Decision-Theoretic Robotic Surveillance

Massios, N.A.

Citation for published version (APA):
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)MDP's 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.
3.2. Decision-theoretic setting

- **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 $(X, A_0, A, F, C, P_0, P)$, 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 $f_i$.
- $C$ is a function assigning to each location $X_i \in X$ the cost $C(f_i)$ 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 $P_0(f_i)$ 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 $P_0(f_i) = 0$ 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 $(Y_k, Y_{k+1}) \in A$.

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 $t \geq 1$, $r_t \in \{X_i : (r_{t-1}, X_i) \in A\}$. 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 $P_{i+1}(f_i) = P(f_i \rightarrow 1)$ if $r_i = X_i$. 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 $P(f_i \to 0) > 0$ and to model the effect of the actions triggered by observing a fire as an increase in $P(f_i \to 0)$. 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 $f_i \to 1$ is constant over time. It is then possible to express $P_t(f_i)$ in terms of $P(f_i \to 1)$ and the amount of time that has passed since the last visit to $X_i$.

3.2.2. Proposition. Let $E = (X, A_0, A, F, C, P)$ be an environment where $P(f_i \to 0) = 0$, and $r_t = X_i$ implies that $P_{t-1}(f_i) = P(f_i \to 1)$. Then $P_t(f_i) = 1 - (1 - P(f_i \to 1))^{t-t'}$, 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 $P_{t+1}(f_i) = 1 - (1 - P(f_i \to 1))^{t'-t'} = P(f_i \to 1)$. This is correct since $P(f_i \to 1)$ is the probability of a fire starting in one time-step.

Assume for $t = t' + k$. Assume $P_{t+k}(f_i) = 1 - (1 - P(f_i \to 1))^k$.

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 $P_{t+k+1}(f_i = 0) = (1 - P_{t+k}(f_i = 1))(1 - P(f_i \to 1)) = (1 - P(f_i \to 1))k(1 - P(f_i \to 1)) = (1 - P(f_i \to 1))^{k+1}$. 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 $P_{t+k+1}(f_i = 1) = 1 - (1 - P(f_i \to 1))^{k+1}$.

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 $\lambda^p$ is used as confidence decrease rate, where $p$ is the transition probability leaving from the observed state and $\lambda$ 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 ($P(f_i \to 1)$ or $P(f_i \to 0)$) 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 \( P(f \rightarrow 0) \). The transition probability \( P(f \rightarrow 0) \) does not play any role.

Since the factor \( \lambda^{P(f \rightarrow 0)} \) 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 \( \lambda^{P(f \rightarrow 0)} \).

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
\( P(f \rightarrow 0)(t - t') \) 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 \( \min(P(f_i), 1 - P(f_i)) \).

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 \( \sum_{X_i \in X} P(f_i) \log(P(f_i)) \)), give rise to the same preferences as \( \min(P(f_i), 1 - P(f_i)) \). 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 3.2.2, 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.

\[
EC_T = \sum_{1 \leq t \leq T, X_i \neq r_o} P_i(f_i)C(f_i).
\]

Notice that a choice to visit location \( X_i \) at \( t \) not only removes the term \( P_i(f_i)C(f_i) \) from the above sum, but also has some indirect benefits, due to the fact that it reduces \( P_i(f_i) \) 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 \([\text{Fab96}]\) 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 \( E = (X, A_0, A, F, C, P) \), and assume that for all locations \( X_i, X_j \in X \), \( P(f_i \rightarrow 1) = P(f_j \rightarrow 1) \) and \( C(f_i) = C(f_j) \). 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 \( \min(P(f_i), 1 - P(f_i)) = P(f_i) \). If that is the case, choosing the action with maximal uncertainty corresponds to using the 1-step minimum expected cost strategy and we have
3.2. Decision-theoretic setting

