TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Mikko Lauri

Sequential Decision Making under Uncertainty for Sensor Management in Mobile Robotics

Julkaisu 1366 • Publication 1366

Tampere 2016

Tampereen teknillinen yliopisto. Julkaisu 1366
Tampere University of Technology. Publication 1366

Mikko Lauri

Sequential Decision Making under Uncertainty for Sensor Management in Mobile Robotics

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Festia Building, Auditorium Pieni Sali 1, at Tampere University of Technology, on the 5th of February 2016, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2016

ISBN 978-952-15-3676-2 (printed)
ISBN 978-952-15-3701-1 (PDF)
ISSN 1459-2045

Abstract

Sensor management refers to the control of the degrees of freedom in a sensing system. The objective of sensor management is to improve performance e.g. by obtaining more accurate information or by achieving other operational goals. Sensor management is viewed as a sequential decision making process, where decisions at any time are made conditional on the past decisions and measurement data. At the time of deciding a control action for a sensing system the measurement data that will be obtained are unknown. Thus, informally speaking, a solution to a sensor management problem is a policy that determines which sensing action to undertake given the current information on the state of the process under investigation and contingent on any possible realisation of future measurement data outcomes.

This thesis studies sensor management framing the contingent planning problem in the partially observable Markov decision process (POMDP) framework. In particular, applications in mobile robotics are considered. Mobile robots are viewed as controllable sensor platforms.

Based on earlier work on POMDP based robot control, and distinguishing between the two cases of either exploiting or gathering information, we define four canonical sensor management problem types in mobile robotics. In each of the problem types, we exploit the structural properties of their inputs to improve efficiency of applicable contingent planning algorithms.

In particular, we consider sensor management problems for information gathering where the utility of the possible control policies is quantified by mutual information (MI). We identify the relationship between the POMDP formulation of an environment monitoring problem and another contingent planning problem known as a multi-armed bandit (MAB). In a robotic exploration task, we derive a novel approximation for MI.

Through both simulation and real-world experiments in mobile robotics domains, we determine the applicability, advantages, and disadvantages of a POMDP based approach to sensor management in mobile robotics.

Preface

During the course of this thesis project, I have had the privilege to interact with many inspiring people both at my home department of Automation Science and Engineering at Tampere University of Technology, and elsewhere. Without them the process would surely have been much harder.

I am sincerely grateful to my advisor, professor Risto Ritala. He has supported and guided me throughout the thesis process, and even earlier when I was not yet quite sure if I want to undertake such a project at all. He has been able to provide all the necessary resources for me to successfully pursue my research, and has succeeded in fostering a collegial atmosphere within the research group.

I have received a substantial amount of support from colleagues at the Department of Intelligent Hydraulics and Automation. For this, I want to acknowledge the head of the department, professor Kalevi Huhtala. Associate professor Reza Ghabcheloo has been tremendously helpful, offering his time for discussions and using his contacts to make my progress smoother. I have had many useful discussions also with others, of whom I especially wish to acknowledge Mika Hyvönen and Antti Kolu.

Associate professor Ville Kyrki from Aalto University and assistant professor Matthijs Spaan from Delft University of Technology kindly agreed to pre-examine my thesis. Their careful reading of the thesis along with the comments and suggestions that helped improve the work are much appreciated. I also thank the anonymous reviewers of the journal and conference publications related to this thesis for their valuable feedback.

During discussions with my colleagues they have provided me with food for thought, challenging my initial ideas thus helping to improve them. Their concrete suggestions on experimental setup, along with their expertise on various technological issues has also been very helpful. For this, I especially wish to thank Marja Mettänen, Pekka Kumpulainen, Timo Salpavaara, Olli Suominen, and Joonas Melin. The many discussions on music and life in general with Marja and Pekka have helped make the workplace environment even more enjoyable and inspiring.

The financial support of the TUT President's doctoral programme for funding my research between 2011 and 2015 is gratefully acknowledged. I thank the Finnish Society of Automation for providing travel grants that have allowed me to present my research results at various conferences and workshops.

I am deeply thankful for the support my parents have provided to me throughout my studies. I also thank my brother Juho for his help in

proofreading the thesis. My girlfriend Saija has been there for me throughout the whole process. I could not have hoped for better support from all my friends and family.

Tampere, January 2016,

Mikko Lauri

Contents

List of abbreviations

CTP	Canadian traveller's problem.
DAG	directed acyclic graph.
DBN	dynamic Bayesian network.
E-PCA	exponential family principal component analysis.
EKF	extended Kalman filter.
FIB	fast informed bound.
GMM	Gaussian mixture model.
GPS	global positioning system.
HMM	hidden Markov model.
HSVI	heuristic search value iteration.
KL	Kullback-Leibler.
LQG	linear-quadratic-Gaussian.
LRF	laser range finder.
MAB	multi-armed bandit.
MC	Monte Carlo.
MCTS	Monte Carlo tree search.
MDP	Markov decision process.
MI	mutual information.
MPC	model predictive control.
OLFC	open loop feedback control.
PBVI	point-based value iteration.
PDF	probability density function.
PMF	probability mass function.
POMCP	partially observable Monte Carlo planning.
POMDP	partially observable Markov decision process.
PWLC	piecewise-linear and convex.
RBPF	Rao-Blackwellized particle filter.
RHC	receding horizon control.

ROS	Robot Operating System.
RTBSS	real-time belief space search.
SLAM	simultaneous localization and mapping.
SMC	sequential Monte Carlo.
UCT	upper confidence bounds for trees.
UMDP	unobservable Markov decision process.

