5$ and $P(f_3 \to 1) = 0.8$. In other words, we now have three rooms in a row, and after discarding room X_0, one obtains the environment of example 3.2.8.

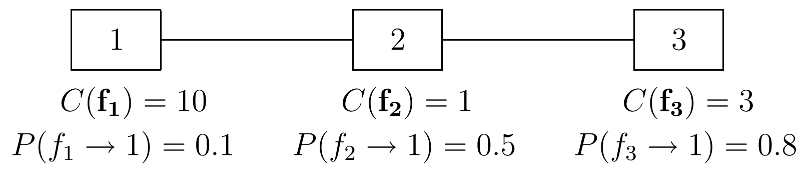


Figure 3.3: The environment of example 3.2.10.

As in example 3.2.8, if $A_0 = X_3$, the 1-step minimum expected cost strategy will always choose room X_3. But now the (possibly justifiable) ignoring of room X_2 will make it impossible to visit room X_1, and, in the long run, the expected cost of not visiting room X_1 will be very high.

Obviously, such problems can be solved theoretically by looking ahead more than one step. However, looking many steps ahead is computationally expensive, since the required computation time depends exponentially in the number of steps considered. In [MV98], it is argued, based on some experiments, that in typical environments it is not feasible for real-time behaviour to use much more than a look-ahead of five steps.

Even for constant costs, the 1-step minimum expected cost strategy is not guaranteed to globally minimise the expected cost.

3.2.11. PROPOSITION. *In the environment of example 3.2.4 the 1-step minimum expected cost strategy does not minimise the global expected cost.*

Proof. In table 3.1 a stable repetitive behaviour can be observed after time-step 4 with a cycle size of 3 (see steps 4-6). The total environment cost for three time-steps of this behaviour is $(0.5 + 0.96) + (0.75 + 0.8) + (0.875 + 0.8) = 4.685$.

A robot moving back and forth between the rooms in the same environment would enter a behaviour consisting of the states reached in time-steps 4 and 5 (a cycle of 2). The cost for two time-steps of this behaviour is $(0.5 + 0.96) + (0.75 + 0.8) = 3.01$.

To make a comparison a cycle of 6 time-steps is examined. The 1-step minimum expected cost strategy gives us a $4.685 \times 2 = 9.37$ utils expected cost while the simple back and forth behaviour gives us a $3.01 \times 3 = 9.03$ utils expected cost. So the 1-step minimum expected cost strategy does not minimise the global expected cost. ∎

Actually, in example 3.2.4, the global expected cost of the 1-step minimum expected cost strategy is higher than that of the back and forth behaviour resulting from the methodical exploration, the maximum likelihood and the minimax

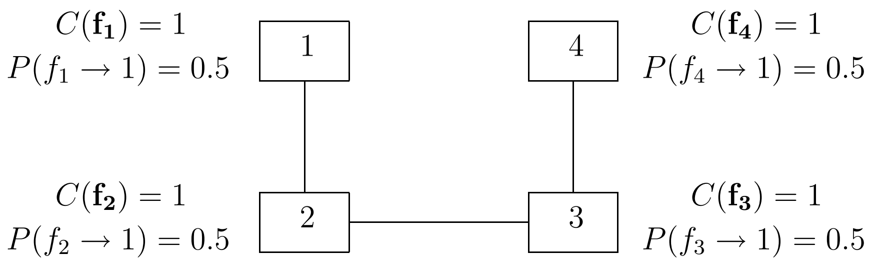


Figure 3.4: The environment of example 3.2.12.

interval strategies. The 2-step minimum expected cost strategy already results in the same back and forth behaviour.

It should be observed that looking ahead more steps does not always result in better performance. The following example shows that sometimes the 1-step minimum expected cost strategy behaves better than the 2-step strategy.

3.2.12. Example. Consider $E = \langle X, A_0, A, F, C, P\rangle$, where X is the set $\{X_1, X_2, X_3, X_4\}$, the accessibility relation A is given by $\langle X_i, X_j\rangle \in A$ iff i and j differ at most 1, and for all locations X_i, $C(\mathbf{f_i}) = 1$ and $P(f_i \to 1) = 0.5$. This environment can model a corridor, with X_1, X_2, X_3, X_4 as sections of the corridor, but it can also model two rooms, represented by X_1, and X_4, connected by a corridor with two sections represented by X_2 and X_3. (See fig. 3.4.) Therefore, this kind of environment is not unusual for an office-like building.

If the agent starts at X_2 or X_3, then the 2-step minimum expected cost strategy results in going back and forth between X_2 and X_3 (without visiting X_1 or X_4), whereas the 1-step strategy results in going back and forth between X_1 and X_4 (and visiting X_2 and X_3 in between). It is easy to see that the latter behaviour is better.

Notice that the above example shows that the 2-step minimum expected cost strategy is not guaranteed to result in an exhaustive exploration of the environment, even if the assumption of constant cost of proposition 3.2.5 is satisfied. The problem is caused by the accessibility relation, since if one additionally assumes universal accessibility ($A = X \times X$), then proposition 3.2.5 generalises to the n-step minimum expected cost strategy.

The subtle role of the accessibility relation is not present in several problems which seem closely related to our robotic surveillance problem. The multi-armed bandit problems are examples of this. There a casino visitor has to select a lever to pull among many bandit machines (fruit machines). There are many variants of this problem but typically the longer a lever has not been pulled the more likely it is to give a big reward. An intelligent visitor should pull a lever that has been used a lot but has not given up a reward for some time. Unlike our problem, the agent may select any bandit and the accessibility relation does play a role. Other similar problems are the problem of maintaining a system with independent component failures, and that of a surveillance camera selecting which part of a scene to focus on. In all of these it seems natural to assume a universal accessibility relation.

The problem with the 2-step strategy in example 3.2.12 can be solved by implementing a commitment to complete the selected path of length 2, and make a new choice only every other time-step. In the concrete, pseudo-realistic example discussed in [MV99b], the 5-step minimum expected cost strategy performs better with commitment than without.

The next example illustrates that a mobile platform may not always be very useful if the relevant events have an insignificant duration.

3.2.13. Example. Consider a robot with a digital photo camera on top of a hill. Assume that it has to take as many pictures of lightning as possible. Further, assume that the camera has a viewing angle of 90 degrees, and that at any time it can be directed in one of four directions: North, East, South, West. Finally, assume that after taking a picture, the robot has sufficient time to change the direction of the camera in any of these four directions before the camera is ready to take the next picture, and that the probability of lightning at a particular direction does not change over time.

In this case, $\{N, E, S, W\}$ can be viewed both as the partition of the environment and as the set of possible sensor ranges at each time. It is clear that the optimal strategy is to direct the camera towards the area where the probability of lightning is maximal, and to keep taking pictures in that particular direction, without making use of the possibility of changing the direction of the camera.

An essential feature of this example is that the probability of an event e at time t always equals the probability of the event starting: $P_t(\mathbf{e}) = P(e \to 1)$. Moreover, it is clear that, in this case, the decision problem at time t is independent of the actions that are chosen at other times. Therefore, the global decision strategy and the 1-step strategy are equivalent. From the description of the goal 'to take as many pictures of lightning as possible' it can be inferred that the cost of not detecting a particular lightning flash does not depend on the area where this flash occurs. In that case, minimising expected cost at time t reduces to minimising the probability of not detecting a particular flash at t, which is equivalent to the strategy mentioned in the example.

3.3 (PO)MDP settings

In the previous section the surveillance problem was set and analysed in a decision-theoretic way. Here it will be set as a (PO)MDP. In fact, due to the generality of the (PO)MDP representation, it is possible to set it in several ways. To make clear that we are dealing with the same problem, we will show that the different

(PO)MDP settings are equivalent to each other and to the original decision-theoretic setting. The (PO)MDP representation will not be used in the later chapters. However it is still used in the next section where we derive the state-space size.

3.3.1 Introduction to POMDPs

A Partially Observable Markov Decision Process (POMDP) is a tuple $\langle S$, A, T, R, Ω, $O\rangle$. We describe in turn each element of the tuple.

States S. S is the finite set of states the world can be in. An element from this set is a possible way the world could exist.

Actions A. A is the set of possible actions. The set of possible actions can be the same for all $s \in S$ and in this case we write A. However, in some situations different actions are available for each state $s \in S$ and then we write A_s.

Transitions $T : S \times A \rightarrow \Pi(S)$. T is the state transition function. Given a state $s \in S$ and an action $a \in A_s$, it gives a probability distribution over resulting states. We often write $T(s, a, s')$ to represent the probability that, given action a was taken at state s, s' is the resulting state. Obviously, although the effects of actions can be probabilistic, they do not necessarily have to. In that case only one resulting state is possible.

Immediate Rewards $R : S \times A \rightarrow \mathbb{R}$. R is the immediate reward function. It gives the expected immediate reward obtained by the agent for taking each action in each state. We write $R(s, a)$ to represent the expected immediate reward for taking action a in state s.

Observations Ω. Ω is the finite set of observations the agent can experience in the world. An agent, instead of directly observing after each action the current environment state, receives an observation. This observation provides a hint about the current state of the environment.

Observation function $O : S \times A \rightarrow \Pi(\Omega)$. O is the observation function[1]. It gives for each action and resulting state the probability distribution over the possible observations. We write $O(s'a, o)$ to represent the probability that observation o was obtained when action a resulted in state s'.

[1]Note that in Puterman's book [Put94] the transition and observation functions are referred to as the transition and observation models.

Further assumptions

The POMDP model makes the assumption that the next state for a given action only depends on the current state. Looking back at the state history should not change our probabilities. So we are making the Markov assumption.

An MDP is a POMDP without imperfect observations and is sometimes called FOMDP (Fully Observable MDP). That is, the observations Ω and the observation function O are not present in MDPs. Instead, the agent always correctly determines the state S' it ends up in, after taking action A at state S. Note that the outcome of the actions is still probabilistic, the difference being that the agent can just determine directly the state it is in.

Optimality criteria

The goal of an agent using a POMDP is to maximise the future reward for an infinite time horizon. However, the methods of optimising over an infinite time horizon for general problems require an infinite amount of computational time. Since this is not possible, two alternative optimality criteria are proposed in the POMDP community. The first one is the *finite horizon* model. The other is the *infinite-horizon discounted* model.

In the finite horizon model we are trying to maximise the reward EU_T over the next T steps:

$$EU_T = \sum_{t=0}^{T} R_t$$

where R_t is the reward obtained at time-step t. It can be claimed that this criterion is inconvenient since the value of T has to be known in advance and it is hard to know T exactly. For our problem it is not too hard to pick a value of T based on the duration in real time units (e.g. seconds) of a time-step and the length of a shift of the robot (e.g. a night). This optimality criterion is almost identical to our expected cost criterion.

In the infinite-horizon discounted model, the discounted sum of the rewards over an infinite horizon EDU is maximised. The discounting is done using a discount factor $0 < \gamma < 1$ in order to guarantee that the sum is finite.

$$EDU = \sum_{t=0}^{\infty} \gamma^t R_t$$

where R_t is as before.

Policies and belief states

A policy describes how the agent should behave in some specific environment. For the MDP case (POMDP without the partial observability) a policy $\pi : S \rightarrow A$

specifies what action should be taken in every environment state. Solving an MDP can be seen as equivalent to finding a policy for the given MDP.

In the case of POMDPs the situation gets slightly more complicated. This is due to the fact that the agent does not know exactly what the current state of the environment is. For POMDPs the policy is specified on belief states rather than on environment states.

A belief state is probability distribution b over the possible actual world states S. This distribution encodes the agent's subjective probabilities about the state of the environment and provides the basis for decision making. We let $b(s)$ denote the belief (subjective probability) of the agent that the current state is s.

A *state estimator* has to compute the new belief state b' given the old belief state b, action a, and observation o.

3.3.1. PROPOSITION. *The new belief in some state s' is $b'(s')$ and can be written as:*

$$b'(s') = P(s'|o,a,b) = \frac{O(s',a,o)\Sigma_{s\in S}T(s,a,s')b(s)}{P(o|a,b)}$$

The following assumptions are made:

Assumption 1: The probability of an observation o depends only on the state s' and action a: $P(o|s',a,b) = P(o|s',a) = O(s',a,o)$.

Assumption 2: The probability of moving into a new state s', given a previous state s and an action a, does not depend on the old belief state b: $P(s'|a,b,s) = P(s'|a,s) = T(s,a,s')$.

Assumption 3: The probability of being at state s, given a belief state b and an action a, is independent of the a and is just the belief b of being at state s: $P(s|a,b) = b(s)$

Proof. The proof is by substitution.

$$\begin{aligned}
b'(s') &= P(s'|o,a,b) && \text{definition of } b \\
&= \frac{P(o|s',a,b)P(s'|a,b)}{P(o|a,b)} && \text{Bayes' rule} \\
&= \frac{O(s',a,o)P(s'|a,b)}{P(o|a,b)} && \text{assumption 1} \\
&= \frac{O(s',a,o)\Sigma_{s\in S}P(s'|a,b,s)P(s|a,b)}{P(o|a,b)} && \text{total probability rule} \\
&= \frac{O(s',a,o)\Sigma_{s\in S}T(s,a,s')b(s)}{P(o|a,b)} && \text{assumptions 2,3}
\end{aligned}$$

∎

The denominator in the equation of proposition 3.3.1 can be seen as a renormalisation factor that causes b' to sum up to 1. It can be computed as:

$$P(o|a,b) = \Sigma_{s'\in S}O(s',a,o)\Sigma_{s\in S}T(s,a,s')b(s)$$

The state estimator $SE(b,a,o)$ gives as an output the new belief state b' by repeating the calculation of $b'(s')$ for all $s' \in S$.

Now that the concept of a belief state has been introduced, we can talk about belief MDPs. A belief MDP is defined as follows:

- B is the set of belief states.
- A is the set of actions, defined as for the POMDP.
- $\tau(b,a,b')$ is the new state transition function and can be computed as follows:
  $$\begin{aligned}\tau(b,a,b') &= P(b'|b,a) = \Sigma_{o\in\Omega}\tfrac{P(b',b,a,o)}{P(b,a)}\\ &= \Sigma_{o\in\Omega}P(b'|b,o,a)P(o|b,a)\end{aligned}$$
  where:
  $$P(b'|b,o,a) = \begin{cases} 1 & \text{if } SE(b,a,o) = b' \\ 0 & \text{otherwise}\end{cases}$$
- $\rho(b,a)$ is the reward function on the belief states and is defined as:
  $$\rho(b,a) = \Sigma_{s\in S} b(s)R(s,a)$$

3.3.2 POMDP setting

In this section the problem of finding fires described in section 3.2.1 will be set as a POMDP. We describe here all the parts of the resulting POMDP plus the initial belief state.

States S**.** $\langle f_1,\ldots,f_n,X_l\rangle, f_i \in \{0,1\}, X_l \in X$, where f_i is a Boolean variable being 1 if a fire is present in room X_i, and X_l is the current robot location.

Actions A_s**.** The actions are state dependent. For a state $S = \langle f_1,\ldots,f_n,X_l\rangle$ the available actions are of the form $GO(X_g)$ with an action for each X_g such that $(X_l,X_g) \in A$. So the actions available at each state are dependent on the robot's immediate accessibility relation ($A \subseteq X \times X$).

Transitions $T : S \times A \to \Pi(S)$ The transitions are specified as follows:

For $s = \langle f_1,\ldots,f_k,\ldots,f_n,X_l\rangle$ and $a = GO(X_g)$ the probabilities are:

$P(s' = \langle f'_1,\ldots,f'_n,X_j\rangle) = 0$ if $j \neq g$

$P(s' = \langle f'_1,\ldots,f'_k,\ldots,f'_n,X_g\rangle) = 0$ if $f'_k < f_k \& k \neq l$

$P(s' = \langle f'_1,\ldots,f'_n,X_g\rangle) = \prod_{f'_a > f_a} P(f_a \to 1) \prod_{f'_b = 0}(1 - P(f_b \to 1))$

The first if-statement states that the robot cannot end up in the wrong location. This is a consequence of the perfect navigation-localisation assumption. The second if-statement denotes that fires do not stop on their own. This is a consequence of the assumption that only the robot can put out fires. The third if-statement calculates the probability of all remaining states (those not covered by the first two if-statements) as the product of the probability of fires starting and the probability of fires not starting.

Immediate Rewards $R : S \times A \rightarrow \mathbb{R}$. If in state s action a is taken, then a lot of resulting states s' are possible with different probabilities (specified in the transitions section). The utility of each possible resulting state $s' = \langle f_1, \ldots, f_n, X_l \rangle$ is $U(s') = C(\mathbf{f_l})$ if $f_l = 1$ and $U(s') = 0$ otherwise. This makes the expected reward for the state/action combination:

$$R(s,a) = \sum_{s'} T(s,a,s') * U(s')$$

Observations Ω. Observations Ω have the form: $\langle f_l \rangle$ where f_l is a Boolean variable signifying whether a fire is present at the current robot location l.

Observation function $O : S \times A \rightarrow \Pi(\Omega)$. The observation function for state $s' = \langle f_1, ..., f_n, X_l \rangle$ and action $a = GO(X_l)$ is:

$$\begin{array}{ll} \text{if } f_k \neq f_l & P(o = f_k) = 0 \\ \text{else} & P(o = f_l) = 1 \end{array}$$

You can notice that although the POMDP model requires us to specify the action on top of the resulting state, this action is not necessary in our case for specifying the observation function.

Initial State Initially, only state A_0 is considered possible. This is represented by having a belief state b such that $b(A_0) = 1$ and $b(s) = 0$ for all other $s \in S$. Then as time passes by, the belief is distributed over more states.

Belief MDP resulting from our POMDP

Since now we know what the POMDP for our problem is, the question of what the resulting belief MDP looks like can be answered. To do that we examine the part of section 3.3.1 on belief states for POMDPs and specify the corresponding belief MDP components step-by-step:

States B. A single belief state is a distribution over all possible states S of the POMDP. For example, $b(\langle 1, 1, \ldots, 1, X_l = 2 \rangle) = 0$ can be our belief that everything is on fire and that we are currently in location $X_l = 2$. This belief is part of one belief state, and B is the set of all possible belief states. One important characteristic of the belief states in our problem is that only the world states with the correct robot location are possible. This is an artifact of the perfect robot localisation assumption. So $b(\langle \ldots, X_k \rangle) = 0$ if $X_k \neq X_l$ where X_l is the robot's location.

Actions A_b. The actions are as before. They are belief state dependent. For a belief state b the available actions are of the form $GO(X_g)$ with an action for each X_g such that $(X_l, X_g) \in A$. Here note that X_l can be extracted from the belief state b by looking at just one state s such that $b(s) > 0$

Transitions $\tau : B \times A \rightarrow \Pi(B)$. These transitions have to be computed as previously described.

Immediate Rewards $\rho : B \times A \rightarrow \mathbb{R}$. The reward function of the belief states B is defined as:

$$\rho(b,a) = \Sigma_{s \in S} b(s) R(s,a) = \Sigma_{s \in S} b(s) \Sigma_{s' in S} T(s,a,s') * U(s').$$

So $R(s,a)$ is substituted with the value it has for the POMDP.

3.3.3 Direct MDP settings

Although it is possible to obtain a belief MDP from the POMDP setting of the problem, another alternative is to define an MDP directly based on the model of the environment introduced at the beginning of this document. The resulting MDPs are "belief MDPs" in the sense that they specify how our belief should be updated. A difference from the belief MDP resulting from the POMDP is that here our state is related to how much we believe a fire to be present in each environment location. In the POMDP case our belief state is how much we believe an entire state of the environment is possible.

First setting as MDP

States S. $\langle t_1, \ldots, t_n, X_l \rangle, t_i \in [0, \infty), X_l \in X$, where t_1 to t_n are the times since a room was last visited and X_l is the current robot location.

Actions A_s. The actions are state dependent. For a state $s = \langle t_1, \ldots, t_n, X_l \rangle$ the available actions are of the form $GO(X_g)$ with an action for each X_g such that $(X_l, X_g) \in A$.

Transitions $T : S \times A \rightarrow \Pi(S)$. The transitions are completely deterministic and so each (s,a) pair corresponds to a single state s'. So $T : S \times A \rightarrow \Pi(S)$ has now the form $T : S \times A \rightarrow S$

$$\begin{aligned} T(s,a) = \quad & T(\langle t_1, \ldots, t_l, \ldots, t_n, X_l \rangle, GO(X_g)) \rightarrow \\ & \langle t_1 + 1, \ldots, t'_l = 1, \ldots, t_n + 1, X_g \rangle) \end{aligned}$$

In the setting of section 3.2.1, once a room is visited, the fire is immediately extinguished. Here we effectively do the same but the effects of the extinguishing action are delayed until the next time-step. This is done so that the reward of visiting a state where the room is on fire can be specified. If the fire was extinguished immediately, then reaching a state where the room is on fire would be no different than reaching a state where the room is not on fire. This would make the reward specification very hard.

Immediate Rewards $R : S \times A \to \mathbb{R}$. Since the actions are deterministic, the state action pair (s, a) always results in the same state s'. Then the reward is: $R(s,a) = C(\mathbf{f_l}) * P(\mathbf{f_l})$ where $P(\mathbf{f_l}) = 1 - (1 - P(f_l \to 1))^{t_l}$ and X_l is the robot location in s'. So given the state s', the *a priori* probabilities of a fire starting, and the cost for each room, the reward for each (s, a) can be computed.

Second setting as MDP

States S. $\langle p_1, \dots, p_n, X_l \rangle, p_i \in [0,1]$[2], $X_l \in X$, where p_i is the probability of a fire being present in a room X_i, and X_l is the current robot location.

Actions A_s. The actions are state dependent. For a state $s = \langle p_1, \dots, p_n, X_l \rangle$ the available actions are of the form $GO(X_g)$ with an action for each X_g such that $(X_l, X_g) \in A$.

Transitions $T : S \times A \to \Pi(S)$. The transitions are completely deterministic and so each (s, a) pair corresponds to a single state s'. So $T : S \times A \to \Pi(S)$ has now the form $T : S \times A \to S$

$$\begin{aligned} T(s,a) = \quad & T(\langle p_1, \dots, p_l, \dots, p_n, X_l \rangle, GO(X_g)) \to \\ & \langle p'_1 = 1 - (1 - p_1)(1 - P(f_1 \to 1)), \\ & \dots, p'_l = P(f_l \to 1), \dots \\ & p'_n = 1 - (1 - p_n)(1 - P(f_n \to 1)), X_g \rangle \end{aligned}$$

So at every time-step the fire starting probabilities are used to update the current state of the environment. The effect of the fire extinguishing action is delayed until the next time-step for the same reason as in the transition model of the first MDP setting.

Immediate Rewards $R : S \times A \to \mathbb{R}$. Since the transitions are completely deterministic each (s, a) pair corresponds to a single state s'. So, given the state s' and the cost for each room, the reward for each (s, a) can be computed as $R(s, a) = C(\mathbf{f_l}) * p_l$, where X_l is the robot location in s'.

[2]note p_i is an alternative notation for $P(\mathbf{f_l})$

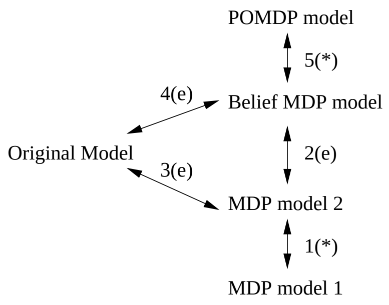


Figure 3.5: Equivalence demonstration schema. Equivalence proof is supplied for (*) and equivalence claim is made plausible by an example for (e).

3.3.4 Equivalence of different settings

In this section we show that the various surveillance models presented so far are representationally equivalent. That is, we show the underlying problem to be the same, no matter which representation is used. The model equivalence is demonstrated by means of a collection of pairwise equivalences. There are some proofs involved and the equivalence demonstration schema can be seen in figure 3.5.

Some of the equivalences are made plausible by means of working out by hand example 3.2.4 for both the belief POMDP and the second MDP settings. The choice to use this example is slightly arbitrary. The main reason for choosing it is that it is simple and this makes it possible to work it out by hand. In fact, for the case of belief MDPs this simulation on example 3.2.4 is still too long and it is exhibited in appendix A to save space in the text of this chapter. Although, this example is simple, it has been used in the section on strategies to demonstrate some differences between them. We believe that it is hard to find an example where these equivalences would not hold but, unfortunately, we have not been able to find proofs.

Two MDP models (equivalence 1)

We begin our equivalence proofs by demonstrating that the following proposition holds (label 1 of fig. 3.5).

3.3.2. PROPOSITION. *The two direct MDP settings are equivalent.*

Proof. To prove this we show that there is a mapping between the states of the two MDPs and that it does not matter which path of figure 3.6 we follow during state transitions.

We begin by showing that there is mapping between states. Based on proposition 3.2.2 and the definition of states in the second MDP model, we can say that the probability p_i of a room being on fire in a state s_2 of the second MDP model is $p_i = 1 - (1 - P(f_i \to 1))^{t_i}$, where t_i is the time since the last visit in the corresponding state s_1 of the first MDP model. This formula gives us an easy way of converting states of the first MDP model into states of the second. This formula can be inverted. Beginning from:

$$p_i = 1 - (1 - P(f_i \to 1))^{t_i}$$

we can write:

$$(1 - P(f_i \to 1))^{t_i} = 1 - p_i \Leftrightarrow t_i \log(1 - P(f_i \to 1)) = \log(1 - p_i)$$

giving finally:

$$t_i = \frac{\log(1 - p_i)}{\log(1 - P(f_i \to 1))}$$

This last formula gives us a way of converting states of the second MDP model into states of the first. Actually, the above equation is undefined if $P(f_i \to 1) = 0$ or $P(f_i \to 1) = 1$. This means that in those two cases we cannot really tell how long ago a room was visited from its probability of being on fire.