$$C(f_1) = 1 \quad \quad C(f_2) = 1$$

$$P(f_1 \rightarrow 1) = 0.5 \quad \quad P(f_2 \rightarrow 1) = 0.8$$

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 $C(f_1) = 1$, $P(f_1 \rightarrow 1) = 0.5$ and $P(f_2 \rightarrow 1) = 0.8$ (Remember that we also assume that $P(f_i \rightarrow 0) = 0$, and that $P_{t+1}(f_i) = P(f_i \rightarrow 1)$, if $X_i \subseteq r_i$). 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 $P(f_2) = P(f_2 \rightarrow 1) = 0.8 > 0.5 = P(f_1 \rightarrow 1) = P(f_1)$. At time 2, $P(f_2)$ is again 0.8, since this room was visited at the immediately preceding time-step. However, $P(f_1)$ has only increased to 0.75, which is not enough to get chosen. Only at time 3, $P(f_1) = 0.875$ 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.
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).

<table>
<thead>
<tr>
<th>time</th>
<th>(EC'(f_1))</th>
<th>(EC'(f_2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>0.75</td>
<td>0.8</td>
</tr>
<tr>
<td>3</td>
<td>0.875</td>
<td>0.8</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>0.96</td>
</tr>
<tr>
<td>5</td>
<td>0.75</td>
<td>0.8</td>
</tr>
<tr>
<td>6</td>
<td>0.875</td>
<td>0.8</td>
</tr>
</tbody>
</table>

In this example, the maximum likelihood strategy does not result in an exhaustive exploration of the environment. Both maximum confidence and minimum expected cost behavior better in this respect if we assume that exhaustive exploration is desirable. The problem with the maximum likelihood criterion is that \(P(f_i) > P(f_j)\) is not guaranteed to result in a preference to visit \(X_i\) rather than \(X_j\) whenever \(P(f_i) > 0.5\). In fact, if \(P(f_i) > P(f_j) > 0.5\), 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 \(P(f_i) > 0.5\). 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 $C(f_i)$ 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 $E = (X, A_0, A, F, C, P)$, and assume that for all locations $X_i, X_j \in X$, $C(f_i) = C(f_j)$. 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 $E = (X, A_0, A, F, C, P)$, with $C(f_i) = C(f_j)$. $P_0(f_i) < 1$, $P(f_i \rightarrow 1) < 1 \ \forall X_i, X_j \in X$. 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 $n < n$.**

**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 $\infty$. Therefore, for $X_k$ the time between visits is $\infty$. 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 $\forall t, P_t(f_i) < p_i$. 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 $\infty$, then $\exists N, \forall X \in X - \{X_k\}, p_i < P(f_k)$. where $P(f_k)$ 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 $\forall X \in X - \{X_k\}, p_i < P(f_k)$, $X_k$ 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 $X$, before $t$, and $T$ be the time of the next visit after $t$. Then the indirect benefits of a visit to $X$ at $t$ are equal to the following:

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

This expression computes for every time-step between $t$ and $T$ two probabilities: (a) the probability of a fire given that room $X$ was last visited at time $t$, and (b) the probability of fire given that room $X$ 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 $X$ 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 $X$, before $t$. Then the indirect benefits of a visit to $X$ at $t$ have the following upper bound.

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

$$= \sum_{n=1}^{t-t'} (1 - P(f_i \rightarrow 1))^{n}.$$
3.2. Decision-theoretic setting

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

\[
\lim_{T \to \infty} \sum_{n=t+1}^{T} (1 - P(f_i \to 1))^{n-t} - (1 - P(f_i \to 1))^{n-t'} = \\
= (1 - P(f_i \to 1)) - (1 - P(f_i \to 1))^{t+1-t'} (1) \\
+ (1 - P(f_i \to 1))^{t+2-t'} - (1 - P(f_i \to 1))^{t+2-t'} (2) \\
+ (1 - P(f_i \to 1))^{t+3-t'} - (1 - P(f_i \to 1))^{t+3-t'} (3) \\
\vdots \\
+ (1 - P(f_i \to 1))^{t+1-t'} - (1 - P(f_i \to 1))^{t+1-2t'} (4) \\
+ (1 - P(f_i \to 1))^{t+2-t'} - (1 - P(f_i \to 1))^{t+2-2t'} (5) \\
+ (1 - P(f_i \to 1))^{t+3-t'} - (1 - P(f_i \to 1))^{t+2-3t'} (6) \\
\vdots \\
\]

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 \( \sum_{n=t+1}^{T} (1 - P(f_i \to 1))^{n-t} \).