Notation

$D(p, q)$	Kullback-Leibler (KL) divergence between two PDFs $p(x)$ and $q(x)$.
$\mathbb{E}_X$	mathematical expectation under $X \sim p(x)$, also $\mathbb{E}$ if probability density function (PDF) or probability mass function (PMF) is clear from context.
$G = (\mathcal{X}, E)$	an (un)directed graph with vertices $\mathcal{X}$ and (un)directed edges E.
$H(X)$	entropy of a random variable X.
$H(X \mid Y)$	conditional entropy of a random variable X given a random variable Y.
$I(X; Y)$	mutual information of random variables X and Y.
$\mathbb{1}_{\mathcal{X}}$	A stochastic identity mapping on the set $\mathcal{X}$. Specifically, for $x_1, x_2 \in \mathcal{X} : \mathbb{1}_{\mathcal{X}}(x_1, x_2) = 1$ if $x_1 = x_2$ and 0 otherwise.
$M(x, d)$	given a graph $G = (V, E)$, the subset of vertices reachable from $x \in V$ by a path of length at most d.
$N(x)$	given a graph $G = (\mathcal{X}, E)$, the set of vertices neighbouring $x \in \mathcal{X}$.
$N(x; m, P)$	a Gaussian PDF over random variable X with mean m and covariance matrix P.
$p(x)$	a PDF or PMF of a random variable X at x.
$p(x \mid y)$	a conditional PDF or PMF of a random variable X at x, conditioned on another random variable $Y = y$.
$\mathcal{P}(\mathcal{X})$	the set of all PDFs or PMFs over the set $\mathcal{X}$.
$2^{\mathcal{X}}$	the power set of set $\mathcal{X}$.
x_t	a variable x indexed by an index t.
$x_{i:j}$	a variable x indexed by an index range from i to j, if $i > j$, interpreted as empty.

X	a random variable.
x	a realisation of a random variable X, or a member of a set.
$\mathcal{X}$	the domain of a random variable X, or a set.
$X \sim p(x)$	a random variable X distributed according to a PDF or PMF $p(x)$.

List of frequently used symbols

a	a decision or action.
$\mathcal{A}$	action space.
b	a belief state, a PDF or PMF over the state.
$\mathcal{B}$	set of beliefs, the set of all PDFs or PMFs over the state space.
d	typically a positive integer denoting search depth in a tree search.
γ	discount factor.
$\mathcal{G}$	generative model of a POMDP, $\mathcal{G} : \mathcal{S} \times \mathcal{B} \times \mathcal{A} \to \mathcal{S} \times \mathcal{Z} \times \mathbb{R}$.
h	a history sequence of actions and observations.
$\mathcal{H}$	set of all possible history sequences.
m	a realization of a map in a robotic exploration problem.
$\mathcal{M}$	the set of possible realizations of a map in a robotic exploration problem.
$\mathbb{O}$	a probabilistic observation model, $\mathbb{O} : \mathcal{Z} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$.
δ	a decision rule, $\delta : \mathcal{B} \to \mathcal{A}$.
π	a policy; a sequence of decision rules.
π^*	an optimal policy.
Π	space of admissible policies.
ρ	a reward function, $\rho : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$.
R	a reward function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.
s	state of a system.
$\mathcal{S}$	state space.
t	an decision epoch or index.
$\mathcal{T}$	set of decision epochs or indices.
τ	belief update equation, $\tau : \mathcal{B} \times \mathcal{A} \times \mathcal{Z} \to \mathcal{B}$.

$\mathbb{T}$	a stochastic state transition model, $\mathbb{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$.
V^{π}	the value function of a policy π, $V^{\pi} : \mathcal{B} \to \mathbb{R}$.
$V^{\pi *}$	the value function of an optimal policy, $V^{\pi *} : \mathcal{B} \to \mathbb{R}$.
Q	a Q-value function, $Q : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$.
J_d	an open loop value function for a look-ahead depth of d decison epochs, $J_d : \mathcal{B} \times \mathcal{A}^d \to \mathbb{R}$.
x	fully observable part of state $s = (x, y)$, or the robot pose in a robotic exploration problem.
$\mathcal{X}$	the fully observable part of a mixed observable state space $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$, or the pose space in an robotic exploration problem.
y	partially observable part of state $s = (x, y)$.
$\mathcal{Y}$	the partially observable part of a mixed observable state space $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$.
z	a data sample or observation.
$\mathcal{Z}$	observation space.

1

Introduction

Sensing systems have become a part of everyday life, and humans today live in an environment with an unprecedented number of sensors. Among other variables, the sensors may monitor the properties of the environment, e.g. the temperature, or the humans themselves, e.g. their heart rate. The sensors are often able to communicate the results of their sensing activities over a network connection for remote access. Likewise, the decreasing price of sensors and increased ability to communicate sensing results has reinvigorated sensing in industrial processes, helping provide information on parts of the process that could not be investigated earlier. Yet another application for deploying sensing systems has emerged with the appearance of mobile robots. Mobile robots are equipped with sensors which enable them to obtain information about areas that are currently not reachable by humans or human-operated vehicles. Examples of such activities include deep sea exploration and study of other planets of our solar system. Modern sensors are often *agile*, meaning that they have multiple possible operating modes, selected by software commands – consider for instance cameras that can be panned and tilted, or a radar whose beam may be steered by controlling the phase of the electrical input signals to its antenna array.

As availability of sensor data has increased, questions on how to manage the collection of the data have become increasingly important. Which temperature sensors should be activated to collect most useful data if we wish to estimate the temperature in a given area? Which operating modes should be chosen for a camera to detect an object of interest? Where should a deep sea mobile sensor platform move to most accurately model the underwater currents and environment? To give justified answers to these types of questions, analysis of the possible decisions and their effects in the context of the sensing system is required. Results of the analysis have the potential to reduce energy consumption, e.g. by reducing the distance a mobile sensor platform needs to travel, and to improve efficiency of executing tasks by reducing the required time, while simultaneously maintaining or even improving the quality of information obtained.

Systems, both human-designed and others, can be characterised using the concepts of state, action, and observation. The state may be thought of as a minimal representation of all internal features relevant to the function of the

system that are required to distinguish different conditions of the system from one another. An action is an external stimulus affecting the system, and an observation is a perceived response from the system conditional on the state. An observation may arise spontaneously without an action affecting the system, but it may also be affected by the action. As a simple example, consider a systems view of making a telephone call to a friend. The caller provides an action by dialling a number on their telephone. Conditional on the state of the line dialled, the caller observes feedback either in the form of a dialling tone or a signal denoting the line as busy. If the line is not busy, the caller may wait for the call to be accepted, which is observed by hearing the receiver of the call answer.

The definition of a system may vary: instead of the states of the two telephones, one could consider the state of the system as the state of the friend, for instance whether they are busy, at home, or at work. One could also consider text messages besides telephone calls as actions. The definition of an observation could then also include the fact whether the friend replied to the text message.