Now that the mapping between states has been demonstrated, we continue by showing the equivalence of state transitions. Suppose that the state in the first MDP is $s_1 = \langle t_1, \ldots, t_l, \ldots, t_n, X_l \rangle$. Using the information in s_1 the corresponding state s_2 of the second MDP can be found:

$$\begin{aligned} s_2 = \langle p_1 = &\ 1 - (1 - P(f_1 \to 1))^{t_1}, \ldots, \\ p_l = &\ 1 - (1 - P(f_l \to 1))^{t_l}, \ldots, \\ p_n = &\ 1 - (1 - P(f_n \to 1))^{t_n}, X_l \rangle \end{aligned}$$

Then suppose that action $GO(X_g)$ is taken while at state s_2, the resulting state s_2' is:

$$\begin{aligned} s_2' = \langle p_1' = &\ 1 - (1 - p_1)(1 - P(f_1 \to 1)), \ldots, \\ p_l' = &\ P(f_l \to 1), \ldots, \\ p_n' = &\ 1 - (1 - p_n)(1 - P(f_n \to 1)), X_g \rangle \end{aligned}$$

State s_2' can be rewritten after substitution of values from s_2:

$$\begin{aligned} s_2' = \langle p_1' = &\ 1 - (1 - P(f_1 \to 1))^{t_1+1}, \ldots, \\ p_l' = &\ P(f_l \to 1), \ldots \\ p_n' = &\ 1 - (1 - P(f_n \to 1))^{t_n+1}, X_g \rangle \end{aligned}$$

Note now that s_2' is equivalent (using the inverse mapping mentioned above) to:

$$s_1'' = \langle t_1 + 1, \ldots, t_l' = 1, \ldots, t_n + 1, X_g \rangle$$

But s_1'' is the same as s_1' that can be directly computed from state s_1 when transition $GO(X_g)$ is followed. We have thus shown that the transitions in the two MDPs produce equivalent results.

∎

$$\begin{tikzcd}
\text{MDP } 1\langle t_1, \ldots, t_l, \ldots, t_n, X_l\rangle \arrow[r, "GO(X_g)"] \arrow[d, leftrightarrow] & \langle t_1+1, \ldots, t'_l = 1, \ldots, t_n+1, X_g\rangle \arrow[d, leftrightarrow] \\
\text{MDP } 2\langle p_1, \ldots, p_l, \ldots, p_n, X_l\rangle \arrow[r, "GO(X_g)"] & \begin{array}{l}\langle p'_1 = 1-(1-p_1)(1-P(f_1 \to 1)), \ldots, \\ p'_l = P(f_l \to 1), \ldots, \\ p'_n = 1-(1-p_n)(1-P(f_n \to 1)), X_g\rangle\end{array}
\end{tikzcd}$$

Figure 3.6: Equivalence sketch for the two MDPs.

Original and Belief MDP model (equivalence 4)

We now make plausible that the following claim holds (label 4 of fig. 3.5).

3.3.1. Claim. *The belief MDP and the original models are equivalent.*

The example given in appendix A can be seen as an informal demonstration that the original setting and the belief MDP are equivalent. The justification of this statement is that although the settings of the given problem are different, the results computed by iterating the two models are the same, both in terms of robot actions taken and in terms of expected action benefit.

Belief MDP and direct MDP model (equivalence 2)

We now make plausible that the following claim holds (label 2 of fig. 3.5).

3.3.2. Claim. *The belief MDP and the second direct MDP models are equivalent.*

To make this equivalence plausible, we will give a setting of example 3.2.4 within the second MDP model, then iterate the resulting MDP, and show that the same results are computed.

States S**.** $\langle p_1, p_2, X_l\rangle, p_i \in [0,1], X_l \in \{X_1, X_2\}$.

Actions A_s**.** Here the actions are not state dependent because the state is too small. Two actions are available: $GO(X_1)$ and $GO(X_2)$.

Transitions $T : S \times A \to \Pi(S)$**.** The transitions are:

$$\begin{aligned}
T(s,a) = \quad & T(\langle p_1, p_2, X_1\rangle, GO(X_g)) \to \\
& \langle p'_1 = 0.5, p'_2 = 1-(1-p_2)(1-0.8), X_g\rangle \\
T(s,a) = \quad & T(\langle p_1, p_2, X_2\rangle, GO(X_g)) \to \\
& \langle p'_1 = 1-(1-p_1)(1-0.5), p'_2 = 0.8, X_g\rangle
\end{aligned}$$

Immediate Rewards $R : S \times A \to \mathbb{R}$**.** The rewards are given as previously described.

Now given this concrete setting, the MDP can be iterated with a one-step look-ahead. The complete iteration can be found in figure 3.7. The starting state is the state where the probability of both rooms being on fire is zero. Then both actions are checked to determine their rewards $R(s, a)$ and the action with the largest reward is taken. This procedure is iterated a few times. It can be seen that the actions taken during this iteration are the same as those in appendix A. Furthermore, the rewards on which the decisions are based are also the same as those found in appendix A.

Original and direct MDP model (equivalence 3)

This example can be seen to make the following claim (label 3 of fig. 3.5) plausible.

3.3.3. Claim. *The original and the second direct MDP models are equivalent.*

POMDP and Belief MDP model (equivalence 5)

3.3.3. Proposition. *The POMDP and belief MDP models are equivalent.*

This proposition corresponds to label 5 of figure 3.5. This is true according to the definition of belief MDPs. A proof that the belief state is a sufficient statistic for past history of observations of the POMDP can be found in [SS73]. Their proof shows that it is not necessary to remember the entire history of the POMDP to determine its next state. That is, knowing the old belief state, the action taken and the observation received are sufficient to determine the new belief state. Furthermore, the state estimator $SE(b, a, o)$ is derived as part of their proof.

3.4 Surveillance state space size

Having described several problem settings and having shown that these are equivalent, we proceed to derive the state space size for the surveillance problem. We begin by giving a simple argument about why it is exponential and give a possible objection to that argument. We then go on by proving that the state space size is indeed exponential.

In the case of surveillance the state space is described as a tuple of times since last visit containing one such time per room. A single state can be described as:

$$\langle t_1, \ldots, t_n, X_l \rangle, t_i \in [0, \infty), X_l \in X$$

The times since last visit are in theory allowed to go to infinity, which makes the state space infinite in size. However, in normal situations not all times are used because the behaviour of the robot exhibits some periodicity. Consequently, a limit T is imposed on how long a time since last visit can be. The state space

$$\begin{tikzcd}
 & s_1 = \langle 0, 0, X_1 \rangle \arrow[dl, "a_1"'] \arrow[dr, "a_2"] & \\
\begin{array}{l} s_2 = \langle 0.5, 0.8, X_1 \rangle \\ R(s_1, a_1) = 0.5 \end{array} & & \begin{array}{l} s_3 = \langle 0.5, 0.8, X_2 \rangle \\ R(s_1, a_2) = \mathbf{0.8} \end{array} \arrow[dll, "a_1"'] \arrow[d, "a_2"] \\
\begin{array}{l} s_4 = \langle 0.75, 0.8, X_1 \rangle \\ R(s_3, a_1) = 0.75 \end{array} & & \begin{array}{l} s_5 = \langle 0.75, 0.8, X_2 \rangle \\ R(s_3, a_2) = \mathbf{0.8} \end{array} \arrow[dll, "a_1"'] \arrow[d, "a_2"] \\
\begin{array}{l} s_6 = \langle 0.875, 0.8, X_1 \rangle \\ R(s_5, a_1) = \mathbf{0.875} \end{array} \arrow[d, "a_1"'] \arrow[drr, "a_2"] & & \begin{array}{l} s_7 = \langle 0.875, 0.8, X_2 \rangle \\ R(s_5, a_2) = 0.8 \end{array} \\
\begin{array}{l} s_8 = \langle 0.5, 0.96, X_1 \rangle \\ R(s_6, a_1) = 0.5 \end{array} & & \begin{array}{l} s_9 = \langle 0.5, 0.96, X_2 \rangle \\ R(s_6, a_2) = \mathbf{0.96} \end{array} \arrow[dll, "a_1"'] \arrow[d, "a_2"] \\
\begin{array}{l} s_4 = \langle 0.75, 0.8, X_1 \rangle \\ R(s_9, a_1) = 0.75 \end{array} & & \begin{array}{l} s_5 = \langle 0.75, 0.8, X_2 \rangle \\ R(s_9, a_2) = \mathbf{0.8} \end{array}
\end{tikzcd}$$

Figure 3.7: Iterating the second MDP setting.

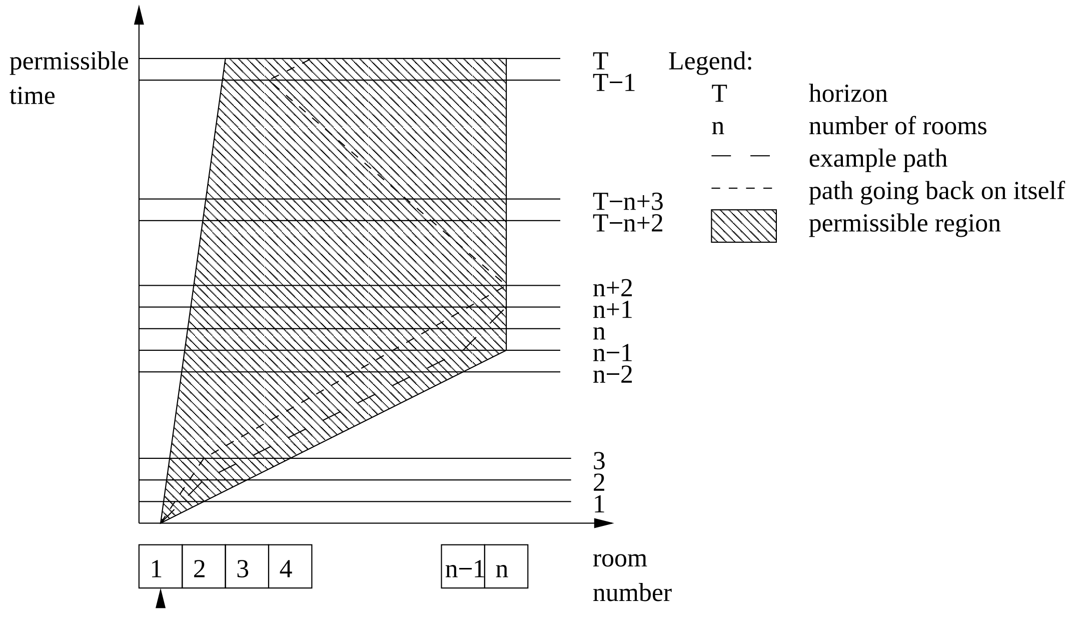


Figure 3.8: Permissible times vs room number.

size is exponential in the number of rooms included in a state (it is T^n, where n is the number of rooms).

One can object to the previous argument and say that the state space size is not exponential in the number of rooms. A specific time since last visit for a single room sets constraints on the times since last visit for the other rooms. Those constraints are imposed by (a) the fact that the robot can be in only one room at a given time and (b) by the connectivity between the rooms. The constraints seem to imply that the state space size does not increase exponentially in the number of rooms.

3.4.1 State space size derivation

Here it is argued that although the argument about the presence of constraints and about the state space size not being T^n is correct, the state space size is, in fact, still exponential.

To show this we first limit the types of states we are examining by: (a) concentrating on a corridor environment, and (b) concentrating on the type of states where the robot is on the leftmost room of the corridor. A diagram representing the admissible states of this type can be generated (fig. 3.8). You can see in the boxes at the bottom of the diagram that the robot is in the leftmost room and this room can only have a time since last visit of 0. The rest of the rooms can have times at some interval between 0 and T (shaded region), but not all values are allowed because of the constraints mentioned.

Assuming that every room was visited at some point, the lowest time value allowed for any room is its distance from the leftmost room. This is 1 for room 2, 2 for room 3 and so on. An interesting observation is that there are $n-1$ places where our constraints apply and that the width of the possible options at each of those places is at least $T-(n-1)$.

A possible MDP state corresponds to a path in this diagram. Note that although the times are drawn as straight lines, they are discrete, and so should be jagged. An example of a state represented as a path is the state $\langle 0, 2, 3, \ldots, n-1, n+1\rangle$ which is depicted as a dashed line in fig. 3.8. These paths have the further constraint of having to be monotonically increasing with a slope of at least 1. Paths that partially go back on themselves are possible but still the time since last visit would be strictly decreasing with a slope of -1. An example of such a path is given in figure 3.8 and the times since last visits generated by this path are represented by its lower line segment as: $\langle 0, 3, 4, \ldots, n+1, n+2\rangle$. Perhaps the important observation is that it is the most recent visit that matters in determining the time since last visit and hence the state of the MDP.

The number of such paths is, at least, like putting T distinguishable balls into $n-1$ indistinguishable boxes and then sorting those indistinguishable boxes based on the size of the ball in ascending order. The number of such possibilities is:

$$\begin{pmatrix} T \\ n-1 \end{pmatrix} = \frac{T!}{(n-1)!(T-(n-1))!} \tag{3.1}$$

This formula gives us for a specific T and n the number paths satisfying the constraints in our diagram of fig. 3.8. So we have a formula to compute how many states of this specific type (corridor, leftmost room) exist. In reality, there are a few more possibilities because the figure is wider on one end, and because paths that go back on themselves are allowed.

3.4.2 The value of T and the exponential expression

The size of T is related to the number of states that can be represented. If $T = n-1$, we get only one state $\left(\frac{(n-1)!}{(n-1)!0!} = 1\right)$, so the rest of the states encountered by the robot in a circular path cannot be represented.

T should be set to a value that is large enough to accommodate all possible interesting paths and all the states that the robot would encounter along those paths. In the case of the corridor the least possible T to pick is $2(n-1)$ because we need so many time-steps to walk along the corridor and return to the original position. So a path just doing a round trip in the corridor needs at least $T = 2(n-1)$ to be representable in our MDP. The example of the path going back on itself should be seen as an indication of why that is so.

We can convert equation 3.1 into one not involving factorials by using their

Stirling approximation. The Stirling approximation of factorial is:

$$n! \approx \sqrt{2n\pi}\left(\frac{n}{e}\right)^n$$

Applying this to the formula for combinations we get:

$$\begin{aligned}\begin{pmatrix} T \\ m \end{pmatrix} &\approx \frac{\sqrt{2T\pi}\left(\frac{T}{e}\right)^T}{\sqrt{2m\pi}\left(\frac{m}{e}\right)^m\sqrt{2(T-m)\pi}\left(\frac{T-m}{e}\right)^{T-m}} \\ &= \frac{\sqrt{2T\pi}\,T^T}{2\pi\sqrt{m(T-m)}m^m(T-m)^{T-m}}\end{aligned}$$

The last equation is the general equation for combinations for any T and m. Now in our case $m = n-1$, and for a corridor case we can set $T = am = a(n-1)$. Now we get:

$$\begin{aligned}\begin{pmatrix} am \\ m \end{pmatrix} &\approx \frac{\sqrt{2am\pi}(am)^{am}}{2\pi\sqrt{m(am-m)}m^m(am-m)^{am-m}} \\ &= \frac{\sqrt{a}\,a^{am}m^{am}}{\sqrt{2\pi m(a-1)}m^m(m(a-1))^{m(a-1)}}\end{aligned}$$

giving:

$$\begin{pmatrix} am \\ m \end{pmatrix} \approx \frac{\sqrt{a}}{\sqrt{2\pi m(a-1)}}\left(\frac{a^a}{(a-1)^{(a-1)}}\right)^m \tag{3.2}$$

Equation 3.2 is the general exponential expression for any a. Taking $a = 2$, the number of states can be approximated as:

$$\begin{pmatrix} 2(n-1) \\ n-1 \end{pmatrix} \approx \frac{1}{\sqrt{\pi(n-1)}}4^{n-1}$$

So for $T = 2(n-1)$ we get an exponential growth. Therefore, even for the simple path in a corridor, the number of states available grows exponentially.

Given that the probabilities and the costs can also vary, we are likely to have more complicated paths with the robot staying in some rooms. Those paths require $T > 2(n-1)$ even for the corridor case, not to mention that there are other types of environment like stars (chapter 5) that might need an even larger T.

3.5 Standard (PO)MDP solving approaches

In this section we discuss several standard (PO)MDP solving approaches. The focus on (PO)MDP based methods is due to the fact that they are what is most commonly used to solve other decision-theoretic problems in robotics. We hope it will become clear that none of these methods is really suited to solve interesting instances of the robot surveillance problem.

3.5.1 Value functions; value and policy iteration

In section 3.3.1 we have already defined a policy as a mapping $\pi : S \rightarrow A$ from states S to actions A that specifies what action should be taken in every environment state.

Given the policy of an MDP, and assuming that we are using the discounted infinite horizon optimality model, we can define for each state its infinite horizon value function as:

$$V_\pi(s) = R(s, \pi(s)) + \gamma \Sigma_{s' \in S} T(s, \pi(s), s') V_\pi(s')$$

The value function is the expected cost resulting from starting in state s and the following policy π. Here the discounted optimality criterion of section 3.3.1 is used in which the value of states away from state s are discounted by γ.

The discounted infinite horizon solution to the MDP is a policy that maximises its expected value V_π. Howard [How60] showed that there is an optimal policy π^* that is guaranteed to maximise V_π, no matter what the starting state of the MDP is. The value function for this policy V_{π^*}, also written V^*, can be defined as:

$$V^*(s) = \max_a [R(s, a) + \gamma \Sigma_{s' \in S} T(s, a, s') V^*(s')]$$

This has a unique solution and if the optimal value function V^* is known, then the optimal policy π^* is defined as:

$$\pi^*(s) = \text{argmax}_a [R(s, a) + \gamma \Sigma_{s' \in S} T(s, a, s') V^*(s')]$$

There are two very standard ways of finding the optimal policy, (a) policy-iteration and (b) value iteration. The policy iteration method (see algorithm 3.9) starts from a randomly chosen policy. The algorithm proceeds by repeatedly trying to find alternative actions for each state that improve the current state value function. The new actions found replace the old policy actions. The iterative improvement of policies stops when no policy-improving actions can be found.

In value iteration (see algorithm 3.10) optimal policies are produced for successively longer finite horizons until they converge with some error less than ϵ. Assuming that for a look-ahead of 0, $V_0(s) = 0$, the algorithm computes value functions V_t based on the policy found using the value function V_{t-1}. The algorithm terminates when the maximum change in the value function is below a threshold.

Policy iteration and value iteration can find optimal policies in time polynomial in the number of states in the MDP. However, as we have shown, the number of MDP states is exponential in the number of locations used to describe the state space. This state space is defined as the cross product of those locations. So these methods cannot be used to solve the Direct MDP formalisations of section 3.3.3.

Furthermore, in the case of POMDPs, the belief MDP corresponding to the POMDP has a continuous state space and this complicates matters further. In fact, in [PT87] it is shown that finding policies for POMDPs is a PSPACE-complete problem and this makes exact solutions in polynomial time less likely than for NP-complete problems. Policy iteration and value iteration can, therefore, find optimal policies but for the smallest POMDPs. In the rest of this section we are going to describe techniques for finding POMDP policies for larger problems.

```
for each $s \in S$ do $\pi(s) \leftarrow RandomElement(A)$ end;
repeat
    Compute $V_\pi(s)$ for all $s \in S$;
    for each $s \in S$ do
        Find some $a$ such that $[R(s,a) + \gamma\Sigma_{s' \in S}T(s,a,s')V_\pi(s')] > V_\pi(s)$;
        if such an $a$ exists then
            $\pi'(s) \leftarrow a$;
        else
            $\pi'(s) \leftarrow \pi(s)$;
        end
    end
until $\pi'(s) = \pi(s)$ for all $s \in S$;
return $\pi$;
```

Figure 3.9: The policy iteration algorithm

```
for each $s \in S$ do $V_0(s) \leftarrow 0$ end;
$t \leftarrow 0$;
repeat
    $t \leftarrow t + 1$;
    for each $s \in S$ do
        for each $a \in A$ do
            $Q_t(s,a) \leftarrow R(s,a) + \gamma\Sigma_{s' \in S}T(s,a,s')V_{t-1}(s')$
        end
        $\pi_t(s) \leftarrow \text{argmax}_a Q_t(s,a)$;
        $V_t(s) \leftarrow Q_t(s, \pi_t(s))$
    end
until $\max_s |V_t(s) - V_{t-1}(s)| < \epsilon$;
return $\pi_t$;
```

Figure 3.10: The value iteration algorithm

3.5.2 Piecewise linear discretisations

As previously mentioned, one of the problems in finding policies for POMDPs is that the state space of the belief MDP is continuous. Several methods have been proposed for solving POMDPs [Lov91]. All of them have to deal with the problem of continuous state spaces. One naive suggestion is that of discretising the continuous state space using a fixed discretisation. However, this is not an efficient way of dealing with the discretisation problem.

In [SS73], it was suggested that for the finite horizon optimality model the optimal POMDP value function is piecewise linear and convex. For a given horizon, the continuous belief space can be split into regions (pieces) and within those regions the optimal value function can be described using a linear function of the belief space. The region boundaries arise naturally as a side effect of the fixed finite horizon. The piecewise value function is convex in that the surface defined by the union of the hyper surfaces within each region is convex. For a proof of those two statements look at [SS73]. These two properties can be used to derive a POMDP version of value iteration that discretises the environment accurately at the borders of these regions in each iteration.

In [Son78] this method has been extended for the infinite discounted horizon POMDP optimality model. The function remains piecewise linear because the value iteration in the infinite horizon case stops iterating when the difference in the value functions between iterations is under a limit. Therefore, the value function computed in the infinite-horizon version of value iteration is still computed in a finite number of iterations. Hence, the piecewise and convex properties are still present. If the value iteration was to continue for an infinite number of iterations, the resulting function would still be piecewise linear, but would possibly have infinite piecewise segments. The discounting is necessary to guarantee that the optimal value function will be finite and this, in turn, is necessary to guarantee that value iteration will eventually converge with this optimal value function. The value iteration methods used for POMDPs using the piecewise linearity property can solve problems where the underlying MDP has around 2-5 states [LCK95b] So these methods are rather restrictive.

In [KLC96] an improved version of value iteration for POMDPs called the "witness algorithm" is proposed. In [LCK95a] it is mentioned that the witness algorithm can deal with up to around 16 states in the underlying MDP, but this is still a rather restrictive result. In [LCK95b] an attempt is presented to find *approximate* solutions that can provide satisfactory solutions for problems with around 100 states. An even newer *approximation* method [MKKC99] has been used to solve problems with up to 1000 states. However, in our problem the underlying MDP has at least $2^{|X|}$ states where X is the set of all possible environment locations (as described in section 3.2.1). For $|X| = 32$ the number of states is close to 4 billion. The conclusion is that the methods based on the piecewise linearity of the value function performing either exact or approximate solution computation cannot be used for our problem.

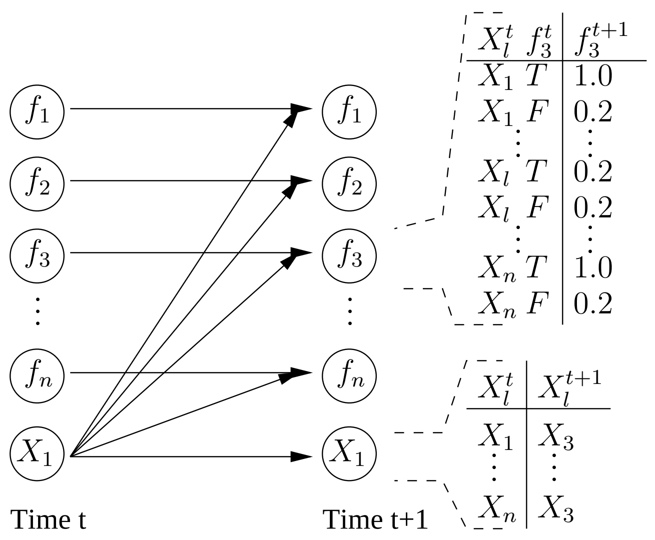
3.5.3 Planning with factored representations

As previously mentioned, the state space of an MDP is exponential in size in the number of literals used to describe it. Consequently, the MDP state transition and state reward tables are also exponential in size in the number of literals used to describe the state space. In [BDH96] the distinction is made between flat and factored state spaces. A flat state space is a state space where a single literal is used to describe each state - the literal taking on as many values as the number of states in the state space. A factored state space is one where each state is composed of many literals (factors / fluents) - each literal taking on fewer values than the size of the state space. Note that we have already been implicitly considering factored state spaces in our discussion of the surveillance problem.

The main observation in [BDH96] and in later articles [BDH99, HSAHB99] is that in some problems independences can be exploited to reduce the size of the problem description. At the action level it may happen that the post action value of a literal depends on the pre-action values of only a few literals. Similarly, the reward of an action and the value of a state might be structurable on the value of state space literals. It can possibly be so that some literals will not influence at all the value of the state.

The suggestion in [BDH96, BDH99] is to use temporal Bayesian networks to represent the state transition probabilities. The claim is that Bayesian networks can reduce the space needed for transitions, from exponential in the number of literals (using transition tables) to polynomial (using Bayesian networks). In figure 3.11 such a Bayesian network is used to represent the action $GO(X_3)$ of our surveillance application. From the network and the conditional probability tables (CPTs) associated with it, one can see that the robot location X_l^{t+1}, after action $GO(X_3)$ is taken, does not depend on what the location X_l^t was in the pre-action state. Similarly, the presence of a room fire at $t+1$ only depends on whether a room fire already existed at t and on what the location X_l^t was. One such Bayesian network has to be introduced for each action in our environment. Actually, the Bayesian network only requires less space than the state transition table when independence structure is present, that is, when each time $t+1$ literal does not have all time t literals as parents. If no independence structure is present, it does not make sense using Bayesian networks in the representation. However, typically such structure exists.