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.

![Image of environment](image)

**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 \( C(f_1) = 1 \) and \( C(f_2) = 3 \) (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.
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 \( \lim_{t \to \infty} EC(t_1) = 1 \). 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 \( \sum_{n=0}^{t-1} (1 - 0.5)^n \). 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 \( E = (X, A_0, A, F, C, P) \), 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 \( (X_i, X_j) \in A \) iff \( i \) and \( j \) differ by at most 1. \( C(f_1) = 10 \). \( C(f_2) = 1 \). \( C(f_3) = 3 \). \( P(f_1 \to 1) = 0.1 \). \( P(f_2 \to 1) = 0.5 \) 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.

<table>
<thead>
<tr>
<th>time</th>
<th>( EC(t_1) )</th>
<th>( EC(t_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>2.4</td>
</tr>
<tr>
<td>2</td>
<td>0.75</td>
<td>2.4</td>
</tr>
<tr>
<td>3</td>
<td>0.875</td>
<td>2.4</td>
</tr>
<tr>
<td>4</td>
<td>0.9375</td>
<td>2.4</td>
</tr>
</tbody>
</table>

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).
3.2. Decision-theoretic setting

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

\[
\begin{array}{ccc}
1 & 2 & 3 \\
C(f_1) = 10 & C(f_2) = 1 & C(f_3) = 3 \\
P(f_1 \rightarrow 1) = 0.1 & P(f_2 \rightarrow 1) = 0.5 & P(f_3 \rightarrow 1) = 0.8
\end{array}
\]

Figure 3.3: The environment of example 3.2.10.
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 $E = (X, A_0, A, F, C, P)$, where $X$ is the set \{X_1, X_2, X_3, X_4\}, the accessibility relation $A$ is given by $(X_i, X_j) \in A$ iff $i$ and $j$ differ at most 1, and for all locations $X_i$, $C(f_i) = 1$ and $P(f_i \rightarrow 1) = 0.5$. 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: \( P_t(e) = P(e \rightarrow 1) \). 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 \((S, A, T, R, \Omega, O)\). 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** \(\Omega\). \(\Omega\) 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 \( \Omega \) 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 \( \pi : S \rightarrow A \)
Chapter 3. Problem Formalisation and Complexity

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:

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

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 $b$: $P(s'|a, b, s) = T(s, a, s')$.

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:

$$b'(s') = \frac{P(s'|a, b)}{P(o|s', a, b)P(s'|a, b)}$$

Bayes' rule

$$= \frac{O(s', a, o)P(s'|a, b)}{P(o|a, b)}$$

assumption 1

$$= \frac{O(s', a, o)\sum_{s \in S} P(s'|a, b, s)P(s|a, b)}{P(o|a, b)}$$

total probability rule

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

assumptions 2,3

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:

$$P(o|a, b) = \sum_{s' \in S} O(s', a, o) \sum_{s \in S} T(s, a, s') b(s)$$
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:
  $$\tau(b,a,b') = P(b'|b,a) = \sum_{o \in \Omega} \frac{P(b'|b,a,o)}{P(b|a)} = \sum_{o \in \Omega} P(b'|b,a,o) P(o|b,a)$$
  where:
  $$P(b'|b,a,o) = \begin{cases} 1 & \text{if } SE(b,a,o) = b' \\ 0 & \text{otherwise} \end{cases}$$
- $\rho(b,a)$ is the reward function on the belief states and is defined as:
  $$\rho(b,a) = \sum_{s \in S} b(s) R(s,a)$$