The state of a system is often partially observable; one may only infer it incompletely based on the perceived observations. For example, suppose you call a friend but your call is not answered. You could for instance deduce that they are busy and unable to talk, or that they have not even observed that you tried to call them and hence did not answer.

A decision making task can be defined as providing stimuli to a system in such a manner that the observed feedback implies a desirable state of the system from the point of view of the values and preferences held by the decision maker. For example, suppose again you wish to talk to your friend. From this perspective, it is desirable to dial the friend's telephone and wait for them to answer the call. Note also that a decision making process does not always prompt action: once you have finished talking to your friend you decide, content with the chat you just had, to not take any further action.

Decision making is a central element of almost all human activity, ranging from financial investment and military decisions to social behaviour in a group. The end result of the decision making process outlined above, a transition from an initial state to a desirable final state after the decisions have been acted upon, is also a key feature of automatic control. Consider controlling the flow rate of a liquid in a pipe by means of a valve that may be adjusted by an electrical motor. A simple proportional feedback controller can automatically control the flow when a measurement of the current flow rate is compared to the preferred flow rate, called a set point. The controller then produces a decision – a control signal to the electrical motor – proportional to the difference between the measured value and the set point.

This thesis studies decision making under uncertainty in the context of control of automated systems. The formal methods applied for representing decision making problems are contained within the rubric of decision theory. Of particular interest are decision making strategies applicable for sensor management, i.e. decision making for controlling sensors with user-selectable operation mode or scheduling operation of a network of possibly heterogeneous sensors. The primary application area considered in the thesis

are mobile robots. Out of the general requirements a robot control system should satisfy (Alami et al., 1998), we focus on enhancing *autonomy* by studying methods that enable robots to independently carry out their assigned information gathering tasks.

Technical context. In the following sections, the three key elements of the thesis; uncertainty in decision making, decision theory, and sensor management are discussed in more detail. The subjects are discussed in the specific technical context of *discrete-time Markovian stochastic systems* (Albin, 2003). This entails the following assumptions:

A1: The state of the system under investigation evolves and decisions are executed sequentially, such that the state at the next time instant depends only on the current state and decision, and

A2: mathematical, stochastic system models are available for the state transition and observation processes.

Decision-making throughout the thesis is considered from a *single-agent* perspective, formalizing the problem mathematically as a partially observable Markov decision process (POMDP). However, further discussion on POMDPs is deferred to Chapter 2.

After detailed discussion of the key elements, this chapter is concluded by presenting the research questions studied, describing the contribution of the thesis with respect to the research questions, and providing an outline of the rest of the thesis.

1.1 Uncertainty in decision making

Figure 1.1 illustrates decision making as a continuous process. The decision maker is located within the box on the lower part of the figure, and the system that is being acted upon is located at the top of the figure. The decision maker and the system can be viewed as two agents communicating via actions and observations.

The main parts of the system are separated by a dashed horizontal line. The real process, located above the dashed line, depicts the real system being acted upon, e.g. the telephone in the example of the previous section. In the information process, located below the dashed line, the primary activity is processing of information contained in the perceived observations from the system, and making decisions based on the information. In the case of a human decision maker, this part depicts the thinking process that precedes acting upon decisions. On the right hand side of the figure, the dashed line represents an interface between *decisions* and *acting* on them. On the left hand side, the dashed line represents an interface between the actual state of the system and the perceived observations.

The type of the interfaces depends on the decision making task and also on how the boundaries of the system itself were defined. In the telephone example one could for instance place the decision-action interface either between the telephone dial and the caller's finger, or at the transfer of neural

impulses from the decision maker's brain to the muscles ultimately dialling a number. The former interface is at the physical separation between the system and the decision maker's embodiment, while the latter corresponds to setting the interface at the point where thinking activity first prompts an observable action that affects the system. Similarly, the sensing subsystem acting as a state-observation interface could be contained within or outside of the decision maker's embodiment. In the telephone example, the sensing subsystem is the auditory perception sense of the caller, contained within their embodiment. For the discussion in this section, we consider the sensing subsystem as external to the decision maker, passively interfacing states to observations without the decision maker being able to affect the process (other than indirectly through perceiving the effects of actions on the actual system). We return to the question of controllable sensing subsystems later in Section 1.3.

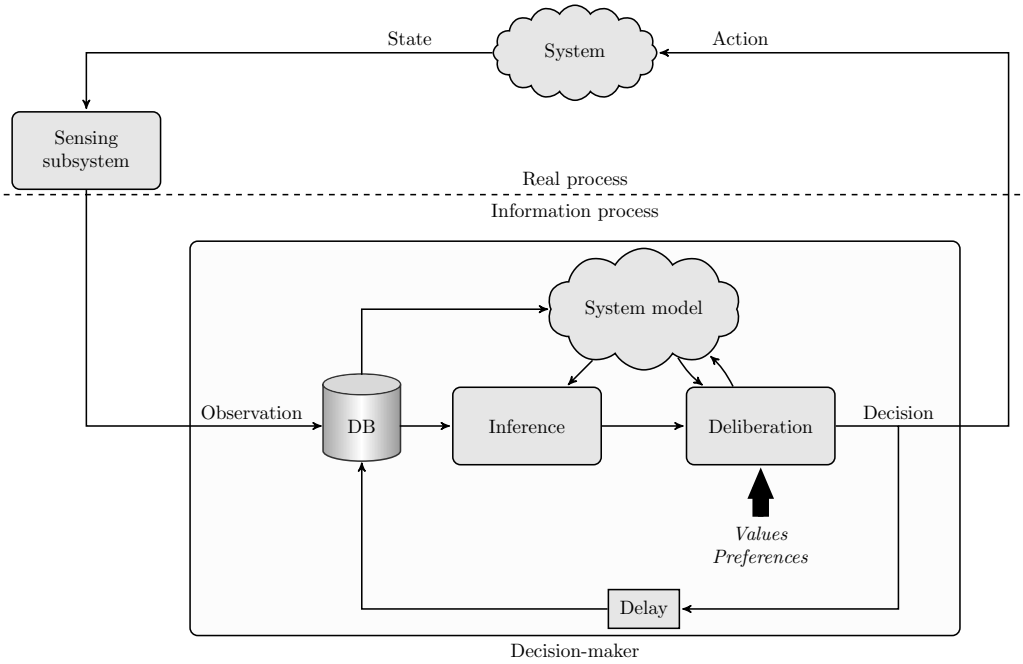


Figure 1.1: Decision making as a continuous process.