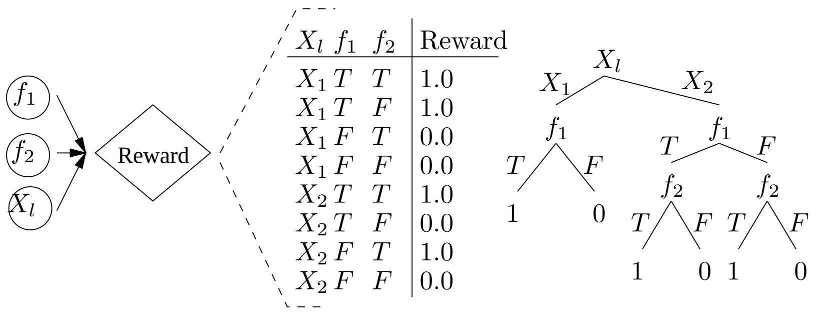
The other suggestion in [BDH96, BDH99] is using influence diagrams to compactly represent the value/reward of a state based on the value of the literals. This makes sense if the reward is only dependent on a few literals. In our case the value depends on the current location X_l and the presence of a fire f_l at the current location X_l. This means that all literals $f_1 \dots f_n$ have to be used in the influence diagram and so the influence diagram is not significantly smaller than a



X_l^t	f_3^t	f_3^{t+1}
X_1	T	1.0
X_1	F	0.2
⋮		⋮
X_l	T	0.2
X_l	F	0.2
⋮		⋮
X_n	T	1.0
X_n	F	0.2

X_l^t	X_l^{t+1}
X_1	X_3
⋮	⋮
X_n	X_3

Figure 3.11: Action network for the $GO(X_3)$ action with two CPTs shown.



X_l	f_1	f_2	Reward
X_1	T	T	1.0
X_1	T	F	1.0
X_1	F	T	0.0
X_1	F	F	0.0
X_2	T	T	1.0
X_2	T	F	0.0
X_2	F	T	1.0
X_2	F	F	0.0

Figure 3.12: The reward influence diagram and the decision tree for the two-room example.

reward table. The decision tree corresponding to the influence diagram, however, is smaller than the reward table.

Both of these suggestions are used in [BDH96, BDH99] to produce an algorithm called SPI. This algorithm is based on Sondik's value iteration, but instead of computing piecewise linear value functions, it is computing new influence diagrams to represent the state value function. The algorithm performs better than classical piecewise linear value function algorithms because only literals that are relevant to the outcome under a particular action are considered. Knowing the state value function trivially gives us the policy by picking the action that takes us to the best state.

In [HSAHB99] algebraic decision diagrams are used for representing rewards and value functions instead of influence diagrams. Algebraic decision diagrams are generalisations of the binary decision diagrams often used in computer circuit design. Algebraic decision diagrams allow assigning non-Boolean values to leaves and so can be used to represent rewards (which can be real valued). Various operations are defined upon algebraic decision diagrams, such as adding two diagrams or multiplying them. The claim in [HSAHB99] is that the algorithms based on algebraic diagrams can benefit from the fast tree merging implementations already existing for Boolean decision diagrams. Furthermore, an approximate method is suggested whereby parts of the algebraic decision diagrams that have little effect on the final value of a state are pruned to reduce the computation.

3.5.4 Reinforcement learning using neural networks

In this section we discuss reinforcement learning using neural networks as a method for approximately computing the value function. We begin by discussing the problem described in [CB96] of optimal elevator dispatching policies for a building with 4 elevators and 10 floors. This problem has several characteristics similar to those of our problem. The elevators are responding to call buttons being pressed. In our problem, if uniform room costs are assumed, the surveillance robot will visit rooms likely to be on fire. In a sense pressed buttons can be thought to be corresponding to fires. Furthermore, a lift takes into account, while parking, where it is more likely to be called up next (in our problem, the robot looks at the probability of a fire starting).

However, there are also some differences. Firstly, the call buttons can be thought of as perfect sensors distributed over all floors (in our case, each room would have a perfectly reliable fire sensor). Secondly, queues of people waiting for a lift can be formed (in our case, multiple fires would be present in a room). Thirdly, once the person is in the lift, the lift still has to transport the person to the right floor so the task is not equivalent to just picking up the person (as it is in our case). Finally, in the elevator problem we are dealing with a one-dimensional motion problem, while in the surveillance problem, connections between rooms play an important role. So the two problems are not immediately equivalent. However, here too, the state space is huge since all possible combinations of call and elevator buttons plus all the possible lift positions and directions have to be represented.

The solution proposed in [CB96] combines the concept of reinforcement learning with a neural network based approach to it. The neural network and reinforcement learning combination performs better in this problem than the standard known solutions to the elevator dispatching problem. However, we have several objections to this solution. Our first objection concerns the number of inputs and the input significance, as part of the state representation is ignored in the inputs, and some inputs correspond to heuristically determined properties. Our second objection has to do with the number of hidden units and various learning parameters, such as the discount factor and learning rate, which were experimentally set . Finally, no attempt of justification for those choices was made. The authors themselves admit that a significant amount of experimentation was necessary for determining the appropriate network shape parameters.

Another frequently cited example of good performance exhibited by reinforcement learning methods is that of TD-Gammon [Tes95, Tes94, Tes92]. TD-Gammon is a backgammon-playing program that uses a version of reinforcement learning called temporal difference learning (TD-learning) [Sut88] to learn the playing policy. The problem in TD-Gammon is very different from ours since a policy that can be followed against intelligent opponents has to be found. This is the domain of game theory instead of decision theory.

In TD-Gammon the agent learns the best playing policy by starting without any *a priori* knowledge about the game of backgammon, apart from the rules. That is, there is no knowledge derived from experts about how a player should play backgammon in order to win. TD-Gammon plays games of backgammon against a copy of itself and learns from the mistakes it makes. At the end of each game a reinforcement (reward) for the entire game is computed (using the backgammon scoring scheme). The problem then becomes one of assigning some of this game reinforcement to each of the actions performed by the program during the game. A similarity between the solution in [CB96] and the elevator problem is that in both cases the value function for each state is represented in a multi-layer perceptron and that the correct value function is learned using back propagation.

The results obtained with TD-Gammon are impressive. A policy, good enough to follow against master level humans, is found. If some *a priori* knowledge is incorporated in the input to the multi-layer perceptron in the form of interesting board configuration features, then TD-Gammon can reach grand master level play. However, as in the elevator scheduling problem, some parameters have to be configured experimentally. Furthermore, and perhaps more importantly, no definite explanation of its good performance can be given. In fact, it is always possible that TD-Gammon will find a locally optimal solution although, according to the authors, that seldom happens.

From our analysis of TD-Gammon and of the elevator scheduling problem solution we conclude that although after some experimentation good reinforcement learning solutions using neural networks can be found for those problems, they are not necessarily open to introspection and cannot be guaranteed to be optimal or near optimal.

3.5.5 State-action pair sampling

As we have previously mentioned, standard exact methods, like value and policy iteration in the case of MDPs, have runtimes polynomial in the number of states in the underlying MDP and exponential in the number of problem variables. In [KMN99] a method is proposed whose runtime is not dependent on the number of states in the MDP. This method relies on sampling the look-ahead tree and producing a sparse look-ahead tree that is dense enough to guarantee some optimality conditions.

The algorithm uses a generative model G that, given an action a and a state s, randomly outputs a new state s' according to the transition probabilities $T(s, a, s')$. The algorithm takes an on-line view; given a state, a decision about the best action has to be taken. No fixed policy is computed as in value or policy iteration. Going back to the optimality conditions the algorithm guarantees that, given the generative model G, an input state $s \in S$ and a tolerance value $\epsilon > 0$, the action output satisfies the following conditions:

- The runtime is $O(kC)^H$, where k is the number of actions available at each state, C is the number of samples taken for each state-action pair, and H is the look-ahead depth of the tree. C and H are determined by the algorithm but are independent of the state space size (see fig. 3.14).

- The value function of the approximate policy $V^A(s)$ is such that its difference from the optimal policy V^* is below ϵ:

$$|V^A(s) - V^*(s)| \leq \epsilon$$

The complete algorithm A can be found in figure 3.14. The top level function calculates the right values for C and H given the required tolerance ϵ and discount factor λ. Subsequently, it proceeds to compute an estimate $\hat{Q}^*_H$ of the optimal state-action value function Q^*_H and select the action with the best $\hat{Q}^*_H$ value. The Q functions should be seen to be a special kind of value functions V with two arguments, one for the states s we begin from and another for the action a taken at that state. The computational advantage of the algorithm derives from the approximate computation of the state-action value function $\hat{Q}^*_H$. In the calculation of $\hat{Q}^*_h(s, a)$ (line 1 in the algorithm), only C sample resulting states are considered instead of the full state space S. If the size of the state space $|S| = N \geq C$, then a computational advantage can be gained because the tree searched is smaller than the full search tree. A proof that the algorithm really satisfies the above-mentioned optimality criteria can be found in [KMN99].

The state-action sampling approach is well suited for situations where an action can take the system to every state $s \in S$. However, as was seen in the example of appendix A, our belief MDPs have rather structured transition probability tables and this structure can probably be better exploited. Furthermore, in [MV98], we have shown that our search for a decision-theoretic strategy using the original problem setting could provide us with a solution of $O(k^h)$ time complexity, where k is the number of actions available at each location and h is the number of steps to look ahead. There, as well as in the MDPs proposed in section 3.3.3, the actions are completely deterministic and only a single resulting state is possible for each state-action pair. So, in fact, the decision-theoretic search can do better than the state-action sampling approach for our specific problem.

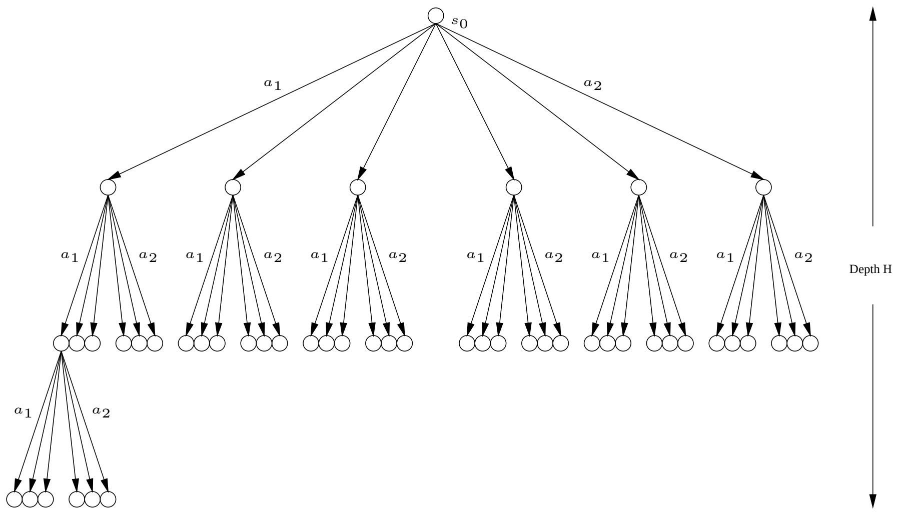


Figure 3.13: The look-ahead tree for the state-action sampling algorithm for $C = 3$ and $k = 2$.

Furthermore, the interpretation of tolerance ϵ here is not immediately intuitive. Tolerance is not defined as a proportional difference from the optimal value function. One needs to have an idea of what the value of the optimal value function is before a reasonable tolerance can be specified.

One seeming advantage of the discussion in [KMN99] over our discussion in [MV98] is that given the tolerance and the discount factor, the depth H that has to be used in the search in order to achieve the desired tolerance ϵ can be computed. In [MV98], no optimality guarantees are given and the maximal depth used is only determined based on computational constraints (the robot has to act in real time). Although the approach of [KMN99] is more formal, if some reasonable values ($\epsilon = 0.2, \gamma = 0.9, R_{max} = 1$) are substituted in the equations of algorithm 3.14, large values are computed for C and H ($C = 3003, H = 39$). For such large values for C and H the runtime will most probably be very large, making the application of this algorithm unrealistic for the desired C and H.

In any case, we hope that from this section it is clear that our problem has transition tables with a rather specific structure and that the state-action pair sampling algorithm is designed for problems with a different type of transition table structure.

3.6 Summary

There are several equivalent ways in which the surveillance problem can be set. The state space size is exponential in the number of locations used to describe the environment in all these settings. The exponential state space size in conjunction with the results in [PT87] implies that standard exact MDP solving methods

```
**Function: Estimate Q**$(h, C, \gamma, G, s)$
**if** $h = 0$ **then return** $(0, \ldots, 0)$;
**for** *each* $a \in A$ **do**
    generate $C$ samples using $G$;
    let $S_a$ be the set containing these $C$ samples;
**end**
**for** *each* $a \in A$ **do**
    let our estimate of $Q^*(s, a)$ be:

        $\hat{Q}^*_h(s, a) \leftarrow R(s, a) + \gamma \frac{1}{C} \Sigma_{s' \in S_a} \text{Estimate V}(h - 1, C, \gamma, G, s')$

**end**
**return** $(\hat{Q}^*_h(s, a_1), \hat{Q}^*_h(s, a_2), \ldots, \hat{Q}^*_h(s, a_k))$ ;

**Function: Estimate V**$(h, C, \gamma, G, s)$
$(\hat{Q}^*_h(s, a_1), \hat{Q}^*_h(s, a_2), \ldots, \hat{Q}^*_h(s, a_k)) \leftarrow \text{Estimate Q}(h, C, \gamma, G, s)$;
**return** $\max_{a \in \{a_1, \ldots, a_l\}} \{\hat{Q}^*_h(s, a)\}$ ;

**Function: Algorithm A**$(\epsilon, \gamma, R_{max}, G, s_0)$
$\lambda \leftarrow \frac{\epsilon(1-\gamma^2)}{4}$;
$V_{max} \leftarrow \frac{R_{max}}{1-\gamma}$;
$H \leftarrow \log_\gamma \frac{\lambda}{V_{max}}$;
$\delta \leftarrow \frac{\lambda}{R_{max}}$;
$C \leftarrow \frac{V^2_{max}}{\lambda^2}(2H \log \frac{kHV^2_{max} \log \frac{1}{\delta}}{\lambda^2} + \log \frac{1}{\delta})$ ;
$(\hat{Q}^*_H(s, a_1), \hat{Q}^*_H(s, a_2), \ldots, \hat{Q}^*_H(s, a_k)) \leftarrow \text{Estimate Q}(H, C, \gamma, G, s_0)$;
**return** $\text{argmax}_{a \in \{a_1, \ldots, a_l\}} \{\hat{Q}^*_H(s, a)\}$;
```

Figure 3.14: Sparse MDP sampling

cannot solve our problem, and that approximate methods need to be tried.

We have demonstrated that all the different settings are equivalent, and so our choice of representation should not affect results. We felt that it would be clearer and more convenient if just one of these settings was used in the rest of this thesis. We are probably right in saying that POMDPs are harder to compute using paper and pencil (e.g. in appendix A). So we decided to use the decision-theoretic setting of section 3.2.1 in the rest of this thesis.

A further observation is that at least some of the POMDP solving methods discussed attempt to use the structure of the specific problem attacked. State-action pair sampling computes solutions faster, due to the fact that in their problem setting some resulting states are more likely than others for a given state-action pair. Factored representations make use of the fact that in some problems the sets of possible actions and resulting states depend only on some of the parameters of the current state. This seems to indicate that the structure of the problem is important in efficiently computing the solution. In fact, in [WM95], it is shown that the best way to solve a specific search problem is to make a customised search method that looks carefully into what the "salient features" of the problem are. However, none of the standard approximate methods proposed can be applied to our problem because they take advantage of different types of structure from that present in our problem.

We decided to concentrate on approximate methods built specifically for our problem and for the scenarios that a robot is likely to encounter in an office-like environment. Our problem has a lot of geometric structure in it. For example, the possible state transitions are greatly constrained by the structure of the physical environment. None of the methods we have seen so far explicitly tries to take into account the geometrical structure present in the motion actions. Our claim is that using it to guide our approximations is the best way to produce fast and accurate solution algorithms for the surveillance problem.

Chapter 4

Hierarchy and Strategies

In this chapter, we show that a hierarchical approach to surveillance can improve the simple minimum expected cost (MEC) strategy proposed in section 3.2.2. The minimum expected cost strategy can only be evaluated with a limited look-ahead and this causes problems related to local minima. One such local minima problem is that of "expected cost obstacles" in example 3.2.10.

The hierarchical decision theoretic strategy proposed here attempts to overcome these problems by simultaneously using descriptions of the environment at different levels of abstraction, where decisions made at more abstract levels guide the decision making at less abstract levels. Although it also uses a limited look-ahead, it deals with much larger time horizons than the original minimum expected cost strategy, because the higher levels of the hierarchy can shift the focus of attention to areas of the environment far away from the current robot position.

While the hierarchical strategies of this chapter are heuristic and the approximation of the expected cost is rather *ad hoc*, they improve performance in situations with local minima. They are presented to justify the use of a hierarchical approach. In later chapters, we will devise an improved hierarchical strategy which better exploits the geometrical structure in our environment.

4.1 An office-like environment

The strategies of this chapter are compared in a realistic office-like environment that is based on an existing office building (fig. 4.1:a). This environment is used to guide our thought on what kind of methods are applicable in a real life scenario. Describing our method in terms of a real environment serves to make our ideas more concrete. This environment will also be used in later chapters to compare surveillance strategies. Although it is just an instance of an office-like building it has features that are common in many such buildings.

It is assumed that an *a priori* map of the building is available. This map is an accurately labeled occupancy grid that is composed of walls, doors, and free space. Locations X_i are rooms and segments of the corridor (fig. 4.1:b) and fires have given starting probabilities $P(f_i \to 1)$ and known costs $C(\mathbf{f_i})$.

4.1.1 Environment graph construction

Before applying the decision strategies to the office environment, each room is identified and marked. The corridor (room 0) is segmented into many smaller rooms. The places where to segment the corridor are determined by the position of the doors connecting other rooms to the corridor. By segmenting the corridor, a mapping of all potential decision points of the robot to individual locations is achieved (fig. 4.1:b). Such a segmentation is also considered natural, for an office environment in [TRS01]. Finally, the marked room environment representation is converted into a graph where rooms correspond to graph nodes and doors to graph links (fig. 4.1:c). This representation is called the *environment graph*.

The n-step minimum expected cost strategy can be applied to the environment graph. An n-step minimum expected cost strategy is essentially doing a depth-first search of a tree of possible decisions. The time needed to take a single decision using such a strategy is exponential with an $O(b^n)$ time complexity (see e.g. [Lug02] page 106), where b is the branch factor of the decision tree being explored and n is the number of steps to look ahead. In our environment, b is the average connectivity of the environment graph being explored and has a value of about 3. In connected environments, where there is a link to a neighbouring room and one to the room itself, the lowest value b can take is 2. This sets a lower bound of 2^n on the time complexity.

4.1.2 The cost barrier problem

It is obvious that under certain circumstances, n-step strategies can face a difficult situation in our environment.

4.1.1. Example. Consider the case in fig.4.2 where $C(\mathbf{f_i}) = 0$ for $i \in \{5, 6, 7, 8, 9, 10, 32, 33, 34, 35, 36\}$, $C(\mathbf{f_j}) = \hat{c}$ for $j \in \{1, 2, 3, 4, 28, 29, 30, 31\}$, $C(\mathbf{f_k}) = 1$ for all $k \neq i \wedge k \neq j$, $P(f_l \to 1) = 0.001$ for all l and $A_0 = 50$. The variable $\hat{c}$ can take any positive value and values from 1 to 100 are used in our simulations.

In this setting an "*expected cost barrier*" is formed by the X_is that an n-step minimum expected cost strategy (section 3.2.2) with $n \leq 5$ cannot cross. All the barrier locations X_i have an expected cost of 0. A robot starting at room 50 can get up to room 37 but then its immediate neighbours will all have an expected cost of 0 and a visit there will not be justified. The rooms X_j will not be visited either because they can only be considered once the robot is in one of the X_is. Therefore, the rooms X_j will never be visited although, after a sufficiently large

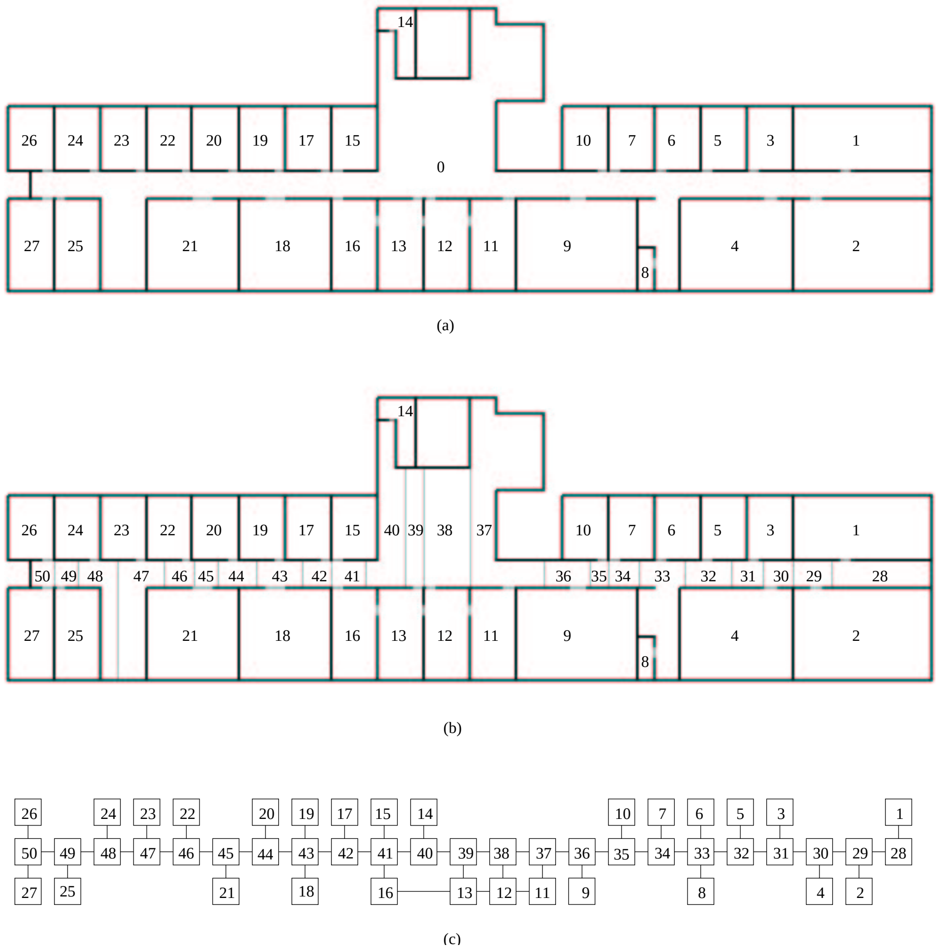


Figure 4.1: (a) Initial map (b) Segmented map (c) Environment graph.

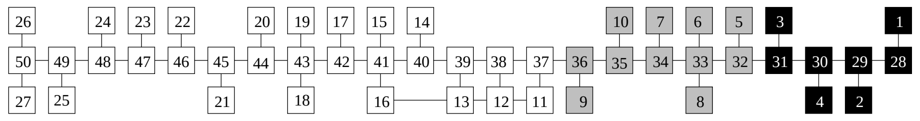


Figure 4.2: Fire costs in example 4.1.1. Rooms for which $C(\mathbf{f}) = 0$ (barrier rooms) are grey, rooms for which $C(\mathbf{f}) = \hat{c}$ are black and rooms for which $C(\mathbf{f}) = 1$ are white.

period, they will almost certainly be on fire and the cost of fire there is relatively high (if $\hat{c} > 1$). This is an example of an expected cost barrier.

Of course, a 6-step strategy can overcome this barrier, but then a larger one can always be constructed. Furthermore, the decision to use 6 rather than 5 steps has to be based on knowledge about the presence of a barrier. Such knowledge must either be given in advance or has to be derived based on an analysis of the environment. It is not obvious how such an analysis might be performed since effective cost barriers can exist in more subtle situations. Finally, as we mentioned in the previous section, there is a real-time, performance-imposed upper limit on the value of n that can be used since the complexity of the search is exponential.

4.1.3 Simulation experiments

The environment graph of this example was used in the simulations. Furthermore, virtual sensors that can detect if a fire is present within a certain graph node were implemented. Initial fire locations, probabilities and costs are specified at the level of environment graph nodes.

The hierarchical as well as the minimum expected cost and the minimax interval strategies were simulated in the environment of example 4.1.1. Several instances of that environment with different $\hat{c}$ values for the barrier rooms were used in the simulations. Further, a "uniform" environment was used where all rooms had costs of 1 and so no barrier was present.

Both the evolution of fires in the environment and the decisions taken by the robot were simulated. Fires could start at all locations and their costs were accumulated over many simulated time-steps. The accumulative costs represent the ground truth on how well the strategy did in that particular run. The smaller the resulting cost, the better the strategy minimised the cost.

The robot had knowledge about the time since last visit for all locations as well as for the $P(f_i \rightarrow 1)$ and $C(\mathbf{f_i})$ for all $X_i \in X$. These values were used to compute the estimate of the expected cost for each location and guide the strategies accordingly. The simulated robot attempted to take the actions that minimised the expected cost.

In the simulations, 10000 iterated time-steps were used for each run of the simulator. Taken together with a probability of fire starting of $P(f_i \rightarrow 1) = 0.001$ means that a fire almost definitely starts in each of the environment's 50 locations during the simulation run. Twenty runs of 10000 were used for each strategy to give some estimate of the effects of the randomness and to compute the standard deviation of our measurements. Since the size of our time-steps in real time units would be determined by the time a real robot would take to move to an adjacent room, one can claim that a $P(f_i \rightarrow 1) = 0.001$ is too high. Although this is true in most cases, a lower fire starting probability would imply that more iterations would be necessary in our simulations to ensure that statistically correct results are gathered. Although, $P(f_i \to 1) = 0.001$ is quite high, we believe that, if this probability is lowered, but at the same time the environment becomes larger (incorporating many floors or buildings), the same qualitative results will be obtained.

We have plotted the number of times each room is visited by the 5-step minimum expected cost strategy (fig. 4.6:i). Since this strategy cannot overcome the expected cost barrier, the groups of nodes a, b, c and d in the plot are not visited. So neither the nodes beyond the barrier nor the nodes in the barrier are visited.

4.2 The hierarchical minimum expected cost strategy

The proposal is to overcome the barrier problems by means of a hierarchical minimum expected cost strategy. In order to implement a hierarchical minimum expected cost strategy two subtasks are performed.

- The environment graph is abstracted. The aim is to generate a hierarchical model of the environment.
- A reasonable strategy is found. This strategy has to apply minimum expected cost to the hierarchical model of the environment, in a way that corresponds in a well-defined approximate manner to the actual MEC strategy at the lowest level.