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

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

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

For $s = (f_1, \ldots, f_n, X_i)$ and $a = GO(X_g)$ the probabilities are:
- $P(s' = (f'_1, \ldots, f'_n, X_j)) = 0$ if $j \neq g$
- $P(s' = (f'_1, \ldots, f'_k, \ldots, f'_n, X_g)) = 0$ if $f'_{k} < f_k \& k \neq l$
- $P(s' = (f'_1, \ldots, f'_n, X_g)) = \prod_{f'_g > f_a} P(f_a \rightarrow 1) \prod_{f'_g = 0} (1 - P(f_b \rightarrow 1))$

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 $s' = (f_1, \ldots, f_n, X_l)$ is $U(s') = C(f_i)$ if $f_i = 1$ and $U(s') = 0$ otherwise. This makes the expected reward for the state/action combination:

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

**Observations** $\Omega$. Observations $\Omega$ have the form: $(f_l)$ 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 $s' = (f_1, \ldots, f_n, X_l)$ and action $a = GO(X_l)$ is:

- if $f_k \neq f_l$ then $P(o = f_k) = 0$
- else $P(o = f_l) = 1$

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.
3.3. (PO)MDP settings

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 $(X_1, X_g) \in A$. Here note that $X_1$ 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:

\[ \rho(b, a) = \sum_{s \in S} b(s) R(s, a) = \sum_{s \in S} b(s) \sum_{s' \in S} T(s, a, s') * U(s'). \]

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

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

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 $T : S \times A \rightarrow S$

\[ T(s, a) = T((t_1, \ldots, t_l, \ldots, t_n, X_i), GO(X_g)) \rightarrow (t_1 + 1, \ldots, t_l = 1, \ldots, t_n + 1, X_g) \]

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 $R : S \times A \rightarrow \mathbb{R}$. Since the actions are deterministic, the state action pair $(s, a)$ always results in the same state $s'$. Then the reward is: $R(s, a) = C(f_i) \times P(f_i)$ where $P(f_i) = 1 - (1 - P(f_i \rightarrow 1))^n$ and $X_i$ 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 $S = \langle p_1, \ldots, p_n, X_i \rangle, p_i \in [0,1]^2, X_i \in X$, where $p_i$ is the probability of a fire being present in a room $X_i$, and $X_i$ is the current robot location.

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

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 $T : S \times A \rightarrow S$

$$T(s, a) = T((p_1, \ldots, p_1, \ldots, p_n, X_i), GO(X_g)) \rightarrow$$
$$\langle p'_1 = 1 - (1 - p_1)(1 - P(f_i \rightarrow 1)), \ldots, p'_n = 1 - (1 - p_n)(1 - P(f_n \rightarrow 1)), X_g \rangle$$

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 $R : S \times A \rightarrow \mathbb{R}$. 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 $R(s, a) = C(f_i) \times p_i$, where $X_i$ is the robot location in $s'$.

\[ ^2 \text{note } p_i \text{ is an alternative notation for } P(f_i) \]
3.3. (PO)MDP settings

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
The transition 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:

$$(1 - P(f_i \to 1))^t_i = 1 - p_i \iff t_i \log(1 - P(f_i \to 1)) = \log(1 - p_i)$$

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 $s_1 = \langle t_1, \ldots, t_i, \ldots, t_n, X_i \rangle$. Using the information in $s_1$ the corresponding state $s_2$ of the second MDP can be found:

$s_2 = \langle p_1 = 1 - (1 - P(f_1 \to 1))^t_1, \ldots, p_i = 1 - (1 - P(f_i \to 1))^t_i, \ldots, p_n = 1 - (1 - P(f_n \to 1))^t_n, X_i \rangle$

Then suppose that action $GO(X_g)$ is taken while at state $s_2$, the resulting state $s'_2$ is:

$s'_2 = \langle p'_1 = 1 - (1 - p_1)(1 - P(f_1 \to 1))^t_1, \ldots, p'_i = P(f_i \to 1), \ldots, p'_n = 1 - (1 - p_n)(1 - P(f_n \to 1))^t_n, X_g \rangle$

State $s'_2$ can be rewritten after substitution of values from $s_2$:

$s'_2 = \langle p'_1 = 1 - (1 - P(f_1 \to 1))^{t_1+1}, \ldots, p'_i = P(f_i \to 1), \ldots, p'_n = 1 - (1 - P(f_n \to 1))^{t_n+1}, X_g \rangle$

Note now that $s'_2$ is equivalent (using the inverse mapping mentioned above) to:

$s''_2 = \langle t_1 + 1, \ldots, t'_i = 1, \ldots, t_n + 1, X_g \rangle$

But $s''_2$ 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.
3.3. (PO)MDP settings

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 $S$. $(p_1, p_2, X_l), p_i \in [0, 1], X_l \in \{X_1, X_2\}$.

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 $T : S \times A \rightarrow \Pi(S)$. The transitions are:

$$T(s, a) = T((p_1, p_2, X_l), GO(X_g)) \rightarrow$$

$$\langle p'_l = 0.5, p'_g = 1 - (1 - p_2)(1 - 0.8), X_g \rangle$$

$$T(s, a) = T((p_1, p_2, X_2), GO(X_g)) \rightarrow$$

$$\langle p'_l = 1 - (1 - p_1)(1 - 0.5), p'_g = 0.8, X_g \rangle$$

Immediate Rewards $R : S \times A \rightarrow \mathbb{R}$. 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:

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

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
3.4. Surveillance state space size

\[ s_1 = \langle 0, 0, X_1 \rangle \]

\[ a_1 \quad a_2 \]

\[ s_2 = \langle 0.5, 0.8, X_1 \rangle \quad s_3 = \langle 0.5, 0.8, X_2 \rangle \]

\[ R(s_1, a_1) = 0.5 \quad R(s_1, a_2) = 0.8 \]

\[ a_1 \quad a_2 \]

\[ s_4 = \langle 0.75, 0.8, X_1 \rangle \quad s_5 = \langle 0.75, 0.8, X_2 \rangle \]

\[ R(s_3, a_1) = 0.75 \quad R(s_3, a_2) = 0.8 \]

\[ a_1 \quad a_2 \]

\[ s_6 = \langle 0.875, 0.8, X_1 \rangle \quad s_7 = \langle 0.875, 0.8, X_2 \rangle \]

\[ R(s_5, a_1) = 0.875 \quad R(s_5, a_2) = 0.8 \]

\[ a_1 \quad a_2 \]

\[ s_8 = \langle 0.5, 0.96, X_1 \rangle \quad s_9 = \langle 0.5, 0.96, X_2 \rangle \]

\[ R(s_6, a_1) = 0.5 \quad R(s_6, a_2) = 0.96 \]

\[ a_1 \quad a_2 \]

\[ s_4 = \langle 0.75, 0.8, X_1 \rangle \quad s_5 = \langle 0.75, 0.8, X_2 \rangle \]

\[ R(s_9, a_1) = 0.75 \quad R(s_9, a_2) = 0.8 \]

Figure 3.7: Iterating the second MDP setting.
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 \( (0, 2, 3, \ldots, n - 1, n + 1) \) 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: \( (0, 3, 4, \ldots, n + 1, n + 2) \). 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:

\[
\binom{T}{n-1} = \frac{T!}{(n-1)!(T-(n-1))!}
\]  

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 \( \left( \frac{n-1}{(n-1)!} = 1 \right) \), 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:

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

Applying this to the formula for combinations we get:

$$\binom{T}{m} \approx \frac{\sqrt{2T\pi} \left(\frac{T}{e}\right)^T}{\sqrt{2m\pi} \left(\frac{m}{e}\right)^m \sqrt{2(T-m)\pi} \left(\frac{T-m}{e}\right)^{T-m}}$$

$$= \frac{\sqrt{2T\pi} T^T}{2\pi \sqrt{m(T-m)} m^m (T-m)^{T-m}}$$

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 $T = an = a(n - 1)$. Now we get:

$$\binom{am}{m} \approx \frac{\sqrt{2am\pi} (am)^{am}}{2\pi \sqrt{m(am - m)} m^m (am - m)^{am-m}}$$

$$= \frac{\sqrt{2\pi m(a-1)} m^m (m(a-1))^{m(a-1)}}$$

giving:

$$\binom{am}{m} \approx \frac{\sqrt{a}}{\sqrt{2\pi m(a-1)}} \left(\frac{a}{a-1}\right)^{(a^a - 1)}^m \quad (3.2)$$

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

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

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:

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

The value function is the expected cost resulting from starting in state $s$ and the
following policy $\pi$. 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 $\gamma$.

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

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

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

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

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 try-
ing 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 suc-
cessively longer finite horizons until they converge with some error less than $\epsilon$.
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 al-
gorithm terminates when the maximum change in the value function is below a
threshold.

Policy iteration and value iteration can find optimal policies in time poly-
nomial 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.

```plaintext
for each \( s \in S \) do \( \pi(s) \leftarrow \text{RandomElement}(A) \) end;
repeat
    Compute \( V_\pi(s) \) for all \( s \in S \):
    for each \( s \in S \) do
        Find some \( a \) such that \( |R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_\pi(s')| > V_\pi(s) \):
        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

```plaintext
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
            \( Q_t(s, a) \leftarrow R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_{t-1}(s') \)
        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 $2^{\mid \mathcal{X} \mid}$ states where $\mathcal{X}$ is the set of all possible environment locations (as described in section 3.2.1). For $\mid \mathcal{X} \mid = 32$ 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_{t+1}^f, after action GO(X_3) is taken, does not depend on what the location X_t^f 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_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_t and the presence of a fire f_t at the current location X_t. This means that all literals f_1, ..., f_n have to be used in the influence diagram and so the influence diagram is not significantly smaller than a
3.5. Standard (PO)MDP solving approaches

![Diagram](image)

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

![Diagram](image)

**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
themselfs 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
optimalit y 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 $\epsilon$:

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

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 $\epsilon$ and discount factor $\lambda$. Subsequently, it proceeds to compute an estimate $Q_H^*$ of the optimal state-action value function $Q_H^*$ and select the action with the best $Q_H^*$ 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 $Q_H^*$. In the calculation of $Q_H^*(s,a)$ (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.
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:
    $$Q_h^*(s, a) = R(s, a) + \frac{1}{\gamma} \sum_{s' \in S_a} \text{Estimate } V(h - 1, C, \gamma, G, s')$$
end
return $(Q_h^*(s, a_1), Q_h^*(s, a_2), \ldots, Q_h^*(s, a_k))$:

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

Function: Algorithm $A(\epsilon, \gamma, R_{max}, G, s_0)$
$$\lambda \leftarrow \frac{1 - \gamma^2}{1 + \gamma^2};$$
$$V_{\text{max}} \leftarrow \frac{R_{\text{max}}}{1 - \gamma};$$
$$H \leftarrow \log \frac{\lambda}{V_{\text{max}}};$$
$$\delta \leftarrow \frac{\lambda}{R_{\text{max}}};$$
$$C \leftarrow \frac{1}{\lambda^2} \left(2H \log \frac{kH \sqrt{2 \log \frac{1}{\delta}}}{\lambda^2} + \log \frac{1}{\delta}\right);$$
$(Q_{H}(s, a_1), Q_{H}(s, a_2), \ldots, Q_{H}(s, a_k)) \leftarrow \text{Estimate } Q(H, C, \gamma, G, s_0)$;
return $\max_{a \in \{a_1, \ldots, a_k\}} \{Q_{H}(s, a)\}$:

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.