Assuming the system being acted upon is fixed, for the remainder of this section we will concentrate on the process of decision making, i.e. the information process in the lower part of Figure 1.1. A decision is made following a three-step process. First, the perceived observation from the system and the previous decision are stored into a database (DB), located on the left hand side of the decision maker box. The database is a representation of the past experiences of the decision maker, a history of decisions and observations. Second, the information currently stored in the database is applied to infer the state of the system being acted upon. Finally, a deliberation that results in a decision is executed, based on results of inference and the decision maker's values and preferences.

The information in the database is interpreted in the context of the system model, as illustrated by the arrow pointing from the system model towards inference. The system model represents, in the case of a human decision maker, his or her understanding of the inner workings of the system. In the case of automatic systems, the model is typically a mathematical object

representing the relationship between states, actions, and observations. The system model is needed since the actual system may be hidden from the decision maker, the only interaction allowed being via the actions and observations. As such, the model is subject to uncertainty: it may well be that the decision maker has an incomplete understanding of the functioning of the system. Thus, it is also desirable in some cases to adapt the system model based on the decision maker's past experiences. This is illustrated by the arrow pointing from the database towards the system model.

Even if the system model were exact, inference is subject to uncertainty if the observations do not allow unambiguous inference of the system state, or conversely, if perception is not perfect. The former case may arise if more than one state can produce the same observation, and the latter case is encountered for instance when observations are being measured by a noisy sensor. Consider the flow rate example presented earlier, where the state of the system is the rate of flow in the tube. A measurement of the flow rate provides an answer to the question, "What is the flow rate in the tube?" However, as measurements are affected by random noise, unknown biases, and other such phenomena, the information about the state provided by a measurement is always incomplete.

According to normative models of decision making, a decision maker acts rationally when their preferences are not circular, i.e. if they prefer outcome a to outcome b and in turn b to c, it should follow that they also prefer a to c (French and Ríos Insua, 2000). Descriptive models of decision making represent actual human decision making processes, and it has been shown that humans are afflicted by a variety of biases and heuristics leading to irrational behaviour and errors in judgement (Tversky and Kahneman, 1974). Under assumption of non-circular preferences, Figure 1.1 is a suitable normative description of decision making in automatic systems. The figure also incorporates the view that the decision maker's values or preferences do not affect their inference.

In the deliberation stage, the decision maker reflects the values and preferences they hold against the effects of decisions as predicted by the system model, given the information about the current state of the system obtained as an output of inference. The process is often iterative, consisting of evaluating various possible future courses of action. This is illustrated in Figure 1.1 by the two-way arrows between the deliberation block and the system model. The effects of decisions cannot always be predicted with complete precision. This is the case especially in stochastic systems, that may respond to the same action in different ways depending on random chance. Stochasticity may arise for example when there are phenomena whose effect on the state of a system is unknown but significant, or when the system model is simplified such that the state does not account for all aspects of how the system functions and thus cannot fully predict its response to an action.

We conclude that uncertainty in decision making manifests itself as incomplete knowledge of the current state, incomplete answers to attempts to gain information about the state, or lack of certainty about the effect of a decision. As discussed, uncertainty may arise because of modelling errors or other reasons. Generally, a decision making task can feature uncertainty

from any subset of the three manifestations.

Depending on the decision making task, some or all of these sources of uncertainty can sometimes be ignored. If uncertainty cannot be ignored, principled methods of coping with it are required.

1.2 Decision theory and utility

Decision theory refers to

> "...the class of statistical problems in which the statistician must gain information about certain critical parameters in order to be able to make effective decisions in situations where the consequences of his decisions will depend on the values of these parameters." (DeGroot, 2004)

Considering the discussion of decision making above, the parameter on which the consequences of decisions depend is the state of a system. Information about the parameters refers to inference on the contents of the database, and helps the decision maker to act in a manner that best agrees with their values and preferences.

Assuming that the information about the system state is incomplete, a probability value depicting the relative likelihoods that the system resides in a state may be assigned to each possible state. This assignment effectively provides a mathematical quantification, in the form of a probability density function (PDF) over the space of possible states of the system, of the *uncertainty* the decision maker is currently experiencing about the state; the value of the above mentioned critical parameter upon which the consequences of their decisions depend. A complementary but fundamentally equivalent way to understand the PDF is as representation of the information currently available to the decision maker about the state of the system. The PDF is often called a belief state. Other methods for representing uncertainty include e.g. fuzzy sets, but they are not considered further in this thesis.

When decisions are implemented as actions and consequential observations are perceived, a mathematical engine for revising existing information is required. When information is represented as a PDF and a mathematical system models for the state and observation process are available, Bayesian filtering (Särkkä, 2013) is applied for revising it. Bayesian filtering consists of a prediction and an update step. In the prediction step, a mathematical model of the system state conditional on the previous state and action is applied to propagate current information into a prediction of the resulting state. This model is often called the dynamical model of the system. Once an observation is perceived, another mathematical model of the observation conditional on the current state and previous action is applied to incorporate information provided by the observation into the state information. This model of the perception process is often called the observation model of the decision maker. From the point of view of the decision maker, these two models, along with any possible prior information, are sufficient to infer a PDF on the current system state based on the history of past actions and observations.

When the values and preferences of the decision maker are likewise quantified, e.g. as a function that assigns a numerical value for each pair of state and action, the desirability of various courses of action may be compared in an objective[1] manner. This forms the basis of statistical decision theory; see e.g. Raiffa and Schlaifer (2000); DeGroot (2004).

The application of statistical decision theory typically results in finding a policy, a prescription of which course of action is to be taken given e.g. the current belief state. An *optimal* policy is the one which, when followed, will produce an outcome that is the best achievable given the notion of optimality applied in the analysis. One commonly applied definition of optimality is given by the theory of expected utility of Von Neumann and Morgenstern (1953), who showed that if there exists a complete ordering of the preferences of possible outcomes, then there exists a utility function that quantifies the desirability of states and actions, and that the decision maker prefers actions that maximise the expected value of the utility function. The expected value of the utility function is calculated under the aforementioned probability density function quantifying the likelihood of each state. Other possible notions of optimality include e.g. maximising the probability of the system reaching some desirable state, or minimising the probability of an adverse state. For the remainder of the thesis, we adopt maximising expected utility as the notion of optimality.