The hierarchical strategy is proposed here to demonstrate the benefit of using a hierarchy in our approximations.

4.2.1 Abstraction tree construction

To construct the abstraction tree, the environment graph was repeatedly clustered and abstracted. Only example 4.1.1 was considered and so the clustering process was not automated as the effort was not justified. However, the following criteria leading to the possibility of its automation were taken into account.

1. A path within the cluster should exist between any two nodes in a cluster.
2. No path between any two nodes within a cluster should be greater than the look-ahead size n (we took $n = 5$).
3. The resulting tree should be balanced. This means that at each level all clusters should have approximately the same number of children.

The clustering was performed recursively until the environment graph could not be further abstracted. In the resulting abstraction tree, the abstract nodes

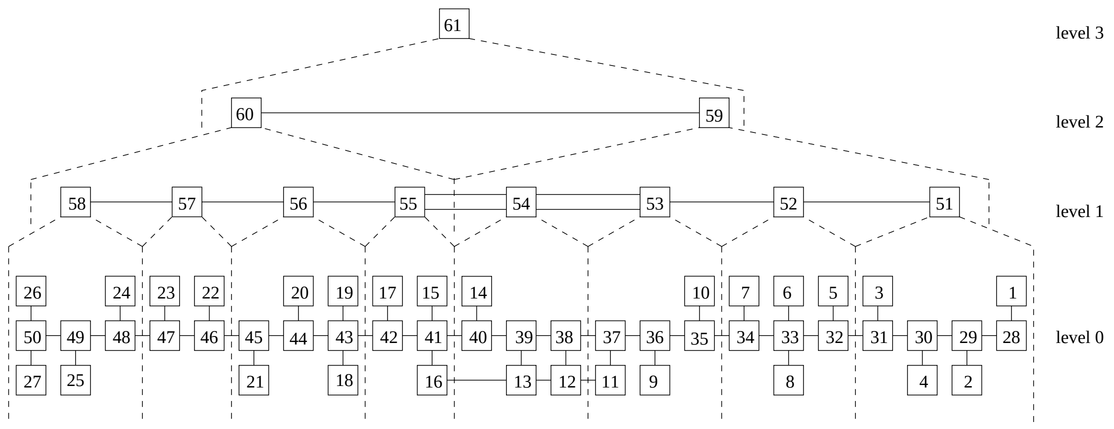


Figure 4.3: Abstraction tree.

at each level are interconnected with links that map to doors (fig. 4.3 (levels 1 and 2)).

4.2.1. Definition. The expected cost EC of an abstract node is defined recursively as:

$$\begin{array}{ll} EC(X_i) = P_t(\mathbf{f_i})C(\mathbf{f_i}) & \text{if } level(X_i) = 0. \\ EC(X_i) = \sum_{j \in ch(X_i)} EC(X_j) & \text{if } level(X_i) > 0. \end{array}$$

where $ch(X_i)$ is the set of children of node X_i, $P_t(\mathbf{f_i})$ is the probability of fire in room X_i at t time-steps since its last visit (definition 3.2.2), and $C(\mathbf{f_i})$ is the cost of fire per time-step at room X_i. So the expected cost of an abstract node is the sum of the expected costs of all its level 0 descendants.

4.2.2. Definition. The expected cost EC_p of a path $p = [X_f|r]$ is defined recursively as:

$$EC_p([X_f|r]) = EC(X_f) + EC_p([r])$$

where X_f is the first node in the path, r is the rest of the path, and EC is the expected cost of an abstract node as computed by definition 4.2.1.

4.2.2 Decision procedure

The hierarchical minimum expected cost strategy (fig. 4.4) begins from the top of the abstraction tree and works its way down through its different levels. The original minimum expected cost strategy is used to select the best node to visit at each level. If at the current abstraction level the best node does not correspond to the robot location at that level, a path is planned to a child of the selected best node, and so the abstract cluster is changed. If the best node corresponds

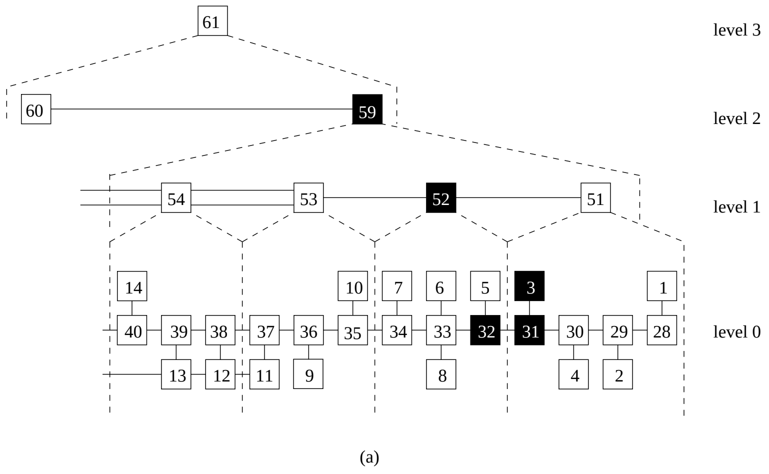
```
present_room ← ROOM;
% where ROOM is the room the robot is currently in
abstraction_level ← MAX ;
% where MAX is the most abstract level with more than 1 node
change_abstract_node ← false;
%if change_abstract_node is set to true then the robot directly plans a
%path to another abstract node

while abstraction_level > 0 do
    present_node ← map present_room to abstraction_level node;
    select best_next_node using 5-step MEC at current abstraction_level;
    if best_next_node = present_node then
        abstraction_level ← abstraction_level − 1;
    else
        change_abstract_node ← true;
        target ← map best_next_node to closest 0 level node to present_room;
        best_path ← plan shortest level 0 path from present_room to target;
        abstraction_level ← 0;
    end
end
if change_abstract_node=false then
    best_path ← plan using 5-step committed MEC at level 0;
end
follow best_path until its end is reached;
repeat the decision procedure to select new best_path;
```

Figure 4.4: Algorithm for the hierarchical minimum expected cost strategy.

to the robot location, the same decision procedure is repeated for the next lower abstraction level.

If no decision is made to change abstract nodes and level 0 is reached, the robot has to plan a 5-step "committed minimum expected path" within its present level 1 node. "Committed minimum expected cost" means that once a path is selected by the strategy, the robot sticks to it until its end. Ordinary 5-step strategies are reevaluated after each step is taken. So once a path is selected by an ordinary strategy, the first location in the path is visited and then a new 5-step reevaluation takes place. By using "committed minimum expected cost" a path staying in the cluster visits five rooms and hence its length is closer to that of a path moving to a different part of the environment. Further, using commitment solves the problem with the n-step strategies described in example 3.2.12.

To clarify the hierarchical strategy we give a couple of concrete examples. In the first example the abstract node is changed (fig. 4.5:a).



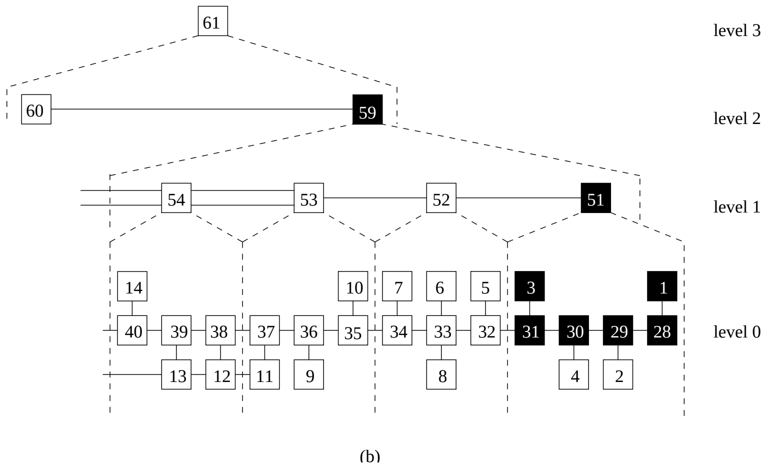


Figure 4.5: Example 4.2.3 (a) and 4.2.4 (b).

4.2.3. Example. The robot starts in room 3 at level 0 of the abstraction tree. The strategy has first to decide which node at level 2 should be visited. Suppose its outcome is that the robot should stay within the abstract node 59. Then another decision has to be made at level 1. If the decision is that it should visit node 52, then the robot has to move to another node. Consequently, a direct path is planned and the robot moves from 3 to 31 and then 32 because this is the shortest path from 2 to some descendant of node 52.

In the second example the robot never changes abstract node (fig. 4.5:b).

4.2.4. Example. The robot again begins at room 3 and the same line of reasoning is followed until abstraction level 1. Then suppose a decision is taken at level 1 to visit node 51. The main loop should be exited and the robot has to use a committed minimum expected cost strategy starting from node 3. Consequently, a longer visit of node 51 is planned rather than a move to a different abstract node.

The proposed hierarchical strategy was applied in the environment of example 4.1.1 for $\hat{c} = 50$. The number of visits per location was again plotted (fig. 4.6:ii). The location groupings a and c are visited indicating that the robot went beyond the expected cost barrier. This is a positive result. Furthermore, the barrier locations in the corridor (group d) are visited when the robot crosses to the other side of the barrier. The barrier locations that are not in the corridor (group b) correctly are not visited since no benefit can be gained by doing so.

4.2.3 Revised decision procedure

One problem we encountered is that the robot frequently swaps between abstract nodes 59 and 60. This problem cannot be seen directly from fig. 4.6:ii, but its effect is that we have rather a lot of visits along the corridor of the barrier (around rooms 32-36) and along the rest of the corridor of node 59 (along rooms 36-40). It occurs because, although the rooms past the barrier can have a rather high $\hat{c}$, they are not sufficiently many to keep the robot busy for a long period. This means that at the top level of abstract nodes 59 and 60 not visiting node 60 for many steps is also comparatively expensive. However, a brief visit to one of these abstract nodes is sometimes enough to tip the scales in favour of the other and then a lot of direct cluster changing paths are planned as a net result. To make things worse, a visit to cluster 59 most of the time results in a visit to cluster 51, which is more expensive, and that is why so many visits along the corridor of cluster 59 take place.

To fix this problem we revised hierarchical MEC to prefer paths that start at the current node X_l. This might seem strange since we just visited those rooms, but at the abstract levels it makes sense since it tends to make the robot explore the local neighbourhood properly before moving elsewhere.

4.2.5. Definition. The revised expected cost EC_p^* of a path $p = [X_f|r]$ is defined as:

$$\begin{array}{ll} EC_p^*([X_f|r]) = \kappa EC_p([X_f|r]), \kappa > 1 & \text{if } X_l = X_f. \\ EC_p^*([X_f|r]) = EC_p([X_f|r]) & \text{if } X_l \neq X_f. \end{array}$$

where X_f is the first node in the path, r is the rest of the path, κ is a delay constant, X_f, X_l are nodes at the same level of the abstraction tree and EC_p is computed using definition 4.2.2.

The purpose of κ is to delay moving from the current node. It can perhaps be thought of as a "hysteresis" parameter. The bigger the value of κ the longer the delay in deciding when to move into a new cluster. It might appear counter-intuitive to want to stay longer at a node that was just examined. However, this

	$\hat{c} = 100$	$\hat{c} = 50$	$\hat{c} = 10$	$\hat{c} = 1$	uniform
$\kappa = 1.0$	407 ± 28	278 ± 17	130 ± 7.0	100 ± 5.5	321 ± 6.4
$\kappa = 1.1$	350 ± 32	230 ± 14	79.5 ± 5.5	27.6 ± 1.7	39.3 ± 2.6
$\kappa = 1.2$	**279±20**	162 ± 11	**61.9±4.1**	$20.4 \pm .93$	29.1 ± 2.2
$\kappa = 1.3$	329 ± 6.6	**145±11**	64.4 ± 3.6	18.9 ± 1.2	25.8 ± 1.1
$\kappa = 1.4$	325 ± 9.2	302 ± 6.0	92.0 ± 5.7	18.0 ± 1.3	26.5 ± 1.7
$\kappa = 1.5$	328 ± 8.3	301 ± 5.7	123 ± 6.9	**17.3±1.2**	**22.0±1.6**
$\kappa = 1.6$	327 ± 8.7	304 ± 7.0	143 ± 7.5	19.3 ± 1.0	28.7 ± 1.4
$\kappa = 1.7$	326 ± 11	304 ± 6.6	158 ± 7.4	22.0 ± 1.4	34.4 ± 1.5
$\kappa = 1.8$	331 ± 9.9	305 ± 7.4	183 ± 9.1	35.0 ± 2.3	47.7 ± 2.4
$\kappa = 1.9$	358 ± 28	316 ± 13	202 ± 9.0	690 ± 3.4	92.9 ± 4.9
$\kappa = 2.0$	$7,460 \pm 320$	$3,830 \pm 130$	975 ± 22	315 ± 7.7	415 ± 6.8

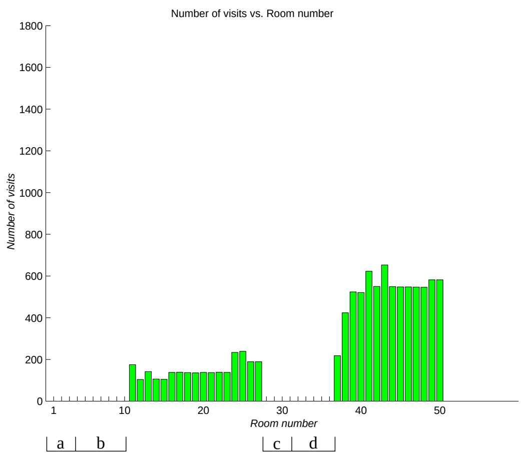
Table 4.1: The search for a κ value. Total costs and standard deviation after 10000 iterations in units of 10^3. Best costs for each environment in bold.

is reasonable if the current node is important and the expected cost computed for the other nodes is an overestimation of what will be seen once there. At the abstract levels of the hierarchy the expected cost of a node *is* an overestimation of what is seen once a robot visits that node. This is demonstrated in the following example.

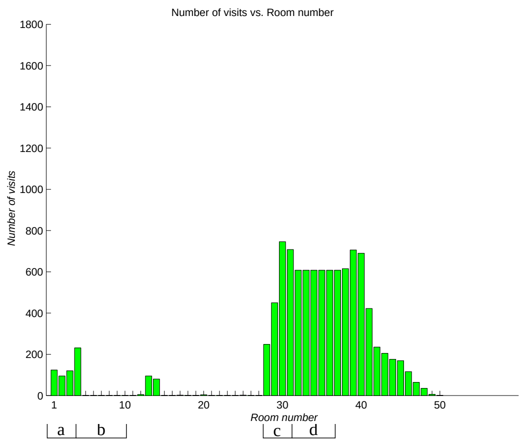
4.2.6. Example. The expected cost computed for node 60 is that of visiting all its subnodes. These subnodes correspond to almost half the environment of example 4.1.1. However, it is possible for the hierarchical expected cost to just spend a 5-step path within node 60 and then move back to node 59. If that is the case, the decision to move to node 60 would be based on the expected cost of half the environment, and consequently on an overestimation of what was seen by moving to node 60.

We have tried various values of κ for different $\hat{c}$ and the results can be summarised in table 4.1. We found that the values of κ that worked best were between 1.2 and 1.5. In the table the best costs for each environment are printed in bold. We decided to set κ to 1.3. This value is probably rather specific for the environment we consider. It is a rather arbitrary *ad hoc* choice and it is the weakest part of this approach.

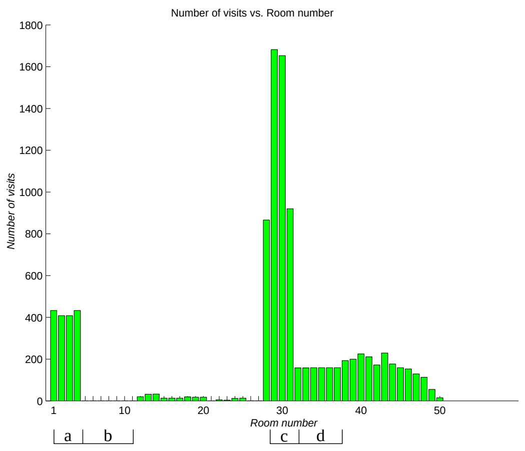
We then replotted the number of visits per room for the revised hierarchical strategy for $\kappa = 1.3$ (fig. 4.6:iii). Although now there are some irregular peaks in the plot, the previous problem of swapping between nodes 59 and 60 is solved. Furthermore, the nodes in groups a and c are visited more often and this also constitutes an improvement.



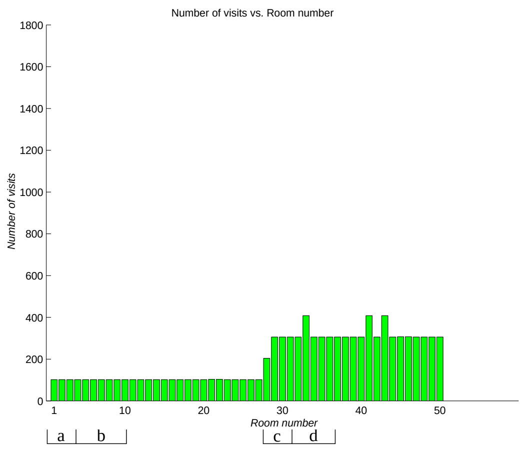
(i)



(ii)



(iii)



(iv)

Legend:

a = 1,2,3,4 = rooms that are beyond the barrier.

b = 5,6,7,8,9,10 = rooms that form part of the barrier.

c = 28,29,30,31 = corridor segments that are beyond the barrier.

d = 32,33,34,35,36 = corridor segments that form part of the barrier

Figure 4.6: Number of visits vs. room number for example 4.1.1 (i) 5-step MEC (ii) hierarchical MEC (iii) revised hierarchical MEC (iv) minimax interval.

	minimax interval	5-step MEC	hierarchical MEC	
			original	revised
uniform	**19.6±1.2**	22.6 ± 1.4	321 ± 6.1	25.8 ± 1.1
$\hat{c} = 1$	**15.3±1.1**	81.4 ± 2.8	102 ± 4.4	18.9 ± 1.2
$\hat{c} = 10$	**47.0±5.3**	734 ± 26	127 ± 9.7	64.4 ± 3.6
$\hat{c} = 50$	196 ± 24	$3,640 \pm 130$	281 ± 14	**145±11**
$\hat{c} = 100$	371 ± 53	$7,250 \pm 210$	410 ± 31	**329±6.6**

Table 4.2: Total costs after 10000 iterations in units of 10^3.

4.2.4 Discussion

The hierarchical strategy succeeds in crossing the barrier. More importantly, however, the hierarchical strategy also reduces the total cost in comparison to MEC. In table 4.2 the total costs (in units of 10^3) and their standard deviations are listed for minimax interval, 5-step MEC, the original hierarchical MEC and the revised hierarchical MEC. Minimax interval and 5-step minimum expected cost were first defined in section 3.2.2. The minimax interval strategy, given a room, selects its neighbour with the largest time since last visit. It completely ignores costs and probabilities and, in the long run, it explores all rooms in our environment. The 5-step MEC strategy minimises the expected cost over 5 time-steps.

Instances of our example with different $\hat{c}$ values were simulated for 10000 iterations to collect the data in the table. The original hierarchical MEC strategy does not always reduce the total cost in comparison with MEC. However, the revised MEC wins over MEC in all cases but the one of uniform costs. This is an indication that we have improved on the original MEC strategy in the cases where costs matter.

The improvement in total cost arising from making "not moving" at higher levels more attractive, is quite dramatic. The delay κ is necessary because the expected cost assigned to the abstract nodes is always an overestimation of the actual cost. The expected cost estimate assumes that once the strategy visits an abstract node, it stays there long enough for all sub-nodes to be visited. However, it is often the case that the strategy stays in a node only long enough for its expected cost to decrease and for the other nodes to attract the attention of the strategy. When that happens, the actual cost is only a portion of the estimated expected cost. We believe that by making "not moving" more attractive, the revised hierarchical strategy stays longer in a node and so our abstracted cost is a better estimate of the real cost (see example 4.2.6).

A possibly more interesting observation that can be made from table 4.2 is that minimax interval, which does not use the probabilities and costs, performs better than MEC and hierarchical MEC in the "uniform" cost case and in the situations where $\hat{c}$ is 1 or 10. These are situations where the cost differences are not very big. Minimax interval moves in the environment in a way that can almost be described as "methodical". Some environment locations are visited more frequently than others only due to the connectivity of the environment (fig. 4.6:iv). In the situations where $\hat{c} = 1$ and $\hat{c} = 10$ this "methodical exploration" of the environment produces better results than our *ad hoc* revised MEC. However, we believe that a good hierarchical MEC strategy should be able to perform better even in those cases and, therefore, we consider this an indication that the realisation in this chapter of this approach is not the best possible.

The fact that MEC performs so badly should not be surprising since MEC never passes to the other side of the barrier. The original hierarchical MEC also appears to be problematic despite perhaps being an improvement on MEC. It is clear that both our original and revised hierarchical MEC strategies are not optimal.

4.3 Summary

In chapter 3 we have shown that a minimum expected cost strategy is a promising method for planning in robotic surveillance. In practice, however, minimum expected cost typically cannot be evaluated globally. Only n-step minimum expected cost strategies are feasible. In order to overcome problems imposed by local minima and barriers, we propose using hierarchical versions of the minimum expected cost strategy.

The hierarchical minimum expected cost strategies in this chapter are *ad hoc*. Three problems were identified:

- The abstracted expected cost is an overestimation of the actual one (example 4.2.6)
- The introduction of κ is not an exact solution to the overestimation problem
- The selection of a value for κ was experimental, will have to be repeated for a different environment and the selected value 1.3 does not have an immediate qualitative explanation.

Despite these problems, the hierarchical minimum expected cost strategy shows the potential for improvement over the unabstracted strategies.

Given that our approach in this chapter was cursory, there are several areas that can be improved. The computation of the expected cost for abstract clusters seems to be the main problem of this approach. In the next two chapters we shall start again working on a strategy that incorporates a hierarchy of the environment but uses its geometrical structure in the clustering and in the computation of the cluster expected costs.

Chapter 5

Path-based Clustering

In chapter 4, it was shown that making a hierarchical description of the environment helped produce better surveillance strategies. However, the method presented there was rather *ad hoc* and did not always improve performance.

One of the problems of the hierarchical strategies presented so far is that the expected cost assigned to the abstract nodes does not closely correspond to the actual cost of visiting one. To be more exact, the expected cost of an abstract node is the sum of the expected costs of its children. However, a 5-step path within an abstract node does not always visit all its children and, further, some children are perhaps visited twice. The parameter κ in the revised hierarchical strategy was introduced to deal with some of the side-effects of this inexact expected cost assignment. Despite being an improvement in some cases, this did not produce better results in all test environments.

A much better assignment of the expected costs can be produced if the geometry of the environment is considered. After discussing some general desiderata for clustering an office building, we will concentrate on our specific case of a corridor-based office building. A better method for assigning expected costs to paths visiting abstract nodes is produced with that instance of an office building in mind.

5.1 Clustering desiderata

A *clustering* is defined to be a graph of connected environment locations fulfilling certain criteria. In general, there are many ways in which one would want to cluster an environment. In our case, the main reason for being interested in clustering is the potential for computational time savings. An abstracted n-step lookahead is prohibitively slow for values of n, large enough to examine every room in our environment before acting. By clustering the environment and approximating the expected cost at higher level nodes, we produce an abstract description of the problem. Then a decision procedure for planning at the simpler higher levels can be produced. Our clustering is driven by the need to produce such a faster decision procedure that computes approximate solutions.

To generate a clustering, a clustering criterion needs to be selected. There are three main possibilities on what could be used to cluster the environment. It is possible to use the *structure* present in the type of the indoor environment considered to guide our clustering. A natural clustering of an office building would be to form clusters consisting of separate floors. Within a floor different corridors can form separate clusters and so on. Another possibility is to let the *costs* guide the clustering. Some environments might have clumps of rooms that are similar, therefore have the same costs in fire presence. A last option is to let the type of permissible *paths* guide the clustering. For example, it might be reasonable to opt for clusters that produce equal path lengths when visited.

In the case of the surveillance application, the types of paths likely to be useful for a robot performing a surveillance are limited by the properties of the surveillance task. For the purposes of a surveillance task a lot of the possible paths can be ignored because they are not efficient in reducing the expected cost. For example, under mild assumptions, a robot should not stay for many steps in one room before moving to another. Typical paths for a robot performing surveillance are: explore a cluster (visit each room in it at least once), transit through it on its way to something interesting (visit only the rooms needed to get through it), ignore it altogether (do not visit anything) or visit a specific location where, for example, the cost is comparatively high (visit just the rooms on the way to that location and out). These task-dependent path preferences can help us both decide what type of clusters to consider and, related to this, decide to only consider specific paths when assigning expected cost to clusters.

Further, there is an interaction between the shape of the selected clusters and the properties of the considered paths. One such property is that of path length. The path length can be defined to be the total number of room visits along the path. The worst and best case scenarios for exploration path length in a cluster are those where the cluster has a star and a linear topology respectively (fig. 5.1). For the case of a star cluster with n rooms the exploration length, if we enter and leave at the centre of the cluster, is $2n$. For the case of a linear cluster the exploration length is n, if we enter from the sides; for loops this is n, no matter where we enter. This can give us an idea on the bounds of how much time is necessary to explore each type of cluster. For the case of transit paths the situation is rather different. To transit a star can be as short as a single room visit if the entrypoints are in the center of the cluster. To transit a loop can be a lot more complicated with up to $n/2$ nodes visited. In fact, for transiting paths the worst case scenario is when the rooms are arranged as a corridor where the path length is n. From the discussion on path lengths it is obvious that when discussing a path within a cluster the entrypoint at which the cluster is entered is relevant.

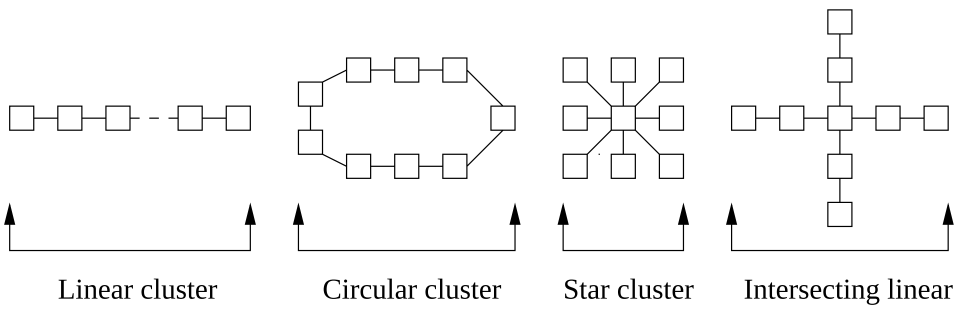


Figure 5.1: Different types of cluster shapes.

The probabilities and costs at the lowest level also play a role in the clustering. An exploration path of fixed allotted time within a cluster assumes that the clusters subnodes are equally important and that therefore staying longer somewhere does not give any benefit. This is definitely the case in office environments where most rooms are equally important offices. However, even within the office environment, one can imagine areas that are more important than others. A fire in the part housing office rooms can be less important than in the area where the computer data storage is housed.

So the boundaries of a cluster could depend on many parameters and in our discussion here we have given a stronger emphasis on paths and costs, because those are more important in our surveillance problem. In the rest of the chapter we work out, in full detail, the office-building example which was given in chapter 4.

5.2 Office-like environments

As mentioned in the last section, office-like environments contain a lot of structure. Offices are normally organised along corridors. These corridors are in turn connected with each other. If they are at different floors they are connected with staircases or elevators. If they are parallel they are connected at some point along their length. If halls are ignored, it is clear that almost any office-like building can be described using corridor shaped structures.

The building blocks of a corridor are, in turn, star-shaped clusters. By connecting the centres of many star-shaped clusters, a long corridor can be formed. The leaves of those clusters correspond to the offices along the corridor. Further, a hall-shaped structure with many rooms connected to it, can be described as a star-shaped cluster with usually a higher number of leaves than a typical corridor star-shaped cluster.

In general, the four different types of cluster depicted in figure 5.1 are relevant within an office building. Linear clusters correspond to corridors, circular clusters correspond to parallel corridors connected at their ends, stars correspond to the

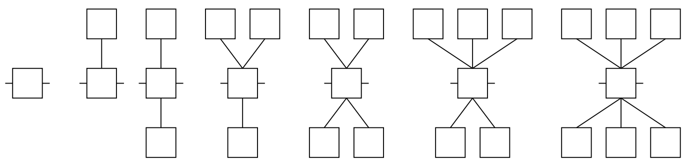


Figure 5.2: Star-shaped blocks of various sizes.

local structure within corridors and crosses correspond to intersecting corridors. The exact geometrical shape of the rooms and corridors can vary from building to building and largely depends on architectural considerations. However, we believe that the topological structure, although different among buildings, has a common basis.

A clustering process can be developed for office-like buildings that treats level 1 clusters as a special kind of cluster. We call level 1 clusters *blocks*. The blocks considered have the shape of a star (fig. 5.2). Of course, several other shapes could be considered, but block clusters of this shape are sensible building blocks of corridor-shaped office buildings. The expected cost of star-shaped blocks can be computed directly without examining individual rooms and this simplifies the computation of the expected cost later in this chapter.

A clustering process can be created for office-like environments. This can be based on repeated clustering until the environment cannot be further abstracted. We propose a clustering process with the following properties:

1. (block-based) star-shaped blocks first have to be found to form level 1 of the abstraction.

2. (connected) a path within the cluster should exist between any two nodes in a cluster.

3. (balanced) the resulting tree should be balanced; this means that at each level the exploration paths within the clusters should have more or less equal length.

4. (uniformity) the fire costs and fire starting probabilities have to be uniform within a block.

The last two criteria deserve further discussion. On the issue of tree balancing several options were available. The tree could be balanced on the number of rooms in each cluster (as in section 4.2.1). It could also be balanced on cost so that all clusters of a certain level have equal expected costs. In the next chapter decisions will be taken between different cluster paths. Choosing to balance the tree on path length makes these comparisons fairer. It is the time others are ignored, instead of the particular cost of a cluster itself, that appears more important in our situation.

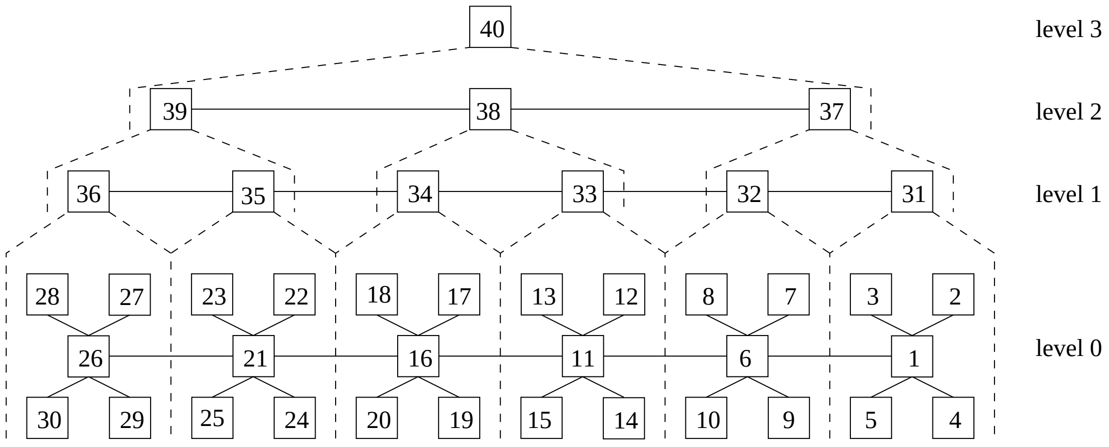


Figure 5.3: A clustered corridor based environment.

The decision for uniformity in the probabilities and costs within star blocks was taken to simplify the computation of block expected costs. The equations for computing the expected cost of a path in a star rely on the assumption that the fire costs and fire starting probabilities are uniform within a block. It should be mentioned however that this assumption is not very essential. If this assumption was not made, the computation of the expected costs at the block level would have to be more complicated.

If we concentrate on an office-like environment with a single corridor an abstraction tree such as that in figure 5.3 can be created. The focus in the rest of this chapter is on an environment with this type of structure. The equations for the derivation of expected cost assume a corridor-like topology in our building. Of course not all office-like buildings are composed of a single corridor, but this will serve in this thesis to investigate the main issues in surveillance planning.

5.3 Route abstraction simplification

The expected cost assigned to a cluster should depend on the route of subclusters followed within the cluster. A specific cluster can be explored using various different routes and not all routes can be expected to incur the same cost. Ideally, all possible routes within the cluster should be examined. A potentially infinite number of routes exists even within a small cluster if revisits are allowed and it is not computationally feasible to consider all those routes. A way of circumventing this difficulty is to examine a few predefined routes within the cluster. Taking the cluster's entrypoints as a basis, the following possibilities are considered:

- **Exploration routes** $r^e_{X_i}$, routes between entrypoints that visit every subcluster.

- **Transit routes** $r^t_{X_i}$, shortest routes between entrypoints.
- **Ignoring clusters**, all the clusters that are neither explored nor transited are ignored.

While the robot is following a transit or an exploration route, all rooms in the route are sensed and, if necessary, their fires are extinguished. The entrypoint of a route is important in a cluster with more than one entrypoints since then, routes of several directions might exist.

When considering a single room both exploration and transit paths just visit the room itself $r^e_{X_i} = r^t_{X_i} = [X_i]$. In the case of star-shaped blocks a reasonable exploration route visits every node in the star once, and a reasonable transit route visits the central node. This single route property of the blocks simplifies matters because, in a sense, blocks can be treated as rooms. A cluster of level $h > 1$ contains many blocks (or subclusters). In the case of our environment the clusters look like corridors of blocks or subclusters and this simplifies matters. A cluster exploration route is a sequence of subcluster exploration routes that begin at one end and finish at the other end of the corridor. For transit routes the situation is similar.

We give an example of route types for the cases of blocks and clusters.

5.3.1. Example. For the case of block 34 (fig. 5.3) such an exploration route is $r^e_{34} = [r^e_{16}, r^e_{17}, r^e_{16}, r^e_{18}, r^e_{16}, r^4_{19}, r^e_{16}, r^e_{20}, r^e_{16}]$. Similarly only one transit route is possible per block, namely that of visiting the center of the block. For the case of block 66 such a transit route is $r^t_{34} = [r^e_{16}]$.

Since clusters 39, 37 each have a single entrylink, only a single exploration route is possible and in cluster 39 such a route is $r^e_{39} = [r^e_{35}, r^e_{36}, r^t_{35}]$. Routes like r^e_{79} can be recursively rewritten to contain just room routes. Cluster 38 has two entrylinks and so two similar exploration routes are possible one for each direction of exploring. For transit routes the situation is similar. For example, for cluster 28 a transit route is $r^t_{38} = [r^t_{33}, r^t_{34}]$.

5.4 Expected cost assignment

Now that a clustering of the environment has been constructed using star-shaped blocks, and standard routes have been assigned to the clusters, we need to assign expected cost specific cluster/route combinations.

To make the equations of this section clearer some shorthand notations are introduced. First, a superscript is used to denote a location's X level h in the abstraction tree. For example, a room will be written as X^0 and a block as X^1. Secondly, p_i is written instead of $P(f_i \rightarrow 1)$ for the probability of a fire starting at location X_i and c_i, instead of $C(\mathbf{f_i})$, for the cost of a fire being present at X_i.

5.4.1 Approximate probability computation

In proposition 3.2.2 the probability $P_t(\mathbf{f_i})$ of the presence of fire at location X_i at a given time since last visit t was defined as:

$$P_t(\mathbf{f_i}) = 1 - (1 - p_i)^t$$

where p_i is shorthand for $P(f_i \to 1)$, the probability of a fire starting during one time-step. The exponential increase of probability $P_t(\mathbf{f_i})$ in time t makes it hard to compute differences of expected costs at different points in time. An approximation of the probability $P_t(\mathbf{f_i})$ is proposed that makes reasoning about the benefit of visiting a cluster easier without affecting the results significantly.

5.4.1. Definition. The approximate probability $\tilde{P}_t(\mathbf{f_i})$ of fire presence at time t is defined as the product of the fire starting probability p_i and time t:

$$\tilde{P}_t(\mathbf{f_i}) = p_i t, \qquad \text{if } t \ll \tfrac{1}{p_i} \tag{5.1}$$

This is not a real probability; the condition of $t \ll \frac{1}{p_i}$ is added to the definition to guarantee that it is not greater than 1. The advantage of this definition over the exact one in proposition 3.2.2 is that $\tilde{P}_t(\mathbf{f_i})$ is linear in t. We will call any computations made using the approximation of equation 5.1 "linear probability approximations". The error of the approximation can be determined directly as:

$$E \;=\; \frac{(1 - (1 - p_i)^t) - p_i t}{1 - (1 - p_i)^t}$$

The equation for E allows us, given some values for p_i and t, to compute the error in our estimation of the probability $P_t(\mathbf{f_i})$. In a concrete example the probability p_i of fires starting would be characteristic of our environment and should be known *a priori*, while the maximum t would be dependent on the size of the environment.

In order to demonstrate that the quality of the approximation is reasonable a graph of E versus p_i and t was plotted (fig. 5.4). Since our environment has 50 rooms, exploring it would take a maximum of 100 time-steps and so values between 1 and 100 were used for the t axis. In fact, 100 is the worst-case number of steps for the case where the environment is a star, while, in our case, it is a corridor of smaller stars and this makes the exploration path smaller. The probability of fire starting p_i used in the simulations was 0.001 but, in reality, smaller values of p_i would be expected. From the graph it can be observed that the error is always below 5%. So for the environment we are considering, but also for more realistic environments, this approximation is reasonable.

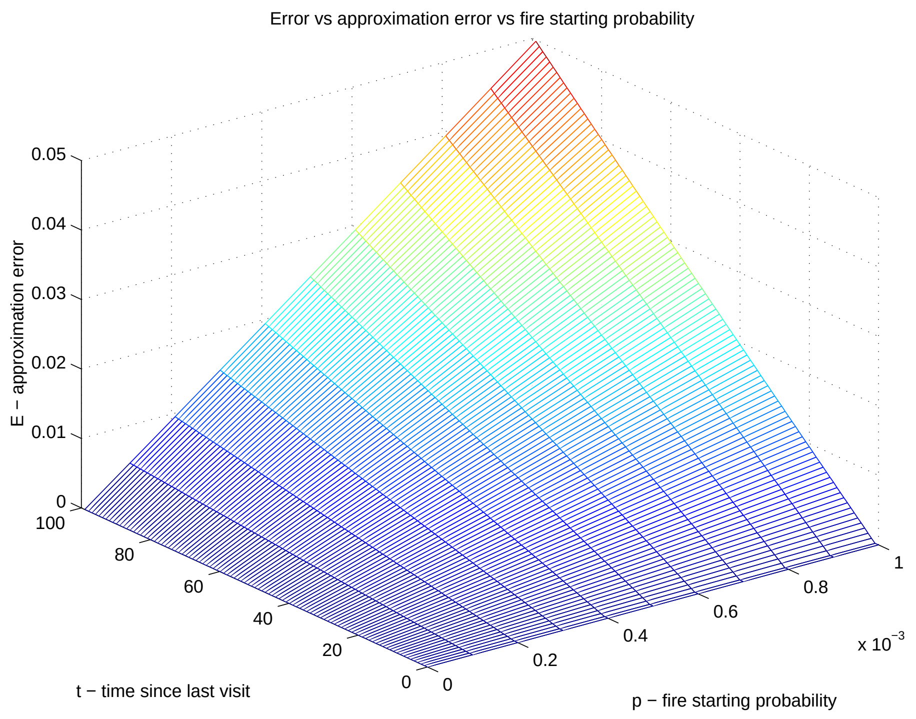


Figure 5.4: Error sensitivity to p_i, t for the approximation of $P(\mathbf{f_i})$

5.4.2 Cost computation

In what follows we give the expected cost computed in the case of rooms, blocks and clusters using the clustering principles of this chapter. Three equations are given for each type of action:

1. $EC_e(X, r^e, T)$ for exploring location X, using exploration route r^e, given a time T since last visit.
2. $EC_t(X, r^t, T)$ for transiting through location X, using transit route r^t, given a time T since last visit.
3. $EC_n(X, I, T)$ for ignoring location X, for I time-steps, given a time T since last visit.

At time t, we need to compute the expected cost of our proposed visiting action (either an exploration or a transit) at a location X. Suppose that the last time location X was visited was T time-steps ago, at time $t' = t - T$. Further assume that a visit to location X takes l time-steps where l is the length of the route corresponding to the visiting action taken. Then, the expected cost of this visit to location X is the sum from time t to time $t + l$ of the expected costs for each of the time-steps in the visit of location X. These individual costs per time-step depend on the time since last visit T. This is seen in figure 5.5. There the

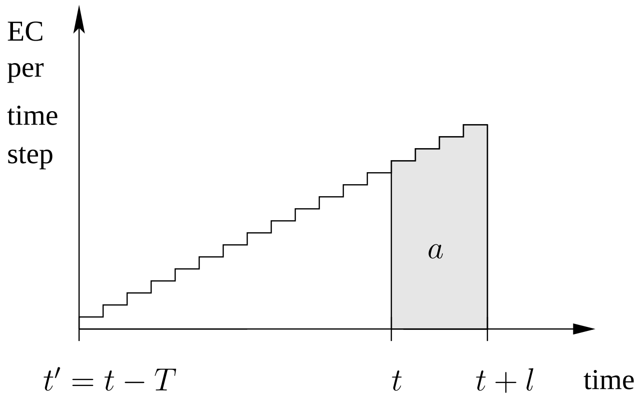


Figure 5.5: The expected costs computed for our actions (area shown in grey).

x-axis corresponds to time-steps and the y-axis corresponds to cost of location X per time-step. The shaded area a is the expected cost EC the rest of this section computes. Costs between t and $t-T$ do not need to be computed, since these are assumed to be costs that are already incurred. Note that we do not talk about t any longer because the computations we will provide depend just on the time since last visit T (when the costs were "reset") and the duration l of the intended visit.

It was already mentioned that only one of the two types of routes can be followed. The robot can either transit through or explore a cluster while everything else is ignored. The expected cost of the entire environment when a location X is explored using path r^e is defined as:

$$EC = EC_e(X, r^e_X, T_X) + \Sigma_{X' \neq X} EC_n(X', l_{r^e_X}, T'_X) \tag{5.2}$$

where $l_{r^e_X}$ is the length of the exploration path r^e_X.

If that location is transited instead of explored this becomes:

$$EC = EC_t(X, r^t_X, T_X) + \Sigma_{X' \neq X} EC_n(X', l_{r^t_X}, T'_X) \tag{5.3}$$

where $l_{r^t_X}$ is the length of the transit path r^e_X.

These environment expected costs are very important. The robot tries to minimise the expected cost of the entire environment. So what is ignored and for how long is just as important as what is visited.

5.4.3 Room expected cost

Focusing on the level of rooms three expected cost equations are again given: one for transiting through, one for exploring and one for ignoring a room. However, the first two are the same because visiting a room or passing through one is the same thing.

When room X^0 with a time since last visit of T_{X^0} time-steps is not visited (ignored) for an extra time-step, an approximate expected cost $EC_n(X^0, 1, T_{X^0}) = (p_{X^0} T_{X^0}) c_{X^0}$ is associated with it. Generally, if room X^0 is not visited for I consecutive time-steps, this incurs an approximate expected cost $EC_n(X^0, I, T_{X^0})$:

$$\begin{aligned} EC_n(X^0, I, T_{X^0}) &= \sum_{t=1}^{I} P_{T_{X^0}+t}(\mathbf{f_{X^0}}) c_{X^0} \approx \sum_{t=1}^{I} p_{X^0}(T_{X^0} + t) c_{X^0} \\ &= \frac{I(2T_{X^0} + I + 1)}{2} p_{X^0} c_{X^0} \end{aligned} \tag{5.4}$$

where T_{X^0} is the time since last visit of room X^0 at the start of the I time-steps.

The exploration route $r^e_{X^0}$ only visits room X^0, and so the robot immediately extinguishes any possible fire in that room. If in proposition 3.2.2 $t = t'$, the time-step that the robot visits the room is considered, then the probability of fire presence in that time-step is $P_t(\mathbf{f_{X^0}}) = 1 - (1 - P(f_{X^0} \to 1))^{t'-t'} = 0$. Hence, the expected cost associated with exploring that room is also:

$$EC_e(X^0, r^e_{X^0}, T_{X^0}) = 0 \tag{5.5}$$

Similarly, the expected cost of a route $r^t_{X^0}$ transiting through a room is:

$$EC_t(X^0, r^t_{X^0}, T_{X^0}) = EC_e(X^0, r^e_{X^0}, T_{X^0}) = 0 \tag{5.6}$$

This is also because only room X^0 is visited by $r^t_{X^0}$.

It is interesting to give again the expected cost of the entire environment for the level of rooms. We will only do it here at this level because paths at the block or cluster levels can have different lengths. However, as we have mentioned room level transits and explorations are indistinguishable. This makes room-level computation of the environment expected cost fairer.

At each time-step only one room X^0 can be explored, or transited through, while every other room $X'^0 \in X$ is not visited. The total expected cost at that time-step of the entire environment is:

$$EC = EC_e(X^0, r_{X^0}, T_{X^0}) + \Sigma_{X'^0 \neq X^0} EC_n(X'^0, 1, T_{X'^0}) \tag{5.7}$$

Because the exploration/transit cost is always 0, this can be simplified to:

$$EC = \Sigma_{X'^0 \neq X^0} (T_{X'^0} + 1) p_{X'^0} c_{X'^0} \tag{5.8}$$

5.4.4 Block expected cost

Having described how expected costs should be assigned to rooms, we proceed to discuss the case of star-shaped block clusters. At section 5.3 we have already described how routes are assigned to clusters. As with rooms, three cases are

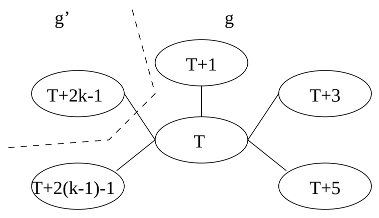


Figure 5.6: A block of size $s = k + 1$ with time T since last visit.

again considered: ignoring, exploring and transiting through a block. A transit through a star-shaped block only visits its central node, while an exploration visits the central node many times on its way to visiting the leaves of the block (as we have shown in example 5.3.1).

5.4.2. Proposition. *(Ignoring after exploration) The expected cost of ignoring a star shaped block X^1 of size s for I time-steps is:*

$$\begin{aligned} EC_n(X^1, I, T) &= (I(sT + 1 + \sum_{i=0}^{s-1} 2i) + s\sum_{i=0}^{I-1} i)pc \\ &= (I(sT + 1 + s(s-1)) + \frac{s(I-1)I}{2})pc \end{aligned} \quad (5.9)$$

where the size s is defined to be the number of rooms in the block, T is the time since last visit of the block, p is the probability of a fire starting, and c is the expected cost of a fire per time step for each room in the block. The assumption is made that p and c are uniform within the block. It is also assumed that the previous visit was an exploration.

Proof.
We do induction on the size s of the cluster.
Prove for $s = 1$
We have, using eq. 5.9, $EC_n(X^1, I, T) = I(T + 1) + \frac{I(I-1)}{2})pc = \frac{I(2T+I+1)}{2}pc$. This is equal to the case of a single room (eq. 5.4), thus correct.
Assume true for $s = k$ Assume that in this case we have: $EC_n(X^1, I, T) = (I(kT + 1 + k(k-1)) + \frac{k(I-1)I}{2})pc$.
Prove true for $s = k + 1$ In this case the expected cost is that of ignoring the first k rooms (call them g) for I steps (the assumed case) + the expected cost of ignoring the $k + 1$ room (call this g') for I steps (see fig. 5.6). In that figure the times since last visit for a star-shaped block are shown. The assumption is that at time T the previous exploration of the block was finished. The times on the leaves in the figure are dependent on the last exploration route followed by the robot. This expected cost can be written as:

$$EC_n(X^1, I, T) = EC_n(g, I, T) + EC_n(g', I, T + 2k - 1)$$

$$= (I((k+1)T + 1 + k(k+1)) + \frac{(k+1)I(I-1)}{2})pc$$

So equation 5.9 also applies in the $k+1$ case. ∎

5.4.3. Proposition. *(Exploring after exploration) The expected cost for exploring a star-shaped block X^1 of size s is:*

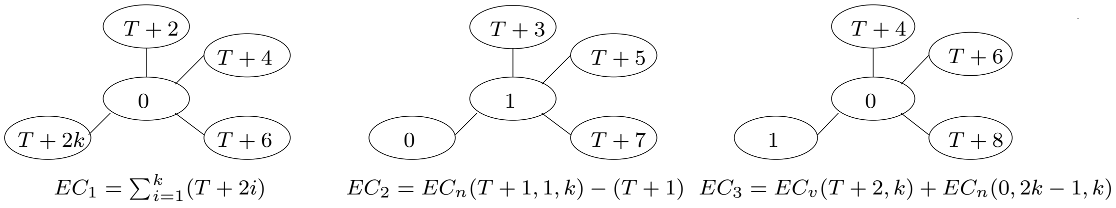
$$EC_e(X^1, r^e_{X^1}, T) = ((s-1)^2T + 2(s-1)^3 + (s-1)^2 + (s-1))pc \qquad (5.10)$$

where s, T, p and c are defined as before and $r^e_{X^1}$ is understood to be the route of length $2s-1$ that visits every leaf node in the star once. Again, it is assumed that p and c are uniform within the block, and that the previous visit was an exploration.

Proof.
Again we provide an induction proof on the size s of the cluster.
Prove for $s = 1$ Again this case is equivalent to having a single room. We have $EC_e(X^1, r^e_{X^1}, T) = 0$ and this is also what you see in eq. 5.5.
Assume true for $s = k$ Assume that in this case we have: $EC_e(X^1, r^e_{X^1}, T) = ((k-1)^2T + (2(k-1)^3 + (k-1)^2 + (k-1)))pc$.
Prove true for $s = k+1$ We first split the block into the part with k rooms g and the new room g'(see fig. 5.6). Suppose that the exploration taken follows the same order as the last exploration of the cluster. Then we split the exploration of the block into three sections and compute the expected costs in conjunction with those parts (see fig. 5.7). The expected EC_1 is that of the first time-step of the exploration k leaf rooms are ignored. The expected cost EC_2 is that of the second time-step of the exploration and can be seen as ignoring g while exploring g'. Finally, the last step EC_3 is that of ignoring g' while exploring g. Writing this out (ignoring the anyhow uniform p, c) gives:

$$\begin{aligned}
EC_e(X_k+1, r^e_{X_k+1}, T) &= EC_1 + EC_2 + EC_3 \\
&= (\sum_{i=1}^{k}(T+2i)) + (EC_n(g, 1, T+1) - (T+1)) \\
&\quad +(EC_e(g, r_g, T+2) + EC_n(g', 2k-1, 0)) \\
&= (kT + k(k+1)) + (kT + k^2 - T) + \\
&\quad ((k-1)^2T + 2(k-1)^3 + (k-1)^2 + (k-1) + (2k-1)k) \\
&= k^2T + 2k^3 + k^2 + k
\end{aligned}$$

The equation thus obtained is essentially eq. 5.10 but for $s = k+1$, so eq. 5.10 is proved.



$T+2$
$T+4$
0
$T+2k$
$T+6$
$EC_1 = \sum_{i=1}^{k}(T+2i)$
$T+3$
$T+5$
1
0
$T+7$
$EC_2 = EC_n(T+1,1,k) - (T+1)$
$T+4$
$T+6$
0
1
$T+8$
$EC_3 = EC_v(T+2,k) + EC_n(0,2k-1,k)$

Figure 5.7: Proof of eq. 5.10.