Often, decisions have both short-term and long-term consequences. When a single decision is considered, an optimal decision may be found by maximising the expected value of the utility. However, focusing on optimality over a single decision may not be sufficient. For instance, a cross-country skier in a long race considering performance only along the next hundred meters will likely not fare well. Instead, both the present and future decisions and their possible outcomes should be considered at the time of decision making. The time frame over which future outcomes and decisions are considered thus also affects the outcome of finding an optimal decision.

The problem of maximising expected utility over multiple decisions has been central in the field of dynamic programming (Bellman, 1957). Bellman's principle of optimality, which states that for an optimal course of action, "whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision", is leveraged to break the multi-stage decision problem into simpler sub-problems. A backward recursive procedure is applied by which the maximal expected utility along with the decision that leads to it may be recovered for any number of remaining decisions after the current one. Not every optimal decision problem has the property that allows this to be done – a utility value must be assignable to each individual decision for dynamic programming to be applicable. In our context of Markovian systems, this requirement translates to a utility value being assigned to each pair of state and decision, with the additional requirement that the decision-maker instantaneously receives this reward as a result of the decision.

The complexity of the problem increases as the number of possible states in the system increases, an issue termed the curse of dimensionality. Reasoning

[1] Provided the decision maker's values and preferences themselves are not reflective of any biases.

over long time horizons constitutes a major source of computational difficulty in both dynamic programming and mathematical decision theory in general.

Statistically optimal decisions are analysed with the aforementioned mathematical models of the system, i.e. the models are applied within the deliberation loop in Figure 1.1. Thus when we discuss optimal decisions, we refer to optimality *with respect to the mathematical model* of the system applied unless otherwise specified. As explained e.g. by Kaelbling et al. (1998), it may seem counter-intuitive that the decision maker achieves a large expected utility merely by *believing* it is in a good state, i.e. attaining a belief state where it is in a preferred state with high probability. However, if the system models are correct, there is no discrepancy between the expected rewards under the decision maker's belief state and the true expected rewards considering the real system. When models cannot be assumed correct, special care must be exercised when results obtained from a decision-theoretic optimisation are applied; especially when there is a risk of great negative effects from bad decisions.

1.3 Sensor management

Sensor management refers to "... control of the degrees of freedom in an agile sensor system to satisfy operational constraints and achieve operational objectives" (Hero and Cochran, 2011). As such, sensor management in automatic systems is intimately related to decision making. The system under study is a sensor itself, or a sensor subsystem of a larger system having also other functions. The operational objectives refer to the values and preferences of the decision maker, expressed e.g. as a utility function.

Resource constraints often prevent simultaneous use and processing of all available measurement data. Although more data processing resources are available today than ever before, the demand for sensor management has simultaneously increased due to technological developments in both sensor technology and communication networks.

As software has become an increasingly important part of modern sensor technology, the agility of sensors has increased dramatically. Multiple degrees of freedom are available for operation of sensors. By allowing rapid changes to the configuration of the sensor via software commands, the traditionally monolithic sensor has been transformed into an array of virtual sensors, each corresponding to a single configuration. Examples include cameras whose focus can be switched to a certain part of the visible scene and radars with variable signal waveform or transmission frequencies.

Another advance in sensor technology over the last decades has been the increasing availability and decreasing cost of increasingly more capable sensors. For instance, a modern mobile phone for consumer use typically has sensors for imaging, detecting tactile inputs, and measuring the position and inertial state of the device.

When sensors are carried by a mobile platform, even more degrees of freedom are introduced to the operation of the sensor subsystem. Mobile sensor platforms that are remotely operated or working autonomously have allowed measurement campaigns in environments where human presence is either

impossible or undesired. Examples include deep ocean measurements, exploration of other planets of the solar system, and other long-lasting data collection missions in remote or hazardous locations.

Advances in communication networks and infrastructure mean that also the availability of remote access to the data produced by the sensors has improved considerably. With the present emphasis on increasing connectivity of individual devices through concepts such as the Internet of Things, the trend of increasing availability of sensor data is likely to continue.

The increasing number of agile sensors may produce data in quantities that are not possible to analyse fully, given constraints on available computing power or time required by the analyses. When multiple sensors are available, it may sometimes be impossible to operate all of them concurrently, or operation may be restricted to certain subsets of sensors at a time. Possible reasons for such restrictions include conflicts in use of the electromagnetic spectrum (e.g. in case of radars), or limited bandwidth to communicate the data to the user. Likewise, it is desirable to operate the sensor in such a way that the operational objectives are most effectively reached; minimising e.g. the time or resources spent. All these factors have contributed to an increasing need for sensor management in automatic systems.

As this thesis considers decision making for controlling sensors, the aspect mentioned in the quotation in the beginning of Section 1.2 about experiments providing new information about the state is particularly interesting. In the context of automated systems, new information typically arrives in the form of measurement data from sensors. From the point of view of sensor management, the objective can be either operating a sensor system to gather the greatest possible amount of information about the system state, or to support execution of a task that is not necessarily related to sensing per se. This has important implications for the utility function of the decision making problem. If the preference of the decision maker is e.g. to reach a certain state in the system, it is sufficient to formulate the utility function as dependent on the state only. Then, an optimal course of action is to gather information only up to the extent that the information best helps to achieve the aforementioned preferred state. However, when information gathering *itself* is an objective, it is in general not possible to formulate the utility function as purely state-dependent: which state the system resides in is often independent of the decision maker's knowledge. Instead, the utility function is defined directly in terms of the belief state: a small utility is assigned to uncertain belief states, and a high reward to certain belief states where probability mass is concentrated on few underlying states.

Figure 1.2 provides a schematic view of the decision making process in a sensor management problem. The figure may be seen as equivalent to the general view of Figure 1.1, however, we have chosen to show this alternative representation to facilitate discussion on aspects specific to sensor management problems. We again view the sensing subsystem as external to the decision maker, although the resulting analysis is fundamentally the same even for an embodied sensing subsystem.