The assumption that the exploration order is the same as that of the last exploration is not essential to the proof. Even if a different order is taken, the differencing in T in combination with the approximation of the probability gives the same result. ∎

5.4.4. Definition. (Transit after any) The expected cost of transiting through a block X^1 of size s is defined to be the expected cost of ignoring it for the time it takes to transit through it. Since a transit route $r^t_{X^1}$ for star-shaped blocks only visits the central node of the star, passing through it only takes one time-step. The expected cost of a transit route is then:

$$EC_t(X^1, r^t_{X^1}, T) = EC_n(X^1, 1, T) \tag{5.11}$$

Computing the transit cost by ignoring the block is essentially equivalent to saying that a robot transiting a cluster is moving through it with "its eyes closed".

5.4.5. Definition. (Exploration and ignoring after transit) The equations for exploration after exploration (eq. 5.10) and ignoring after exploration (eq. 5.9) are also used for these cases.

We will use equations 5.9, 5.10 even when transits are considered. In those propositions, the assumption was made that the previous visit was an exploration. If that is the case, the expected cost computed by equation 5.10 is a "linear approximation" for $p \to 0$. With the introduction of transit routes the state of a block is not necessarily that of figure 5.6 when an exploration begins. In that figure, the times since last visit in each of the nodes are left as if the last visit was an exploration and this figure is used in the proof of "linear approximation" of equations 5.10 and 5.9.

To make everything "linearly approximate", we would have to consider the interaction between the types of routes at the computation of block expected costs. However, this would yield complicated equations and would evolve many cases based on the type of last visits to a cluster. Since this would be complicated, the decision was made to sacrifice strict "linear approximateness" and to opt instead for a definition of transit where the robot is essentially moving with "its eyes closed". Although this is still an approximation, it will be shown in the next chapter that the results obtained in this way are very good.

5.4.5 Higher-level cluster expected cost

Now that the equations for the block case have been defined, we can proceed by defining how the expected cost should be computed for the case of higher-level clusters. Equations 5.12, 5.13 and 5.14 have to be taken to be correct by definition.

5.4.6. Definition. Ignoring a cluster X^h of level $h > 1$ for I time-steps gives an expected cost that can be computed as:

$$EC_n(X^h, I, T) = \sum_{X^{h-1} \in children(X^h)} EC_n(X^{h-1}, I, T_{X^{h-1}}) \tag{5.12}$$

where $children(X^h)$ are the subclusters of cluster X^h, and $T_{X^{h-1}}$ is the time since last visit for cluster X^{h-1}. So ignoring a cluster is the sum of ignoring its subclusters.

5.4.7. Definition. Exploring corridor cluster X^h of level $h > 1$ using a route $r^e_{X^h} = [r^e_{X_1^{h-1}}, r^e_{X_2^{h-1}}, \ldots, r^e_{X_n^{h-1}}]$ gives an expected cost that can be computed using:

$$\begin{aligned} EC_e(X^h, r^e_{X^h}, T) \quad = \quad & \sum_{i=1}^{n}(\sum_{j=1}^{i-1} EC_n(X_i^{h-1}, l_j, T_{X_i^{h-1}} + \sum_{k=1}^{j} l_k) + \\ & EC_e(X_i^{h-1}, r^e_{X_i^{h-1}}, T_{X_i^{h-1}} + \sum_{k=1}^{i-1} l_k) + \\ & \sum_{j=i+1}^{n} EC_n(X_i^{h-1}, l_j, \sum_{k=i+1}^{j-1} l_k)) \end{aligned} \tag{5.13}$$

where $T_{X_i^{h-1}}$ is the time since last visit for subcluster X_i^{h-1}, X_i^{h-1}, X_1^{h-1}, X_2^{h-1}, …, $X_n^{h-1} \in children(X^h)$ are the subclusters of X^h, and l_k is the exploration route length of cluster X_k^{h-1}.

In this equation, a cost is added for each subcluster. This corresponds to what happens in each subcluster during an exploration route. A subcluster is first ignored $i-1$ times (while other clusters are explored), then explored, then ignored again $n-i$ times (while other clusters are explored). The position of the cluster in the route is important in deciding how much it is ignored and when. For instance, during the jth time the cluster is ignored, it is ignored for l_j room-level time-steps. This is because l_j is the exploration path length of cluster j in the path. This definition makes the assumption that the blocks are organised in a corridor, which is true in the case of the our environment but not in all cases.

5.4.8. Definition. Transiting through cluster X^h of level $h > 1$ using a route $r^t_{X^h} = [r^t_{X_1^{h-1}}, r^t_{X_2^{h-1}}, \ldots, r^t_{X_n^{h-1}}]$ gives a cost that can be computed using:

$$\begin{aligned} EC_t(X^h, r^t_{X^h}, T) \; = \; & \sum_{i=1}^{n} (\sum_{j=1}^{i-1} EC_n(X_i^{h-1}, l_j, T_{X_i^{h-1}} + \sum_{k=1}^{j} l_k) + \\ & EC_t(X_i^{h-1}, r^t_{X_i^{h-1}}, T_{X_i^{h-1}} + \sum_{k=1}^{i-1} l_k) + \\ & \sum_{j=i+1}^{n} EC_n(X_i^{h-1}, l_j, \sum_{k=i+1}^{j-1} l_k)) \end{aligned} \tag{5.14}$$

where $X_1^{h-1}, X_2^{h-1}, \ldots, X_n^{h-1} \in children(X^h)$ are the subclusters of X^h, $T_{X_i^{h-1}}$ is the time since last visit of subcluster X_i^{h-1} and l_k is the transit route length of cluster X_k^{h-1}.

So the only difference in the expected cost computation between transiting through X^h and exploring X^h is that the central term in the sum EC_t in the case of transits gets replaced by EC_e in the case of explorations.

Knowing the cluster structure, which includes all probabilities, costs and routes, and the equations of sections 5.4.4 and 5.4.5 is enough to compute the expected cost for any cluster/route combination. In fact, the equations about rooms in section 5.4.3 are not necessary, since they are special cases of the equations for blocks for $s = 1$. The definitions of section 5.4.5 treat the block expected cost computation as a closed blackbox computation. This implies that our corridor could consist of blocks of a different shape to stars without any need to modify the equations for computing the expected cost within the corridor.

5.5 Properties of expected cost equations

Several observations can be made about the use of star-blocks to structure the environment graph and the computation of expected cost:

1. For corridor clusters, the entrylinks used in the visit are important and are considered in the computation of expected cost. Routes visiting the rooms in the cluster in an opposing order produce different costs.

2. The probability of fire starting and the cost of room fire have to be kept uniform at the block level but not necessarily at higher levels. This is not an important limitation. For a different distribution of expected cost the equations of block expected cost would have to become a bit more involved.

3. The block expected cost is computed directly and then accumulation of costs (i.e. summations in eqns. 5.12, 5.13, 5.14) is only used for clusters. This makes expected cost computation slightly faster.

4. The available routes within each cluster are stored together with the cluster. When a route cost needs to be evaluated they can be retrieved from the cluster information.

The expected cost of transiting through, ignoring or exploring a cluster is a linear function its time since last visit T. More formally:

5.5.1. Proposition. *For a cluster X^h of any level h, the expected cost of transiting through, ignoring, or exploring a cluster can be written as an equation of the form $EC(X^h, I, T) = aT + b$.*

The exact parameters a and b depend on the level of the cluster and its shape, size etc, but the proof of this proposition is relatively easy.
Proof.
The proof is by induction. For the case of ignoring a cluster:
Prove for $h = 1$ At the block level, eq. 5.9 can also be rewritten to be of the form $aT + b$.
Assume true for $h = k$ Assume that the equations for clusters also have the form $aT + b$
Prove true for $h = k + 1$ Then for the collection of blocks of level $h = k + 1$ we have using eq. 5.12 that, $EC_n(X^{k+1}, I, T) = \sum_{X^k \in children(X^{k+1})} EC_n(X^k, I, T_{X^k})$. But then, in turn, this equation is a sum of linear equations and thus also linear.

The proofs for transiting through and exploring a cluster work in the same way and are not reproduced here. ∎

Further, another important observation is that the way in which clusters are decomposed into subclusters is no longer significant. As proof of this statement we consider the simpler possible case of subcluster decomposition in our following proposition. It should be clear that this can be extended to more complicated cluster decompositions.

5.5.2. Proposition. *Given three clusters X_1, X_2, X_3, that eventually comprise cluster X_5 (see fig. 5.8). The expected cost computation for exploring, transiting or ignoring cluster X_5 is affected neither by introducing an extra cluster X_4 in a position between X_5 and X_1, X_2, X_3 nor by which subclusters are contained within the introduced cluster X_4.*

Proof. For the case of ignoring cluster X_5 in the situation of fig. 5.8(a) as:

$$\begin{aligned} EC_n(X_5, I, T) &= EC_n(X_4, I, T_{X_4}) + EC_n(X_3, I, T_{X_3}) \\ &= EC_n(X_1, I, T_{X_1}) + EC_n(X_2, I, T_{X_2}) + EC_n(X_3, I, T_{X_3}) \end{aligned}$$

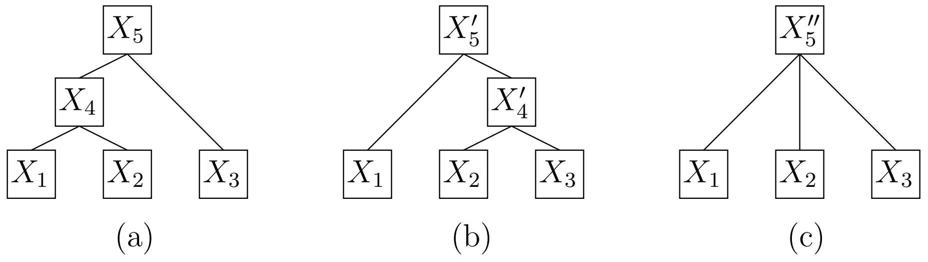


Figure 5.8: Different groupings.

For the case of ignoring cluster X_5 in the situation of fig. 5.8(b) the EC is:

$$\begin{aligned} EC_n(X'_5, I, T) &= EC_n(X_1, I, T_{X_1}) + EC_n(X'_4, I, T_{X'_4}) \\ &= EC_n(X_1, I, T_{X_1}) + EC_n(X_2, I, T_{X_2}) + EC_n(X_3, I, T_{X_3}) \end{aligned}$$

For the case of ignoring cluster X''_5 in the situation of fig. 5.8(c) the EC is:

$$EC_n(X''_5, I, T) = EC_n(X_1, I, T_{X_1}) + EC_n(X_2, I, T_{X_2}) + EC_n(X_3, I, T_{X_3})$$

The substitutions given above are due to definition 5.12 on the cost of ignoring a cluster. So the expected cost is the same and hence the position or presence of the cluster X_4 does not matter. The proofs for the case of explorations and transits work in a similar way. ∎

5.6 Summary

In this chapter we discussed mainly the use of star-shaped blocks in the abstraction. This rather specific shape is at the basis of the description of any office building, so explicit formulas were developed for ignoring, exploring and transiting through star-shaped blocks. If one liked to choose other primitives for a building, introducing a new cluster shape would only affect the computation of the expected cost at the block level since new equations have to be introduced for the new shape. The accumulation of expected costs at the cluster level should remain the same provided, of course, that the case of a corridor environment is considered.

The linearity property makes comparison of expected costs and predictions of their evolution simpler. It is a property which can simplify the process of decision-making between clusters. We also proved a property on the interaction between the abstraction hierarchy and the expected cost assignment which states that the assignment is independent of the exact ordering of the clusters over the block level. This suggests robustness of the proposed method against arbitrary decisions on the position of cluster boundaries and this is something we will confirm in the next chapter.

Chapter 6

Path-based Decisions

In chapter 5 it was decided to cluster the environment based on paths and to compute the expected costs for specific cluster-route combinations. It was hoped that this new clustering would enable us to design a decision procedure that best corresponds to the optimal but is still computable. This chapter treats the fixed cluster route strategy which works on this path-generated abstraction of the environment.

6.1 New clustering of our office building

Before starting with the description of the new decision procedure, a revision is proposed of the clustering that was presented in section 4.2.1, for example 4.1.1. The new clustering stems from the considerations presented in the previous chapter and its novelty lies in the introduction of a lowest clustering unit at which the expected costs and "atomic" paths are specified. This new clustering is used in the simulation results that will be presented later.

The new clustering of the office building of example 4.1.1 treats level 1 clusters as star-shaped *blocks* (fig. 5.2), in accordance with our observation in section 5.2 that these are sensible, lower level building blocks for any office-like building. The central nodes in the star blocks correspond to corridor locations while the leaves of the star blocks correspond to rooms.

The clustering is generated by repeatedly clustering the room nodes until the environment cannot be further abstracted. The aim is to produce the abstraction tree (see fig. 6.1). The clustering process can be automated using the clustering criteria of section 5.2. However, the clustering presented here was created manually.

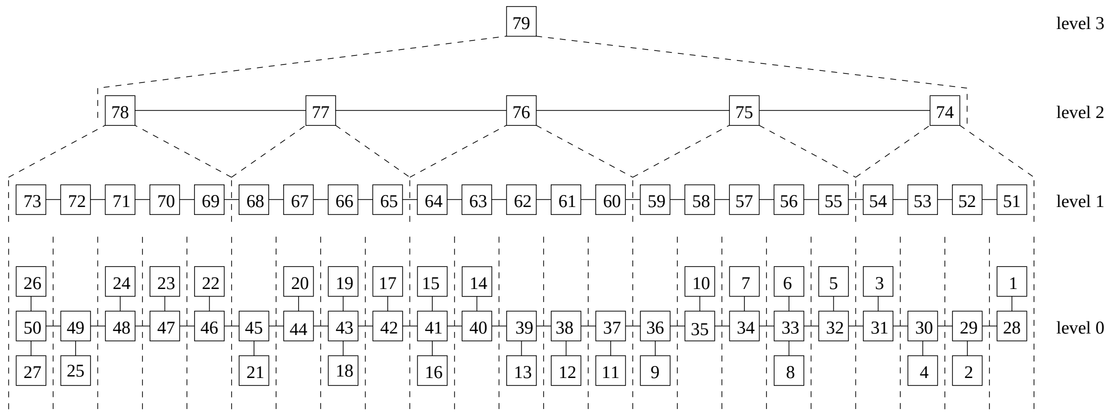


Figure 6.1: The new block-based abstraction tree.

6.2 The fixed cluster route strategy

6.2.1 Overview

We now define a new decision strategy for such an abstracted building called the **fixed cluster route** strategy. The full decision procedure is summarised as pseudocode in figure 6.2. In essence, the fixed cluster route strategy computes the cluster expected cost for a predefined route within the cluster instead of giving a heuristic estimate of the cost for all possible routes within the cluster. The robot then has to stick to the predefined route it selected by comparing the expected costs. Although the computation of the route cost is more accurate than before, the method is still heuristic in that not all routes are examined. The expectation, however, is that examining the expected cost of predefined routes while forcing the robot to follow those routes produces a strategy that approximates to the ideal infinite horizon minimum expected cost strategy.

This can be compared to the decision procedure of the hierarchical minimum expected cost strategy of figure 4.4. Our strategy for making decisions at the decision level can be split into two parts. The first part combines routes of the lower level clusters in order to produce the options available at the higher levels. These options can then be evaluated in the manner described in the previous chapter. The second part uses the evaluation of these options to make decisions on which option should be chosen. These are no longer between clusters, as in chapter 4, but between cluster-route combinations.

6.2.2 Candidate trajectory generation

In the last chapter two types of cluster routes were considered: explorations and transits. A *trajectory* is defined to be a concatenation at the same level

```
$look\_ahead \leftarrow NUMBER$;
% where NUMBER is the maximum number of rooms in a trajectory
% and was determined in section 6.2.4.
$present\_room \leftarrow ROOM$;
% where ROOM is the room the robot is currently in
$decision\_level \leftarrow LEVEL$ ;
% where LEVEL is the bottommost level where enough computationally
% tractable but maximally representative choices are available
% Determined at the end of section 6.2.2.

$present\_node \leftarrow$ map $present\_room$ to $decision\_level$ node;
$trajectory\_list \leftarrow$ fixed cluster route concatenations at $decision\_level$
                  starting from $present\_node$ with length less
                  than $look\_ahead$ rooms ;
for $trajectory \in trajectory\_list$ do
    $cost[trajectory] \leftarrow$ expected cost of $trajectory$
end
$selected\_trajectory \leftarrow$ select $trajectory$ with lowest $cost[trajectory]$;
follow $selected\_trajectory$ until its end is reached;
$present\_room \leftarrow$ end room in $select\_trajectory$;
repeat the decision procedure to select new trajectory;
```

Figure 6.2: Algorithm for the fixed cluster route strategy.

of cluster routes of these two types. These concatenated trajectories form the options available for our strategy.

When combining subcluster routes to form cluster routes or when combining cluster routes to form trajectories, special care has to be taken to ensure that all route concatenations are valid. The trajectories should not contain any "jumps", that is the robot should not consider moving in one time-step between locations that are not adjacent. To guarantee that, cluster entrylinks are given a special status in the decision procedure.

A link $(X_i, X_j) \in A$ is a member of the set of entrylinks A^X of cluster X if:

$$\begin{array}{rl} \text{either } X_i \in children(X) & \text{and } X_j \notin children(X) \\ \text{or } X_i \notin children(X) & \text{and } X_j \in children(X) \end{array}$$

The possible routes R^X within cluster X are found by focusing on its subclusters $children(X) = \{X_1, \ldots, X_n\}$ and forming combinations using routes from $children(X)$ of length equal to the cardinality of $children(X)$. However, not all such combinations are guaranteed to be valid robot routes and they have to be pruned using the following constraints:

- If $R_x = [X_s, \ldots, X_e]$ is a new route of R_X, then it should be that $(\ldots, X_s) \in A^X$ and $(X_e, \ldots) \in A^X$. So the start and end points of a new route should be entrylinks to the cluster.

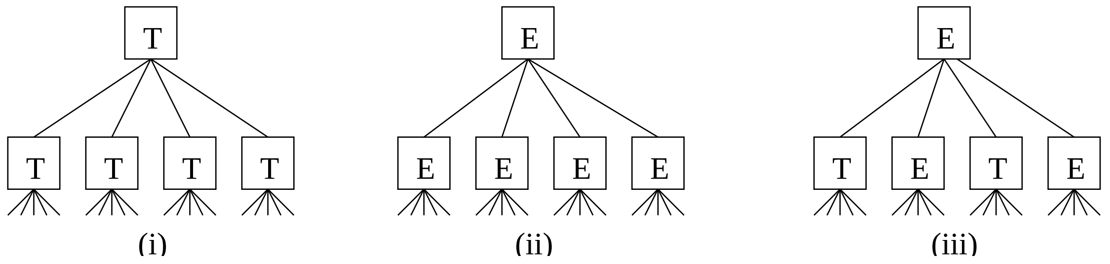


Figure 6.3: Ways of combining transits (T) and explorations (E) at the cluster level. (i) and (ii) are permitted. Permutations of T and E are not permitted (iii).

- If $[\ldots, X_e, X'_s, \ldots]$ is a new route of R_X formed by the concatenation of $R_{X'} = [X_s, \ldots, X_e]$ and $R_{X''} = [X'_s, \ldots, X'_e]$, where $X', X'' \in X$, then $(X_e, X'_s) \in A$. Or in words: for every possible route concatenation the startpoint of the second route must be accessible from the endpoint of the first route.

- If two routes $R_{X'}$, $R_{X''}$, $X', X'' \in children(X)$ are concatenated, they should *both be of the same type*, i.e. both either exploration or transit (see fig. 6.3).

The reason why the third constraint is wanted is computational. If this constraint was not present many combinations of subcluster routes would be offered as choices to the decision procedure (fig 6.3:iii). The third constraint limits the number of choices per cluster offered to the decision procedure to 3 (ignore, explore and transit). It can be justified for environments with uniform costs because there is no local cost-based reason to discriminate among subnodes. Offering just 3 choices is enough for that case. For non-uniform environments it is not fully justified, but we choose the clusters such that a high degree of cost uniformity still exists at the cluster level; then even in that level the above argument applies.

6.2.3 The decision level

In the fixed cluster route strategy, the decision theoretic comparison, and the trajectory generation, only take place at one specific abstraction level, which is called the *decision level*. This is again done mainly for computational reasons. A natural choice for such a level to take decisions exists because the costs in the clusters should be almost uniform and because the routes in any level are not be allowed to be combinations of explorations and transits. We decided to use the bottommost level where enough computationally tractable but maximally representative choices are available (e.g. level 2 in fig. 6.1).

Following the identification of the decision level, the collection of trajectories can be automatically generated using the method just described for single clusters.

It is possible to combine decision level routes to form many-step trajectories that satisfy the above-mentioned constraints. At the level of trajectories the final constraint about concatenated route types is lifted. A trajectory can contain a combination of explorations and transits. It is only important that the routes are not mixtures of transits and explorations to guarantee that the number of options within a cluster is limited (fig. 6.3).

6.2.4 Trajectory selection

Given that the robot is in a specific entrylink on the boundary between clusters at the decision level, several actions are available based on which trajectories begin from its current location. In the n-step look-ahead case, possible trajectories of routes are computed to the depth of the desired look-ahead and the best is selected. The trajectory which produces the lowest environment expected cost is considered to be the best trajectory. Once a trajectory is selected, the robot should commit itself to the selected route or trajectory.

The most obvious look-ahead limit is, perhaps, to look ahead a specific number of cluster actions, whether explorations or transits. Then, all the trajectories evaluated would be visiting a fixed number of clusters starting from the current entrylink.

However, this does not hold, not even in balanced trees, because the exploration route length is significantly greater than a transit route length and this makes computation biased. Taking cluster 77 in figure 6.1 as an example, cluster 77 has an exploration length of 14 visits, which correspond to 14 time-steps, while a transit just takes 4 time-steps. In the case of two 8-step trajectories, one just exploring cluster 77 eight times and one just transiting it eight times, a significant difference in path length develops ($14 * 8 - 4 * 8 = 80$). Such differences develop, even when just considering exploration routes, because the tree can never be completely balanced. A route of greater length typically results in greater environment expected cost and would not be preferred even though perhaps it should.

To focus on the last statement, routes of larger length do produce, most of the time, higher costs. This can be founded on equation 5.8, which describes how the expected cost of the environment evolves during one time-step. This says that the expected cost EC of the entire environment is the expected cost EC_e of the room X^0 currently being explored plus the sum of the expected costs EC_n for all the ignored rooms.

$$EC = EC_e(X^0, r_{X^0}, T_{X^0}) + \Sigma_{X'^0 \neq X^0} EC_n(X'^0, 1, T_{X'^0})$$

Since, the exploration cost EC_e for rooms is 0, this can be rewritten as:

$$EC = \Sigma_{X'^0 \neq X^0} EC_n(X'^0, 1, T_{X'^0})$$

Now, if a trajectory k of length l is used to visit many rooms X^0 in some sequence, the expected cost EC_k of the trajectory is:

$$EC_k = \Sigma_{X^0 \in k} \Sigma_{X'^0 \neq X^0} EC_n(X'^0, 1, T_{X'^0})$$

So the expected cost of the trajectory is reduced to the sum of ignored rooms per time step. The longer the trajectory, the longer some rooms are inevitably ignored and the higher, most of the time, the expected cost of the trajectory.

To make fair comparisons then, it is important, just as we did while clustering, to *focus on balancing the room-level lengths of trajectories.* Since we want trajectories to be combinations of explorations and transits at the decision level, the solution is to look ahead a specific number of time-steps rather than a specific number of cluster visits (whether explorations or transits). Picking a large time horizon, we can try to fit within it as many permutations of transits and explorations as possible that result in a trajectory-length as close as possible to the limit. Then a choice is made among those trajectories.

The "candidate" trajectories are automatically generated as follows: starting from a specific entrylink a trajectory begins with the first possible route also starting there. Then more routes are concatenated until the length of the total trajectory is over 30. These are considered to be "candidate". By back tracking the space of possible trajectories, they can all be found automatically.

6.2.1. Example. We give some examples of possible trajectories. Assume the look-ahead is set at 30 rooms and that we start at the link between clusters 77 and 78 in figure 6.1. A possible trajectory is $[r^e_{78}, r^t_{77}, r^t_{76}]$, which first explores cluster 78 and then transits clusters 77,76. This trajectory has a length of exactly 30 rooms. An example of a trajectory that is not allowed is $[r^e_{78}]$ since, this is shorter than 30 rooms. An example of a trajectory that is unintuitive but is allowed is the one that transits cluster 77 eight times. Although it is unlikely that this trajectory will ever be selected, it has a total length of 32 which is over 30. However, transiting cluster 77 seven times is not a "candidate" trajectory because its length is just below 30.

The fixed cluster route strategy needs to be compared to the n-step minimum expected cost strategy and to the hierarchical minimum expected cost strategy. In chapter 4, a 5-step look-ahead was used for these strategies. The exploration routes in the level two clusters of the abstraction tree in fig. 6.1 have a length of about 15 rooms. As a consequence, the decision was taken to use a look-ahead of 75 time-steps for the fixed cluster route strategy because 75 time-steps roughly correspond to setting a lower bound of 5 cluster level decisions. The upper bound is however much higher. Assuming that a level 1 transit takes 5 steps, up to around 15 transits can be considered.

6.3 Simulation results

In this section, the fixed cluster route strategy is compared with the minimax interval, the 5-step minimum expected cost, and the revised hierarchical strategies of section 4.2.3. All strategies were simulated in the office building environment of example 4.1.1. For the revised hierarchical strategy, κ was set at 1.3 as in section 4.2.3. The comparison was for a run of 10.000 iterations; in fact, the methodology of section 4.1.3 was followed to collect the simulation data.

6.3.1 Cost-based comparison

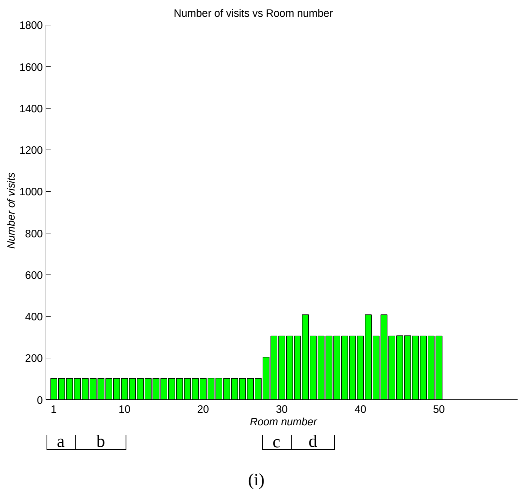
The actual costs after 10000 simulated iterations of example 4.1.1 are summarised in units of 10^3 in table 6.1. Four different values of $\hat{c}$, the fire cost for rooms "past" the barrier, were used. Further, a "uniform" environment setting with no cost "barrier" helps determine what happens when the cost differences are unimportant.

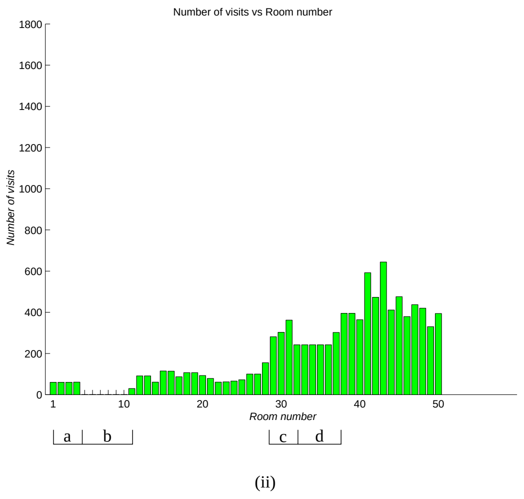
	minimax interval	5-step MEC	revised($\kappa = 1.3$) hierarchical MEC	fixed cluster route
uniform	**19.6±1.2**	22.6 ± 1.4	25.8 ± 1.1	$23.8 \pm .66$
$\hat{c} = 1$	15.3 ± 1.1	81.4 ± 2.8	18.9 ± 1.2	**13.9±.62**
$\hat{c} = 10$	47.0 ± 5.3	734 ± 26	64.4 ± 3.6	**42.8±2.1**
$\hat{c} = 50$	196 ± 24	$3,640 \pm 130$	145 ± 11	**133±6.0**
$\hat{c} = 100$	371 ± 53	$7,250 \pm 210$	329 ± 6.6	**190±5.8**

Table 6.1: Total costs and standard deviation after 10000 iterations in units of 10^3. Best costs for each environment in bold.

From the table it is clear that the fixed cluster routes strategy outperforms the others in all cases where a barrier is present, irrespective of what value $\hat{c}$ takes. These cases were the most interesting because it is there that the decision theoretic comparison becomes important. It is also a clear improvement on the naive revised hierarchical strategy which does not beat minimax interval for $\hat{c} = 1$ and $\hat{c} = 10$. In the remaining case, where the costs and probabilities are uniform, minimax interval outperforms all expected cost-based strategies.

The minimax interval strategy always follows the same route, no matter what the costs or probabilities are. Its behaviour is only dependent on the times since last visit and the connectivity of the environment. It is, therefore, close to optimal for the specific case of uniform costs. In the case where probabilities and costs are uniform, simple 1-step minimum expected cost reduces to minimax interval; this was proven in proposition 3.2.3. In table 6.1, 5-step minimum expected cost is worse than minimax interval and consequently worse than the 1-step minimum expected cost behaviour, which coincides with that of minimax interval.





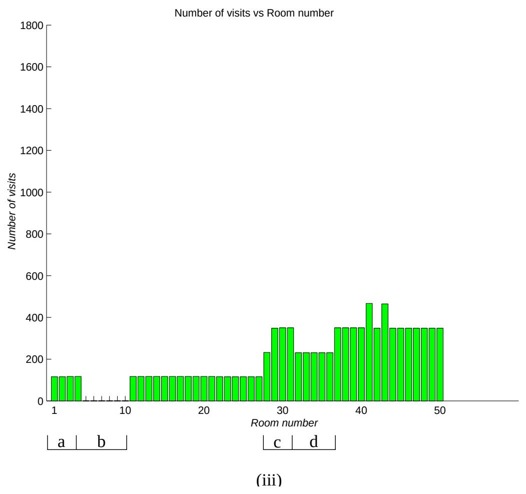
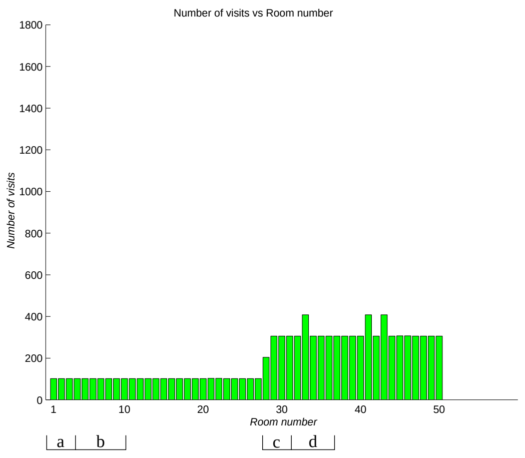
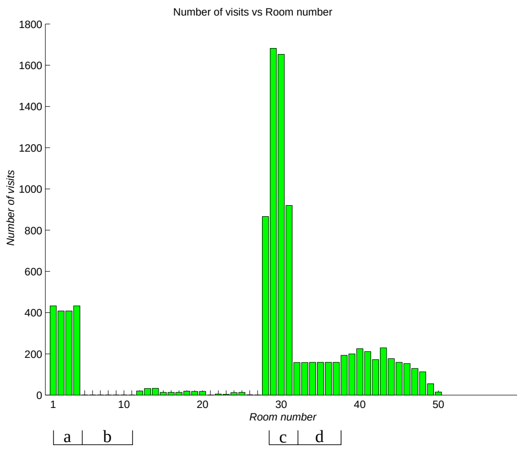


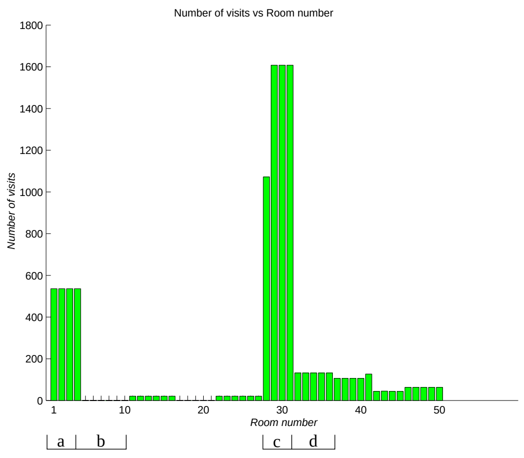
Figure 6.4: Number of room visits vs. room number using $\hat{c} = 1$: (i) minimax interval, (ii) revised hierarchical minimum expected cost, (iii) fixed cluster routes.



(i)

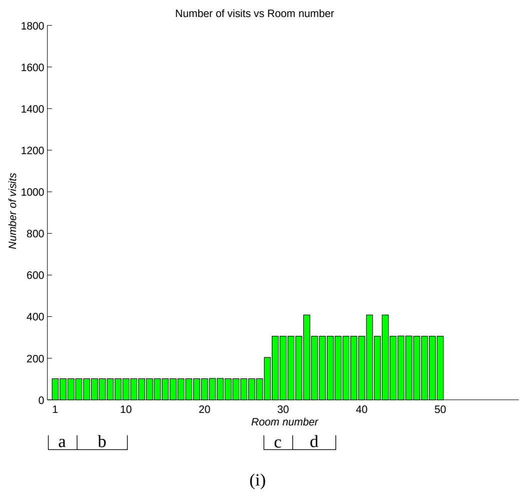


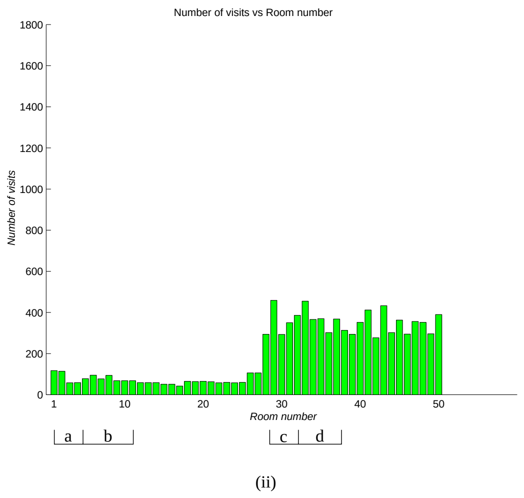
(ii)



(iii)

Figure 6.5: Number of room visits vs. room number using $\hat{c} = 50$: (i) minimax interval (ii) revised hierarchical minimum expected cost (iii) fixed cluster routes.





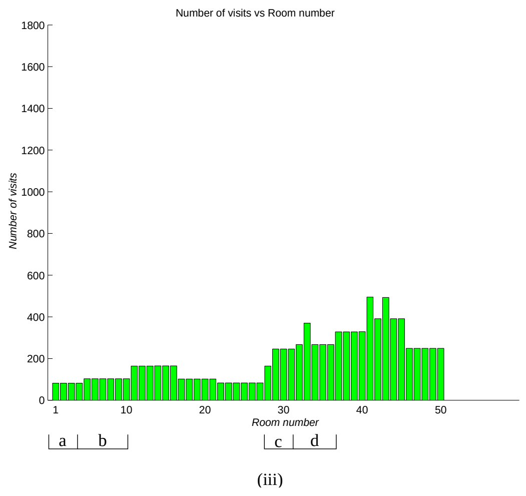


Figure 6.6: Number of room visits vs. room number using the “uniform” setting: (i) minimax interval, (ii) revised hierarchical minimum expected cost, (iii) fixed cluster routes.

In figures 6.4, 6.5, 6.6, the number of visits per room are plotted for the $\hat{c} = 1$, $\hat{c} = 50$ and "uniform" settings respectively. These figures, are included to give some insight into the type of the trajectories chosen, but the order of visits within the trajectories is not visible. Using the figures, the room preferences of the three main strategies, namely minimax interval, revised hierarchical minimum expected cost, and fixed cluster routes can be compared. Four groupings of rooms are marked at the bottom of each graph. These groupings correspond to those of figure 4.6 of chapter 4. Grouping a contains the rooms "past" the barrier, grouping b contains the rooms on the barrier, grouping c contains the corridor locations "past" the barrier and grouping d contains the corridor locations on the barrier. Before discussing each cost setting separately, it can be said that all the plots for the minimax interval strategy (subfigures (i)) are the same. This again shows that minimax interval is not adapting to cost changes.

The case where $\hat{c} = 1$ is a situation where the new strategy wins over minimax interval and the revised hierarchical strategies. In figure 6.4 the fixed cluster route strategy can be seen to ignore the rooms that are unimportant (grouping b in the figure), but still transits through the corridor rooms there (grouping c in the figure); so the transit routes were used. The revised hierarchical strategy also ignored the unimportant rooms. The improvement on the revised hierarchical strategy is due to the regularity of the visits in the rest of the environment. The fixed cluster routes and minimax interval strategies demonstrate a geometry-dictated pattern. The minimax interval strategy performs worse for $\hat{c} = 1$ because the unimportant rooms (grouping b) are visited although $c = 0$ there.

For the case where $\hat{c} = 50$ the fixed route strategy slightly improves performance upon that of revised hierarchical strategy (table 6.1). Both seem to be taking very similar decisions but the fixed route strategy displays again a more regular behaviour and further spends more time in the costly part of the environment (groupings a and c). The fixed cluster route strategy again does not visit grouping b corresponding to the barrier rooms. In all cases where a barrier is present, the fixed cluster route strategy correctly applies transfer paths to cross the barrier. This reduces the expected cost in relation to the minimax strategy, which still visits those rooms.

In the case of the "uniform" setting, the minimax interval strategy outperforms the minimum expected cost strategies. The revised hierarchical is clearly very "noisy" in its choices (fig 6.6:ii). This is due to its *ad-hoc* nature. In this setting, the fixed cluster route strategy is again more regular(fig 6.6:iii). The pattern of visits is geometrically influenced in the same way as for minimax interval. The "peaks" in the room visits correspond to corridor locations of high connectivity that need to be visited more often on the way to real rooms. The main problem of the fixed cluster route strategy is that the central cluster (cluster 76 in figure 6.1) is explored more often (it is a lumpy minimax interval). That the central node is explored rather than transited becomes obvious from the fact that the rooms of that cluster (rooms 11-16) are visited more than the rest of the rooms (rooms

1-10,17-27).

The irregularity in cluster visits by the fixed cluster route strategy that is visible in figure 6.6:iii is a type of discretisation noise. It originates from the cluster boundary choices and the limited trajectory length. Since the clusters are uniform in cost, there is no reason to prefer one cluster to another apart from these small noise generated differences. So the noise dominates the choices of the fixed cluster route strategy. In the "barrier" cases, the cost differences are greater than the noise and can guide the behaviour as should be done if we were approximating the optimal one.

6.3.2 Sensitivity to clustering

To assess the effect of the abstraction on the behaviour of the fixed cluster route strategy, a simulation experiment on a modified star-based abstraction tree was performed. In the modified abstraction tree, two of the decision level clusters are joined to form a new larger cluster (fig. 6.7). The new abstraction tree is no longer balanced at that level, since the new cluster 76 is much larger.

	minimax interval	fixed cluster route	Modified clustering
uniform	**19.6±1.2**	23.8 ± .66	20.7 ± 1.1
$\hat{c} = 1$	15.3 ± 1.1	13.9 ± .62	**13.5±.76**
$\hat{c} = 10$	47.0 ± 5.3	42.8 ± 2.1	**40.6±2.2**
$\hat{c} = 50$	196 ± 24	133 ± 6.0	**123±7.2**
$\hat{c} = 100$	371 ± 53	190 ± 5.8	**184±8.4**

Table 6.2: Total costs and standard deviation after 10000 iterations in units of 10^3.

The fixed route strategy was reapplied to this abstraction tree and the resulting costs are summarised in units of 10^3 in table 6.2. What can be observed from this table is that the new abstraction tree slightly changes the behaviour of the fixed cluster route strategy in all cases. The actual costs incurred by the simulation of the fixed cluster route strategy are a bit smaller in all cases for the modified clustering. However, this difference is within the standard deviation and this is a demonstration of the stability of the strategy in this type of variation. It has already been proved in proposition 5.5.2 that the clustering order is commutative in the cluster cost assignment. Further, the new distributions of number of visits per room in figure 6.8:ii,iii are very close to those in figures 6.4:iii and 6.5:iii respectively. This indicates that no difference exists in the qualitative behaviour.

In the case of the "uniform" environment, setting the performance of the fixed route strategy on the modified abstraction tree almost matches that of minimax interval. In figure 6.8:i the new distribution of number of visits comes closer to that generated by minimax interval. However, cluster 75 is still visited more

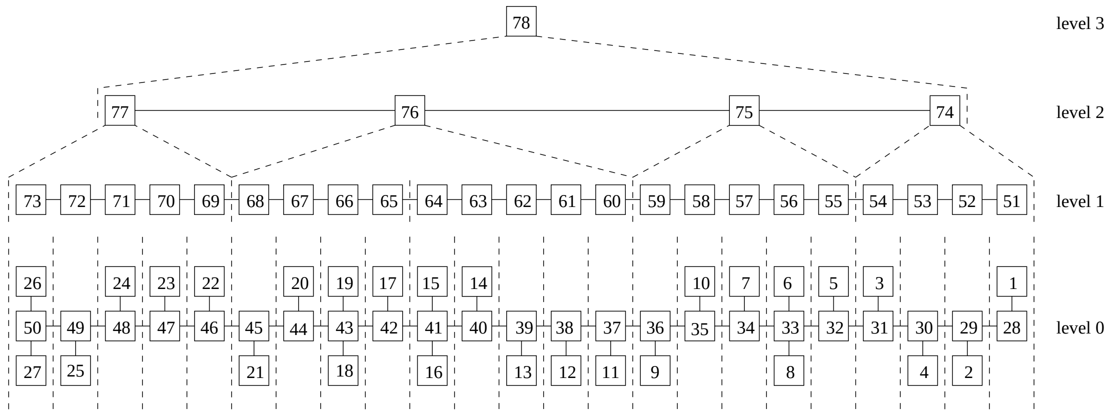


Figure 6.7: The modified block-based abstraction tree.

often and that can be seen from the frequency of visits to the rooms in the *b* grouping. This frequency is comparatively higher and is not desirable, but it is due to artifacts similar to those in example 3.2.12 and in figure 6.6:iii of the look-ahead decisions. This is the reason why the fixed cluster route cannot match the performance of minimax interval in the uniform environment.

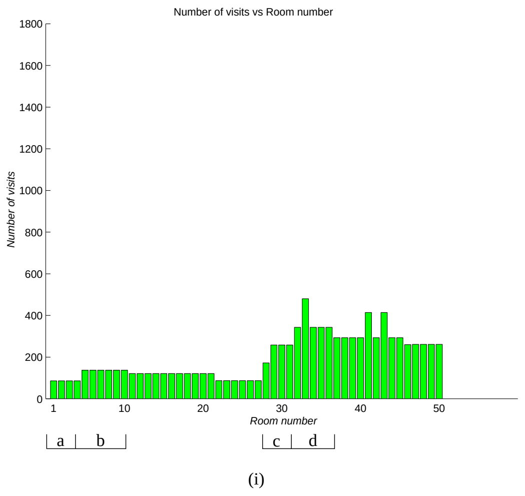
These simulation experiments show that the clustering does not significantly affect the decision procedure. It can perhaps be suggested that clustering areas with similar costs into large lumps is desirable since a small difference in the actual costs is present. However, the improvement from doing this is not very dramatic, especially in the qualitative comparison of the case where cost differences are present. We suspect when cost differences are not present the interaction between the limited look-ahead and the clustering does play a more significant role in the decisions of the robot.

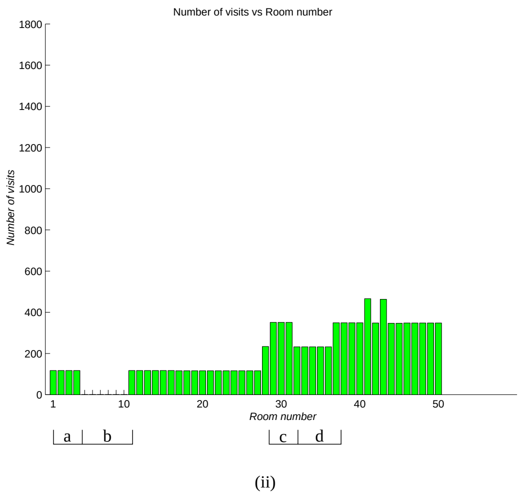
6.4 Summary

The fixed cluster route strategy presents a clear improvement on the naive hierarchical strategy of chapter 4. Especially under the condition of non-uniform cost, the fixed cluster route strategy is a considerable improvement. It behaves in a much more regular fashion and is more open to analysis. This vindicates our decision to cluster the environment based on cluster routes and to assign expected costs to specific routes.

Further, the fixed cluster route strategy shows some robustness in the face of clustering changes. This, again, is true, especially under the condition of a non-uniform cost.

For the uniform case, minimax interval is hard to beat in this environment. It is strongly suspected that the minimax interval is the optimal strategy for that





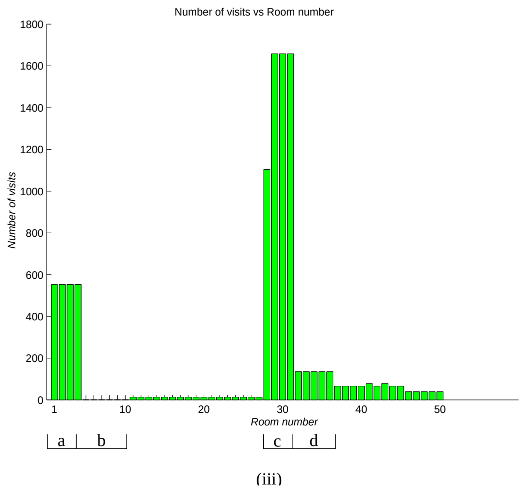


Figure 6.8: Number of room visits vs. room number using the fixed cluster route strategy and the modified abstraction tree. (i) “uniform” (ii) $\hat{c} = 1$ (iii) $\hat{c} = 50$.

case, but even then, the fixed cluster route strategy displays some of the qualities of minimax interval. In the graphs of number of room visits, they have the same local shape, but the fixed cluster route strategy is not following exactly the right distribution of room visits.

Chapter 7

Conclusions

7.1 Overview

This thesis deals with surveillance planning, seen as an optimal search problem. This is a relatively new field and few quantitative results are known. We focused on the problem of fire surveillance in office-like buildings. Although this is a specific instance of surveillance planning, finding a solution to it proves to be exponentially hard computationally.

For this exploratory research effort, various representations and solution methods of a decision-theoretic nature were considered. The problem can be mapped into formalisms, like (PO)MDP, or classical decision theory in many seemingly different ways, which are in fact thought to be equivalent. The formalisation conveys the exponential nature of surveillance planning viewed as an optimal search problem. Consequently, the focus of this thesis is mostly on the computational issues raised by this conceptually simple, yet computationally very hard problem.

The first option for dealing with the computational issues is to limit the look-ahead. This is what is typically done in optimal search problems in order to control the size of the search space. However, if a small look-ahead is used, the results generated are not acceptable because they fall prey to local minima problems, such as the "barrier problem" exposed in chapter 4.

Our solution is to take a step away from the details and to abstract the problem. Search methods based on abstraction boost the effective look-ahead but are necessarily approximate. This creates a hard balancing act between finding a method that is coarse enough to be computable and fine enough to closely approximate the optimal solution. Deciding on this dilemma is not easy, but we have shown that the structure of the problem can be useful. In the surveillance planning problem for an office building, the structure of the topology and that of the cost of the environment largely guide the actions of the robot and this should be reflected in appropriate abstractions. It turns out that for office buildings, there is a sensible general method for grouping locations of similar topological structure in star and corridor shaped clusters.

A new decision strategy for such an abstracted building called the fixed cluster route strategy is proposed. The fixed cluster route strategy computes the cluster expected cost for a predefined route within the cluster instead of giving a heuristic estimate of the cost for all possible routes within the cluster. Just three route types are considered, explore, transit and ignore. The robot then has to commit itself to the predefined route it selected by comparing the expected costs at a fixed decision-level. Although the computation of the route cost is more accurate than before, the method is still heuristic in that not all routes are examined. However, this strategy minimises cost better than the other strategies considered.

7.2 Claims

It is better to base decisions on costs computed for paths rather than on costs computed for clusters.
Independently of how a decision is taken, the robot spends some time following a path. However, if costs are computed per cluster visit, the path and duration of the visit are left unspecified. This implies that computing expected costs for paths is more informed since it takes into account the parameters of the path which in turn affect what the robot encounters. Moreover, simulations suggest that the fixed cluster route strategy minimises expected cost more than the strategy based on cluster costs that was presented in chapter 4.

Progress has been made in analysing the elements of the clustering process for office-like buildings.
Several basic cluster shapes were identified that are useful in clustering office buildings. Further, some ideas were presented on how these basic cluster shapes can be combined to generate a clustering of the environment. The focus was on corridor-shaped buildings. However, generalising to more types of office-like buildings should not be too difficult. The structured approach to clustering, followed in the thesis provides opportunities for automating the clustering process.

Fixed cluster route strategy performs well.
In the simulations performed in an environment with varied costs, our strategy was only beaten by the minimax interval strategy when the environment had uniform cost throughout; in all other cases it beat the other strategies we tried. We believe minimax may actually be optimal for the uniform cost case, the fact that we cannot equate it suggests that our strategy could still be improved in this and the other cases.

Apart from being quantitatively better, fixed cluster routes strategy is also qualitatively better
The fixed cluster routes strategy behaves fairly regularly and is open to analy-

sis by examining the number of room visits or otherwise. Further, this strategy shows robustness in the face of some changes in the clustering.

7.3 Further work

Establish upper and lower bounds on expected cost approximation for clusters
Knowing these bounds is important, since it would permit to trade accuracy for computational complexity. It is believed that the strongest influence on the bounds on expected costs of cluster-route comes from the interactions of routes of different types. So to come up with bounds on the cluster-route expected cost approximations, all issues related to combinations of routes of different types would have to be solved first.

Automation of the clustering procedure
The discussion in this thesis, especially of the simulation results, focuses on a specific instance of an office building. This specific instance has properties that are common in the class of office buildings, but also lacks features such as loops (caused by multiple floors etc). The idea that parts of the environment can be treated in a uniform way is likely to be universal. The decision to limit the attention to paths of equal length which have their cost rather accurately approximated is also recommended in other environments. Coming up with a clustering process for other types of environments, would enable us to develop a general *automated* clustering method. This automated method could be designed based on a combination of formal descriptions and the clustering criteria mentioned in this thesis.

7.4 Final word

To sum up, the focus of this thesis has been the decision-theoretic surveillance planning for office buildings. An analysis of the underlying problem structure has led to solutions that work well for the instance at hand. It is hoped that a contribution has been made to the understanding of the difficulties pertaining to the decision-theoretic approach to surveillance.

Appendix A

Example POMDP application.

In this appendix we use our POMDP setting of the surveillance problem in example 3.2.4. Our first goal here is to give a demonstration of a POMDP application. Our second goal is to show that the result obtained using our POMDP matches that of applying our decision theoretic setting in example 3.2.4. We hope that by achieving this second goal the reader will be, at least partially, convinced that the two settings are equivalent and that hence the same problem is being solved.

A.1 Setting example 3.2.4 as a POMDP

We begin by setting example 3.2.4 as a POMDP. This is essentially the POMDP setting of the surveillance problem (section 3.3.2) with the specific parameters of the example explicitly represented.

States S**.** $\{s_1 = \langle 0,0,X_1\rangle,\ s_2 = \langle 0,1,X_1\rangle,\ s_3 = \langle 1,0,X_1\rangle,\ s_4 = \langle 1,1,X_1\rangle,\ s_5 = \langle 0,0,X_2\rangle,\ s_6 = \langle 0,1,X_2\rangle,\ s_7 = \langle 1,0,X_2\rangle,\ s_8 = \langle 1,1,X_2\rangle\}$

Actions A**.** Here the actions are independent of the state s and are $a_1 = GO(X_1)$ and $a_2 = GO(X_2)$.