The sensing subsystem shown on the left hand side of Figure 1.2 consists of n sensors labelled by consecutive integers from 1 to n. Depending on the case these sensors may either refer to multiple distinct sensors, or represent the

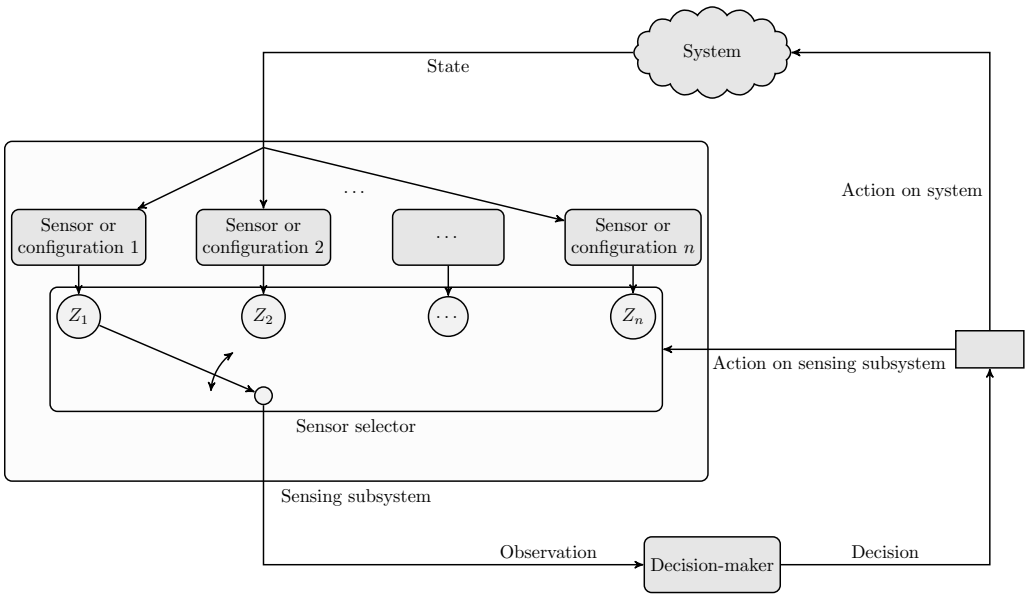


Figure 1.2: Sensor-management as a decision making problem.

multiple configurations that a single sensor may be operated in. Each sensor $1 \leqslant i \leqslant n$ provides an observation Z_i conditional on the state of the actual system. A sensor selector, presented on the bottom of the sensing subsystem, selects which sensor's observation will be recorded by the decision maker. The sensor selector may either select a single sensor or a subset of them at a time, by connecting them with the system on the top of the figure. The possible configurations the sensor selector can choose are such that the physical and other constraints outlined in the paragraphs above are satisfied.

The sensor selector can be acted upon by the decision maker to choose which sensor's observation to perceive next. In the general case, the decision maker is jointly applying control actions on the system itself, and actions on the sensing subsystem to control the observation process. In case the decision maker has no control over the evolution of the system state and is merely an observer, the arrow connecting the decision maker to the system is removed. If the system state cannot be acted upon, the utility function must depend on the decision maker's belief state: if it did not, no action on the sensing subsystem could affect the state of the actual system and hence the reward, rendering the resulting optimal decision problem meaningless.

1.4 Research questions

This thesis investigates the application of model-based methods for decision making under uncertainty, applied to sensor management. In particular, the subset of Markovian systems with hidden state, also called partially observable Markov decision processes (POMDPs) is studied. The two cases where utility is measured by either a state or belief state dependent function are considered. Mobile robots are considered as the primary application area.

The following research questions (RQ) are studied in this thesis.

RQ 1: How should sensor management problems be formulated as POMDPs?

RQ 2: Which structural properties of sensor management problems can be taken advantage of to tailor existing exact and approximate algorithms for solving POMDPs to be efficient in solving sensor management problems in mobile robotics?

RQ 3: For the approximate methods considered, what guarantees on the quality of the solution can be provided compared to the optimal solution?

1.5 Contribution

The objective of the thesis is to discuss alternative formulation of sensor management problems in mobile robotics as POMDPs, to identify structural properties of sensor management problems that can be exploited in finding a useful solution more effectively, and to demonstrate the applicability of POMDP-based sensor management in the domain of mobile robotics. In view of these objectives, the contributions of the thesis are as follows.

- A comparison of state-of-the-art algorithms for solving POMDPs applied to sensor management problems is provided.
- Based on a review of existing work on applying POMDPs to control of real robots, a set of canonical sensor management problems relevant in mobile robotics is defined, and suitable algorithms for solving the problems are identified. The applicability is empirically verified via simulation.
- For each of the canonical problems, structural properties that can be exploited in solving the problem are identified. In particular,
  - For problems with a state dependent utility function, bounds and heuristic approximations for the optimal value function are provided.
  - For problems with a utility function that is *non-linear* in the belief state, connections to multi-armed bandit problems are identified.
- Applicability of a POMDP-based approach to sensor management in autonomous robotic exploration of an unknown environment is demonstrated.

1.6 Outline of the thesis

The thesis is organised as follows. Chapter 2 reviews the state-of-the-art of partially observable Markov decision processes and multi-armed bandits. Chapter 3 formulates four canonical sensor management problems in mobile robotics domains for this study. Necessary background information on graphs and graphical models is also provided. Chapter 4 gives the main contribution

of the thesis. Features of the canonical problems and suitable POMDP solution techniques are analysed. Exploiting structural features of each canonical problem type is described. Chapter 5 validates the analysis of the previous chapters and provides empirical comparison between suitable solvers for the canonical problems via simulation experiments. A sensor management system applied to autonomous robotic exploration is described, and the system is validated in a real world exploration task. Finally, Chapter 6 provides discussion of the results and concludes the thesis.

2

Markovian systems and decision processes

As discussed in the Chapter 1, sensor management may be seen as a decision-making process. Markov decision processes (MDPs) are a subset of general stochastic decision processes, with the distinction that the effects of decisions depend only on the decision-maker's action and the current state of the system. This property, known as the Markov property, may be interpreted as a lack of memory in a stochastic process.

The Markov property is a useful assumption in modelling sequential data. Sequential data models are applied in a wide range of applications, ranging from text or speech recognition (Bishop, 2006), modelling of biological signals such as sequences of nucleic acids (see e.g. Eddy, 1996) to tracking targets by radar (Moran et al., 2008).

The Markov property is often justified from a practical point of view and many stochastic processes that do not satisfy the Markov property can be well approximated assuming they do (Albin, 2003). Systems with a finite memory longer than a single time step may be transformed into Markovian systems with one-step memory by state augmentation. In decision processes, the Markov property together with a suitable objective function allows solutions based on dynamic programming methods (Bellman, 1957).

The objective of this chapter is twofold. The first objective is to provide background information on the mathematical formulation of decision-making processes that can model sensor management problems. This facilitates further discussion on problem formulation in subsequent chapters. The second objective is to identify which optimal or approximate algorithms for finding solutions to Markovian decision processes are most relevant for sensor management problems.

Markov decision processes are defined in Section 2.1. Both fully observable states and partially observable states are considered. The latter is especially important for sensor management applications. Section 2.2 reviews the state-of-the-art in solving Markov decision processes with partially observable state. The chapter is concluded in Section 2.3 by giving the definition of a related decision process model called a multi-armed bandit (MAB).

2.1 Sequential decision processes

In a stochastic decision process, an agent sequentially applies actions on the process and perceives their effects. The process is called stochastic, as the action effects are not known exactly, but rather a PDF over the possible effects is available. In this section, we consider discrete-time Markovian decision processes. The motivation for this choice is principally practical; tractable solutions are available for discrete-time decision processes corresponding to realistic problems. For an overview of continuous-time stochastic decision processes we refer the reader to the classic book of Åström (1970) on stochastic control theory.

We give definitions for discrete-time Markov decision processes with both fully observable and partially observable state. Partial observability means that the state of the process is not directly observable, but rather information about the state is obtained via noisy or incomplete observations. Partial observability is an especially useful feature for modelling sensor management problems.

Subsections 2.1.1 and 2.1.2 define the sequential decision making framework with full and partial observability, respectively. Discussion on how to act optimally in the context of this framework is deferred to Subsection 2.1.3, including formalisation of decision rules and policies. The basic recipe for finding optimal policies via dynamic programming is presented in Subsection 2.1.4. Finally, Subsection 2.1.5 presents a brief taxonomy of Markovian decision processes and their relation to each other, including cases with multiple decision-makers.

Notation. Throughout the text, we denote random variables by uppercase italic letters, e.g. X. The space of possible realisations of a random variable X is denoted by a calligraphic letter, e.g. $\mathcal{X}$. A similar calligraphic notation is adopted for sets. Members of sets and realisations of random variables are denoted by lowercase italic letters, e.g. $x \in \mathcal{X}$. Sequences are denoted as $(x_k)_{i \leqslant k \leqslant j} \equiv x_{i:j}$. If a continuous random variable X is distributed according to a given PDF $p : \mathcal{X} \to \mathbb{R}^+$, we write $X \sim p(x)$. For brevity, we occasionally denote different PDFs by the same function p – in this case, the arguments of the function distinct which PDF is referred to by each expression. The space of all possible PDFs over $\mathcal{X}$ is denoted by $\mathcal{P}(\mathcal{X})$. For discrete random variables and their probability mass functions (PMFs) over finite sets, we adopt a similar notation.

2.1.1 Markov decision processes

Throughout this section, we assume that the realisations of all random variables can be completely observed. We define a stochastic process which satisfies the Markov condition. After defining a controllable Markov process, we ultimately define a Markov decision process.

Definition 2.1 (Stochastic process)**.** *A stochastic process is a collection of random variables $\{S_t\}$ indexed by time $t \in \mathcal{T}$, where $\mathcal{T}$ is an index set.*

The index set $\mathcal{T}$ typically models time and can be either discrete or continuous, e.g. $\mathcal{T} = \{0, 1, \ldots\}$ or $\mathcal{T} = \{t \in \mathbb{R} \mid 0 \leqslant t < \infty\}$, respectively, or some closed interval subset of either.

Consider a discrete-time stochastic process $\{S_t\}$ with $\mathcal{T} = \{0, 1, \ldots\}$. Each S_t models the system state at time t and assumes values in a state space $\mathcal{S}$. In a general causal stochastic process, S_{t+1} may depend on the realisations of any S_k, $k \leqslant t$. This leads to a state transition model defined as a conditional PDF $p(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_0)$. Note that we assume the state transition model to be independent of t. This assumption is made to simplify notation, and no conceptual difficulties arise from defining state transition models dependent on t, as is done e.g. by Puterman (1994).

When dependency on past states is reduced to S_t only, a Markov process is defined.

Definition 2.2 (Markov process)**.** *A discrete-time stochastic process $\{S_t\}$ with $\mathcal{T} = \{0, 1, \ldots\}$ is a Markov process if its state transition model satisfies the Markov condition*

$$p(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_0) = p(s_{t+1} \mid s_t) \tag{2.1}$$

for all $t \in \mathcal{T}$ and all $s_0, s_1, \ldots, s_{t+1} \in \mathcal{S}$.

In a decision process, an agent has the opportunity of influencing a stochastic process by applying actions. Fundamentally this means that the state transition model of the process is an action-dependent function. Actions can be selected at the *decision epochs* determined by an index set $\mathcal{T}$. If $\mathcal{T} = \{0, 1, \ldots\}$, the decision process is of infinite horizon. If $\mathcal{T} = \{0, 1, \ldots, d\}$, $d \in \mathbb{N}$, the decision process is of finite horizon, and the last decision is made at epoch $(d-1)$. If at some decision epoch the system is in state $s \in \mathcal{S}$, the agent may choose the action to apply from the set of actions allowed in s, denoted $\mathcal{A}_s$. Let $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}_s$ denote the action space of the decision process.

With the introduction of actions, a controlled stochastic process is defined, where in general the next state depends on the past state and actions both via a state transition model $p(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_0, a_t, a_{t-1}, \ldots, a_0)$. In a controlled Markov process, state transitions satisfy the Markov condition with respect to the states and actions.

Definition 2.3 (Controlled Markov process)**.** *In a controlled Markov process, the state transition model satisfies*

$$p(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_0, a_t, a_{t-1}, \ldots, a_0) = p(s_{t+1} \mid s_t, a_t) \tag{2.2}$$

for all $t \in \mathcal{T}$ and all $s_0, s_1, \ldots, s_{t+1} \in \mathcal{S}$, $a_0, a_1, \ldots, a_t \in \mathcal{A}$.