Transitions $T : S \times A \rightarrow \Pi(S)$ The following (very symmetrical) transition probability table can be derived using our POMDP setting.

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_2, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_3, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_4, a_1	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
s_5, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_6, a_1	0.1	0.4	0.1	0.4	0.0	0.0	0.0	0.0
s_7, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_8, a_1	0.0	0.0	0.2	0.8	0.0	0.0	0.0	0.0
s_1, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_2, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_3, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_4, a_2	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5
s_5, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_6, a_2	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4
s_7, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8
s_8, a_2	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8

Immediate Rewards $R : S \times A \to \mathbb{R}$. The immediate rewards table calculated as described in the POMDP section is:

	a_1	a_2
s_1	0.5	0.8
s_2	0.5	1.0
s_3	0.5	0.8
s_4	0.5	1.0
s_5	0.5	0.8
s_6	0.5	0.8
s_7	1.0	0.8
s_8	1.0	0.8

Observations Ω. The set of observations Ω is $\{o_1 = \langle f_1 = 0\rangle, o_2 = \langle f_1 = 1\rangle, o_3 = \langle f_2 = 0\rangle, o_4 = \langle f_2 = 1\rangle\}$

Observation function $O : S \times A \to \Pi(\Omega)$. The observation table has the form:

	o_1	o_2	o_3	o_4
$s_1, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_2, a_1 \vee a_2$	1.0	0.0	0.0	0.0
$s_3, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_4, a_1 \vee a_2$	0.0	1.0	0.0	0.0
$s_5, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_6, a_1 \vee a_2$	0.0	0.0	0.0	1.0
$s_7, a_1 \vee a_2$	0.0	0.0	1.0	0.0
$s_8, a_1 \vee a_2$	0.0	0.0	0.0	1.0

Initial State Initially only state $A_0 = \langle 0, 0, X_1 \rangle = s_1$ is considered possible. This is represented by having a belief state b such that $b(s_1) = 1$ and $b(s) = 0$ for all other $s \in S$.

A.2 One step look-ahead POMDP application

Here we apply the POMDP of example 3.2.4 using a one step look-ahead horizon. This creates a sequence of decisions based on the POMDP belief states resulting after each action is taken (see table A.1).

The robot starts with $b(s_1) = 1$ and its first task is to decide which action to take. There are two alternative actions a_1 and a_2. By looking at the rewards table it sees that $R(s_1, a_1) = 0.5$ while $R(s_1, a_2) = 0.8$ so action a_2 is taken first.

The robot takes action a_2. Now suppose observation o_3 is received[1]. Using this information, let us compute the robot's new belief b' over the possible states. To compute the new belief we begin first by computing $P(s'|b, a_2) = \Sigma_{s \in S} T(s, a_2, s') b(s)$. The result of this computation is given in the following table:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b, a_2)$	0.0	0.0	0.0	0.0	0.1	0.4	0.1	0.4

Then given this table, $P(o|b, a_2) = \Sigma_{s' \in S} O(s', a_2, o) P(s'|b, a_2)$ can be computed:

	o_1	o_2	o_3	o_4
$P(o\|b, a_2)$	0.0	0.0	0.2	0.8

And this finally gives $b'(s') = \frac{O(s', a_2, o_3) P(s'|b, a_2)}{P(o_3|b, a_2)}$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b'(s')$	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0

Now a new decision can be taken using the new belief b'. The expected rewards of actions a_1, a_2 are $ER(a_1) = b'(s_5)R(s_5, a_1) + b'(s_7)R(s_7, a_1) = 0.5 \cdot 0.5 + 0.5 \cdot 1.0 = 0.75$ and $ER(a_2) = b'(s_5)R(s_5, a_2) + b'(s_7)R(s_7, a_2) = 0.5 \cdot 0.8 + 0.5 \cdot 0.8 = 0.8$ respectively. So action a_2 is chosen this time.

Further, supposing that observation o_4 is received after a_2 is taken, we can compute the new belief b'' as previously.

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b', a_2)$	0.0	0.0	0.0	0.0	0.05	0.2	0.15	0.6

[1]Since action a_2 was taken the robot gets to location X_2. In location X_2 either observation o_3 or observation o_4 could be received. However, for the purposes of the example only one observation can be used (the robot either finds a fire or not). Fortunately, the observation received has no effects in this example

Then given this table, $P(o|b', a_2)$ can be computed:

	o_1	o_2	o_3	o_4
$P(o\|b', a_2)$	0.0	0.0	0.2	0.8

And this finally gives $b''(s')$:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b''(s')$	0.0	0.0	0.0	0.0	0.0	0.25	0.0	0.75

Again a new decision can be taken using belief b''. The expected rewards of actions a_1, a_2 are $ER(a_1) = b''(s_6)R(s_6, a_1) + b''(s_8)R(s_8, a_1) = 0.25 \cdot 0.5 + 0.75 \cdot 1.0 = 0.875$ and $ER(a_2) = b''(s_6)R(s_6, a_2) + b''(s_8)R(s_8, a_2) = 0.25 \cdot 0.8 + 0.75 \cdot 0.8 = 0.8$ respectively and so action a_1 is chosen this time.

Supposing that the robot receives the observation o_1 the new belief b''' can be computed as:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b'', a_1)$	0.025	0.1	0.175	0.7	0.0	0.0	0.0	0.0

Then given this table, $P(o|b'', a_1)$ can be computed:

	o_1	o_2	o_3	o_4
$P(o\|b'', a_1)$	0.125	0.875	0.0	0.0

And this finally gives $b'''(s')$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b'''(s')$	0.2	0.8	0.0	0.0	0.0	0.0	0.0	0.0

Now a new decision can be taken using the new belief b'''. The expected rewards of actions a_1, a_2 are $ER(a_1) = b'''(s_1)R(s_1, a_1) + b'''(s_2)R(s_2, a_1) = 0.2 \cdot 0.5 + 0.8 \cdot 0.5 = 0.5$ and $ER(a_2) = b'''(s_1)R(s_1, a_2) + b'''(s_2)R(s_2, a_2) = 0.2 \cdot 0.8 + 0.8 \cdot 1.0 = 0.96$ respectively. So action a_2 is now chosen again.

Using this action we want to compute the new belief b''''. Suppose that the robot receives the observation o_4. The resulting tables are:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s'\|b''', a_2)$	0.0	0.0	0.0	0.0	0.02	0.48	0.02	0.48

Then given this table, $P(o|b''', a_2)$ can be computed:

	o_1	o_2	o_3	o_4
$P(o\|b''', a_2)$	0.0	0.0	0.04	0.96

And this finally gives $b''''(s')$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$b''''(s')$	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5

Now a new decision can be taken using the new belief b''''. The expected rewards of actions a_1, a_2 are $ER(a_1) = b''''(s_6)R(s_6, a_1) + b''''(s_8)R(s_8, a_1) = 0.5 \cdot 0.5 + 0.5 \cdot 1.0 = 0.75$ and $ER(a_2) = b''''(s_6)R(s_6, a_2) + b''''(s_8)R(s_8, a_2) = 0.5 \cdot 0.8 + 0.5 \cdot 0.8 = 0.8$ respectively. So action a_2 is now chosen again.

Belief state	$ER(a_1)$	$ER(a_2)$
b	0.5	**0.8**
b'	0.75	**0.8**
b''	**0.875**	0.8
b'''	0.5	**0.96**
b''''	0.75	**0.8**

Table A.1: The expected rewards in the POMDP version of example 3.2.4.

A.3 Remarks related to the example

This example is longer than example 3.2.4. This demonstrates that straight forward computation of the POMDP solution is far from trivial even for the 1-step case. Furthermore, it is worrying that a lot of intermediate tables have to be computed before a decision can be taken by comparing expected rewards.

However, there are some positive results of the comparison between this example and example 3.2.4.

- The sequence in which rooms are explored here is the same as that of example 3.2.4.
- The expected rewards for each action before each decision is taken here (table A.1) are the same as the expected costs in table 3.1.
- The observation of the robot does not affect the decision taken. Try it! Compute the resulting decision for a different observation. A different belief state results due to the different observation, but the actions taken and the expected rewards for this belief state are the same. Still, the observation is needed in the POMDP model to keep track which belief state we are currently in. This, however, shows that there is always some dual belief state (which results in the same decisions) depending on what observation is taken.

In any case, this example should go part of the way towards convincing the reader that our POMDP setting is an accurate description of the problem we have been solving so far.

Bibliography

[BDH96] C. Boutilier, T. Dean, and S. Hanks. Planning under uncertainty: Structural assumptions and computational leverage. In M. Ghallab and A. Milani, editors, *New Directions in AI Planning*, pages 157–172. IOS Press (Amsterdam), 1996.

[BDH99] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 1:1–93, 1999.

[Bel97] J.G Bellingham. New oceanographic uses of autonomous underwater vehicles. *Marine Technology Society Journal*, 31:34–47, 1997.

[BG95] H. Buxton and S. Gong. Visual surveillance in a dynamic and uncertain world. *Artificial Intelligence*, 78:431–459, 1995.

[BK00] M. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *IEEE PAMI Special Section of Video Surveillance and Monitoring*, 22(8):844–851, 2000.

[BK01] D. Beymer and K. Konolige. Tracking people with a mobile platform. In *IJCAI'01 workshop on reasoning with Uncertainty in Robotics (RUR'01)*, pages 57–62, August 2001.

[BW96] J.G. Bellingham and J. Scott Willcox. Optimised AUV oceanographic surveys. In *Autonomous Underwater Vehicles (AUV'96)*, pages 391–398, 1996.

[CB96] R.H. Crites and A.G. Barto. Improving elevator performance using reinforcement learning. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1017–1023. The MIT Press, 1996.

[CD00] R. Cutler and L.S. Davis. Robust real-time periodic motion detection, analysis and applications. *IEEE PAMI Special Section of Video Surveillance and Monitoring*, 22(8):781–796, 2000.

[CLK$^+$00] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2000.

[Fab96] P.J. Fabiani. *Représentation dynamique de l'incertain et stratégie de prise d'information pour un système autonome en enviroment évolutif.* PhD thesis, L'école nationale supérieure de l'aéronautique et de l'espace, November 1996.

[FGBLL01] P. Fabiani, H.H. Gonzalez-Banos, J.C. Latombe, and D. Lin. Tracking a partially predictable target with uncertainties and visibility constraints. *to appear in Journal of Autonomous Robots*, 2001.

[Fri94] J.R. Fricke. Down to the sea in robots. *MIT Technology Review*, October 1994.

[HHD00] I. Haritaoglu, D. Harwood, and L.S. Davis. $W4$: Real time surveillance of people and their activities. *IEEE PAMI Special Section of Video Surveillance and Monitoring*, 22(8):809–830, 2000.

[How60] R.A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, 1960.

[HR98] T. Huang and S. Russell. Object identification: A bayesian analysis with application to traffic surveillance. *Artificial Intelligence*, 103:1–17, 1998.

[HSAHB99] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. Spudd: Stochastic planning using decision diagrams. In *UAI'99*, pages 279–288, 1999.

[IB00] Y.A. Ivanov and A.F. Bobick. Recognition of visual activities and interactions by stohastic parsing. *IEEE PAMI Special Section of Video Surveillance and Monitoring*, 22(8):852–872, 2000.

[KCL$^+$97] T. Kanade, R.T. Collins, A.J. Lipton, P. Anandan, P. Burt, and L. Wixson. Cooperative multi-sensor video surveillance. In *DARPA Image Understanding Workshop (IUW '97)*, New Orleans, LA, May 1997.

[KLC96] L. Pack Kaelbling, M.L. Littman, and A.R. Cassandra. Partially observable markov decision processes for artificial intelligence. In L. Dorst, M. van Lambalgen, and F. Voorbraak, editors, *Reasoning with Uncertainty in Robotics (RUR'95)*, Lecture Notes in Artificial Intelligence 1093, pages 146–163, Berlin, 1996. Springer.

[KML$^+$96] J.H. Kim, B.A. Moran, J.J. Leonard, J.G. Bellingham, and S.T. Tuohy. Experiments in remote monitoring and control of autonomous underwater vehicles. In *Oceans'96 MTS/IEEE*, Ft. Lauderdale, Florida, 1996.

[KMN99] M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI'99*, pages 1324–1331, August 1999.

[Kon96] K. Konolige. A refined method for occupancy grid interpretation. In L. Dorst, M. van Lambalgen, and F. Voorbraak, editors, *Reasoning with Uncertainty in Robotics (RUR'95)*, Lecture Notes in Artificial Intelligence 1093, pages 338–352. Springer, 1996.

[LCK95a] M.L. Littman, A.R Cassandra, and L. Pack Kaelbling. Efficient dynamic-programming updates in partially observable markov decision processes. Technical Report TR-95-19, Computer Science Dept, Brown University, December 1995.

[LCK95b] M.L. Littman, A.R. Cassandra, and L. Pack Kaelbling. Learning policies for partially observable environments: scaling up. In *12th International Conference on Machine Learning*, pages 362–370, 1995.

[Lov91] W.S. Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

[LRS00] L. Lee, R. Romano, and G. Stein. Monitoring activities from multiple video streams: Establishing a common coordinate frame. *IEEE PAMI Special Section of Video Surveillance and Monitoring*, 22(8):758–767, 2000.

[Lug02] G.F. Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley, fourth edition, 2002.

[MBB$^+$97] D.W. Murphy, J.P. Bott, W.D. Bryan, J.L. Coleman, D.W. Cage, H.G. Nguyen, and M.P. Cheatham. MSSMP: No place to hide. In *Association for Unmanned Vehicle Systems International 1997 Conference (AUVSI'97)*, Baltimore, MD, June 1997.

[MDV01] N. Massios, L. Dorst, and F. Voorbraak. A strategy for robot surveillance using the hierarchical structure of the environment. In *IJCAI'01 Workshop on Reasoning with Uncertainty in Robotics*, pages 43–50, Seattle, USA, August 2001.

[MKKC99] N. Meuleau, K.-E. Kim, L. Pack Kaelbling, and A.R Cassandra. Solving pomdps by searching the space of finite policies. In *UAI'99*, pages 417–426, 1999.

[MV98] N. Massios and F. Voorbraak. Planning strategies for decision-theoretic robotic surveillance. In *Tenth Netherlands/Belgium Conference on Artificial Intelligence (NAIC'98)*, pages 117–126, Amsterdam, The Netherlands, November 1998.

[MV99a] N. Massios and F. Voorbraak. Hierarchical decision theoretic planning for autonomous robotic surveillance. In *EUROBOT'99 3rd European Workshop on Advanced Robotics*, pages 219–226, Zurich, Switzerland, September 1999.

[MV99b] N. Massios and F. Voorbraak. Hierarchical decision-theoretic robotic surveillance. In *IJCAI'99 Workshop on Reasoning with Uncertainty in Robot Navigation*, pages 23–33, Stockholm, Sweden, August 1999.

[O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms.* International Series of Monographs on Computer Science. Oxford University Press, 1987.

[ORP00] N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE PAMI Special Section of Video Surveillance and Monitoring*, 22(8):830–843, 2000.

[PROR99] H. Pasula, S. Russell, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In *IJCAI'99*, August 1999.

[PT87] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987.

[Put94] M.L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., 1994.

[RLGIG99] H.R. Everett R.T. Laird, G.A. Gilbreath, R.S Inderieden, and K.J. Grant. Early user appraisal of the mdars interior robot. In *American Nuclear Society, 8th International Topical Meeting on Robotics and Remote Sensing. (ANS'99)*, Pittsburgh, PA, April 1999.

[Son78] E.J. Sondik. The optimal control of partially observably markov processes over a the infinite horizon: discounted costs. *Operations Research*, 26(2):282–304, 1978.

[SS73] R.D. Smallwood and E.J. Sondik. The optimal control of partially observably markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[Sut88] R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Tes92] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–277, 1992.

[Tes94] G. Tesauro. TD-Gammon, A self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[Tes95] G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, March 1995.

[Tes97] C. Tessier. Reconnaissance de scènes dynamiques à partir de données issues de capteurs: le project perception. In *NATO Research and technology organisation meeting on "Multi-sensor and data fusion for telecommunications, remote sensing and radars"*, Lisbon, Portugal, September 1997.

[TKT$^+$00] S. Tadokoro, H. Kitano, T. Takahashi, I. Noda, H. Matsubara, A. Shinjoh, T. Koto, I Takeuchi, H. Takahashi, F. Matsuno, M. Hatayama, J. Nobe, and S. Shimada. The robocup-rescue project: A robotic approach to the disaster mitigation problem. In *IEEE International Conference on Robotics and Automation (ICRA '00)*, April 2000.

[TRS01] G. Theocharous, K. Rohanimanesh, and S.Mahadevan. Learning hierarchical partially observable markov decision process models for robot navigation. In *IEEE International Conference on Robotics and Automation*, pages 511–516, Seoul, Korea, May 2001.

[TT98] R. M. Turner and E. H. Turner. Organization and reorganization of autonomous oceanographic sample networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-98)*, pages 2060–2067, Piscataway, May 16–20 1998. IEEE Computer Society.

[VM98] F. Voorbraak and N. Massios. Decision-theoretic planning for autonomous robotic surveillance. In *ECAI'98 Workshop on Decision* *theory meets artificial intelligence - qualitative and quantitative approaches*, pages 23–32, Brighton, UK, August 1998.

[VM01] F. Voorbraak and N. Massios. Decision-theoretic planning for autonomous robotic surveillance. *Applied Intelligence Journal*, 14(3):253–262, May 2001.

[WM95] D.H Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, The Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA, February 1995.

[WYBM96] J.S. Willcox, Y.Zhang, J.G. Bellingham, and J. Marshall. AUV survey design applied to oceaning deep convection. In *Oceans'96 MTS/IEEE*, Ft. Lauderdale, Florida, 1996.

Index

Samenvatting

Het onderwerp van dit proefschrift is *autonome planning van een surveillancetaak in kantoorachtige omgevingen*. Het surveilleren kan gezien worden als "het nauwkeurig in de gaten houden van iets of iemand met als doel het detecteren van relevante gebeurtenissen". Mensen zijn erg bekwaam in het surveilleren doordat zij feilloos de waarnemingen, acties en besluitvorming weten te integreren. Het automatiseren van deze aspecten om robotsurveillance mogelijk te maken is niet triviaal. In dit proefschrift leggen we de nadruk op besluitvorming betreffende het "waar naartoe"-aspect van het surveilleren.

We benaderen het probleem van *surveillanceplanning* door het te beschouwen als een *probabilistisch besluitvormingsproces*, waarbij we voorlopig het afzonderlijke probleem van het bepalen van waarschijnlijkheden en kosten in realistische situaties buiten beschouwing laten. We zijn uiteindelijk geïnteresseerd in de algoritmische implementatie van zulk een besluitvormingsproces, zodat we de aspecten *formalisering* en efficiënte *berekenbaarheid* in beschouwing moeten nemen.

Om de discussie te vereenvoudigen leggen we de nadruk op één bepaald type relevante gebeurtenissen. De te beschouwen gebeurtenissen zijn probabilistisch, onafhankelijk van elkaar, gelocaliseerd binnen de kamers en resulteren in hoge kosten door schade. We nemen een geïdealiseerde brand als voorbeeld van zo'n gebeurtenis.

Het plannen in een surveillancetaak is een relatief nieuw gebied, en er zijn slechts weinig kwantitatieve resultaten bekend. Voor deze onderzoeksbijdrage werden verscheidene beslissingstheoretische representaties beschouwd. Het probleem kan worden beschreven als een (PO)MDP (*partieel observeerbaar Markov beslissingsproces*) of met klassieke beslissingstheorie op vele schijnbaar verschillende manieren, waarvan we sterk vermoeden dat ze equivalent zijn. De formalisering laat het *exponentiële karakter* zien van surveillanceplanning, gezien als een optimaal zoekprobleem. Daarom zal dit proefschrift de nadruk leggen op de aspecten van berekenbaarheid, die voortkomen uit onze wens.

De eerste optie voor het omgaan met de berekenbaarheid is het beperken van het vooruitkijken tijdens het zoeken. Dit is wat normaliter gebeurt in optimale zoekproblemen, om de grootte van de zoekruimte te beperken. Echter, bij het beperkt vooruitkijken zijn de geboekte resultaten niet acceptabel, daar ze ten prooi vallen aan 'locale minima problemen': wanneer een gebied niet belangrijk genoeg is om te bezoeken, zal het mogelijkerwijs voorkomen dat gebieden verderop ook niet worden geëxploreerd.

Onze oplossing is om niet bij de details te blijven, maar het probleem te abstraheren. Een abstracte representatie van een doelomgeving voor een surveillancetaak kan geconstrueerd worden door gelijkende locaties te groeperen in clusters. De beslissingen worden dan genomen op basis van de diverse manieren waarop de clusters bezocht kunnen worden. Zoekmethoden gebaseerd op abstractie vergroten het effectieve vooruitkijken maar zijn noodzakelijkerwijs een benadering. Dit resulteert in een afweging tussen het vinden van een methode die grof genoeg is om het berekenbaar te houden en fijn genoeg om de optimale oplossing goed genoeg te benaderen. De keuze in dit dilemma is niet makkelijk, maar we laten zien dat de structuur van het probleem nuttig kan zijn. In ons planningprobleem van de surveilleertaak voor een kantoorgebouw bepalen de topologie en het patroon van kosten in de omgeving grotendeels de optimale acties van de robot en dit moet teruggezien worden in geschikte clustervormingen. Het blijkt dat we voor kantoorgebouwen een redelijk algemene methode kunnen geven voor het groeperen van locaties van vergelijkbare topologische structuur in ster- en gangvormige clusters.

We introduceren een nieuwe beslissingsstrategie voor zulke abstracte gebouwen, de strategie van vaste clusterroutes (*fixed cluster route strategy*). Deze strategie berekent de verwachte kosten voor een vooraf gedefinieerde route binnen het cluster, in plaats van het geven van een heuristische schatting van de totale kosten van alle mogelijke routes in het cluster. Drie route types worden onder beschouwing genomen: *exploreer*, *doorkruis* en *negeer*. De robot kiest dan een vooraf gedefinieerde route die geselecteerd wordt door de verwachte kosten op een eindig beslissingsniveau met elkaar te vergelijken.

De strategie van vaste clusterroutes is nog steeds heuristisch, maar simulatie-experimenten laten zien dat het andere nog simpeler strategieën verslaat (ook gepresenteerd in dit proefschrift), wanneer er locale minima aanwezig zijn. Het lijkt erop dat deze strategie verder verbeterd kan worden, omdat het verliest van een simpele één-staps-vooruitkijkminimalisatie van de tijd tussen het bezoeken wanneer er geen kostenstructuur aanwezig is. De hoofdbijdrage van dit proefschrift is waarschijnlijk het theoretische begrijpen van het probleem van surveillanceplanning. De fixed cluster route strategy suggereert dat abstractie de manier is om automatische surveillanceplanning te bereiken.

Abstract

The subject of this thesis is *the investigation of autonomous surveillance planning for an office-like environment.* Surveillance can be informally defined as “a close watch kept over something or someone with the purpose of detecting the occurrence of some relevant events”. Humans perform surveillance tasks quite well, intergrating sensing, action, and decision-making flawlessly. Automation of each of these aspects to enable robotic surveillance is non-trivial. In this thesis, we focus on the decision-making invloved in “where to go next”.

We approach this problem of *surveillance planning* by viewing it as a *probabilistic decision process*, ignoring for now the separate problem of knowing the probabilities and cost in actual situations. We are eventually interested in an algorithmic implementation of such a decision process, so we need to consider aspects of *formalisation* as well as of efficient *computability.*

To simplify the discussion we focus on one type of relevant events. The events considered are probabilistic, independent of each other, localised within office rooms and produce some costly damage when present. We took an idealised version of fire as an example of such an event.

Surveillance planning is a relatively new field and few quantitative results are known. For this exploratory research effort, various representations and solution methods of a decision-theoretic nature are considered. The problem can be mapped into formalisms like (PO)MDP or classical decision theory in many seemingly different ways, which are in fact thought to be equivalent. The formalisation conveys the *exponential nature* of surveillance planning viewed as an optimal search problem. Consequently, this thesis emphasises the computational issues raised by the desire to compute decisions in reasonable time.

The first option for dealing with the computational issues is to limit the look-ahead of the search. This is what is typically done in optimal search problems to control the size of the search space. However, if a small look-ahead is used, the results generated are not acceptable because they fall prey to local minima problems: if a certain area is not important enough to be visited, it may also prevent other areas beyond it from being explored.

Our solution is to move up from the details and to abstract the problem. An abstracted representation of a target environment for surveillance can be constructed by grouping similar locations into clusters. The decisions then are taken among the various ways in which the clusters can be visited. Search methods based on abstraction boost the effective look-ahead but are necessarily approximate. This creates a hard balancing act between finding a method that is coarse enough to be computable and fine enough to closely approximate the optimal solution. Deciding on this dilemma is not easy, but we show that the structure of the problem can be useful. In our surveillance planning problem for an office building, the topology and the pattern of costs of the environment largely guide the actions of the robot and this should be reflected in appropriate clusterings. It turns out that for office buildings, a sensible general method can be presented for grouping locations of similar topological structure into clusters shaped as stars and corridors.

A new decision strategy for such an abstracted building called the *fixed cluster route strategy* is proposed. The fixed cluster route strategy computes the expected cost for a predefined route within a cluster instead of giving a heuristic estimate of the cost for all possible routes within the cluster. Three route types are considered: *explore*, *transit* and *ignore*. The robot then commits itself to the predefined route it selects by comparing the expected costs at a fixed decision-level.

The fixed cluster route strategy is still heuristic, but simulation results show that it beats other simpler strategies, also presented in this thesis, in cases where local minima are present. It is believed that this strategy can be further improved, since it loses from a simple one-step look-ahead minimisation of time between visits when no cost structure is present. The main contribution of this thesis is probably to the theoretical understanding of the surveillance planning problem. The fixed cluster route strategy suggests that abstraction may be the route to achieving automated surveillance planning.