The state transition model in a controlled Markov process is concisely represented as a function $\mathbb{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$, such that $\mathbb{T}(s', a, s)$ is the value of the PDF over the new system state s' when the system is currently in state s and action a is executed. A valid state transition model must satisfy $\int_{\mathcal{S}} \mathbb{T}(s', a, s) \mathrm{d}s' = 1$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$[1].

[1] For finite $\mathcal{S}$, $\mathbb{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ and the integration is replaced with a sum.

The actions are applied in a sequential manner. At decision epoch t, the system is in a state s_t. The agent then executes action a_t, and the system transitions to a new state s_{t+1} according to $\mathbb{T}$. The agent receives a reward $R'(s_t, a_t, s_{t+1})$, which is a random quantity as it depends on the system state at decision epoch $(t+1)$. The reward function is assumed to be independent of the decision epoch, although no extra difficulty beyond notational inconvenience arises from epoch-dependent rewards. Positive reward is interpreted as an income, and negative reward as a cost. We adopt the alternative view of the reward function R' where it is replaced by its expected value calculated by

$$R(s_t, a_t) = \mathbb{E}_{S_{t+1}}\left[R(s_t, a_t, s_{t+1})\right], \tag{2.3}$$

defining a new expected reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The expectation in the expression above is taken with respect to $\mathbb{T} \equiv p(s_{t+1} \mid s_t, a_t)$. In a finite-horizon decision process with $\mathcal{T} = \{0, 1, \ldots, d\}$, the last action is selected at decision epoch $(d-1)$, and an additional real-valued terminal reward $R_d(s_d)$ is sometimes defined. Throughout the rest of the thesis we assume the terminal reward is equal to zero.

Adding the reward process to a controlled Markov process defines a Markov decision process (MDP).

Definition 2.4 (Markov decision process; Puterman, 1994)**.** *A Markov decision process is a tuple $\langle \mathcal{T}, \mathcal{S}, \{\mathcal{A}_s\}, \mathbb{T}, R\rangle$, where $\mathcal{T}$ is the set of decision epochs, $\mathcal{S}$ is the state space, $\mathcal{A}_s$ is the set of actions allowed in $s \in \mathcal{S}$ such that $\mathcal{A} = \bigcup_{s\in\mathcal{S}} \mathcal{A}_s$, $\mathbb{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the state transition model, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the real-valued reward function.*

2.1.2 Partially observable Markov decision processes

We now consider the case where the state of the system $\mathcal{S}$ is not directly observable by the agent. After a state transition the agent now perceives an observation $z_{t+1} \in \mathcal{Z}$ instead of learning the value of the state $s_{t+1} \in \mathcal{S}$. We assume that the observations are conditionally independent given the current state and previous action, i.e. $Z_{t+1} \sim p(z_{t+1} \mid s_{t+1}, a_t)$. This probabilistic observation model is assumed independent of the decision epoch. The observation model is represented as a PDF $\mathbb{O} : \mathcal{Z} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$, such that $\mathbb{O}(z', s', a)$ is the value of the PDF for observation z' when the system is in state s' after action a was executed at the previous decision epoch. A valid observation model must satisfy $\int_{\mathcal{Z}} \mathbb{O}(z', s', a)\mathrm{d}z' = 1$ for all $s' \in \mathcal{S}$ and $a \in \mathcal{A}$[2].

As the state is not directly observed, the agent's knowledge about the state at the start of the process is modelled by a PDF $p(s_0) \in \mathcal{P}(\mathcal{S})$. If no prior information exists, $p(s_0)$ is defined to be an uniform distribution over $\mathcal{S}$. Knowledge that the initial state is s_0 can be modelled by setting $p(s_0)$ to a degenerate distribution at s_0.

At decision epoch t, the prior $p(s_0)$ and past actions and observations contain all information that the agent has available about the current and past states of the system. Let $h_0 = p(s_0) \in \mathcal{H}_0$, and for $t \geqslant 1$, let

[2]For finite $\mathcal{Z}$, $\mathbb{O} : \mathcal{Z} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ and the integration is replaced with a sum.

$h_t = (p(s_0), a_0, z_1, a_1, z_2, \ldots, a_{t-1}, z_t) \in \mathcal{H}_t$ denote the history at decision epoch t. We have $\mathcal{H}_0 = \mathcal{P}(\mathcal{S})$, and for any $t \geqslant 1$ the recursive relationships $\mathcal{H}_t = \mathcal{H}_{t-1} \times \mathcal{A} \times \mathcal{Z}$ and $h_t = (h_{t-1}, a_{t-1}, z_t)$.

State estimation is a procedure by which a history h_t is mapped into a PDF over the state. In a general decision process, one may be interested in the PDF over all past states of the system, as they can all affect future states and rewards. In a partially observable Markov decision process (POMDP), the Markov property implies that a PDF over the current system state summarizes all relevant knowledge. This conditional PDF $p(s_t \mid h_t)$ is called the *belief state*. We adopt the notation $b_t(s_t) = p(s_t \mid h_t)$[3], and denote $\mathcal{P}(\mathcal{S}) = \mathcal{B}$, and call this set the belief space.

State estimation is carried out recursively as the process progresses. For brevity of notation, we refer in the following to s_t, a_t, b_t, z_{t+1}, and b_{t+1} as s, a, b, z', and b', respectively. Suppose we are given b, and the agent then executes an action $a \in \mathcal{A}$, and perceives $z' \in \mathcal{Z}$. The posterior belief state $b' \equiv p(s' \mid b, z', a)$ is given by the *belief update equation* $\tau : \mathcal{B} \times \mathcal{A} \times \mathcal{Z} \to \mathcal{B}$, defined via the Bayes' rule as

$$b' = \tau(b, a, z') = \frac{\mathbb{O}(z', s', a)p(s' \mid b, a)}{p(z' \mid b, a)}, \tag{2.4}$$

where $p(s' \mid b, a)$ is the predictive PDF of the state at the next decision epoch given the current belief state b and action a. This PDF is obtained from the Chapman-Kolmogorov equation (Brzeźniak and Zastawniak, 1999) as

$$p(s' \mid b, a) = \int_{\mathcal{S}} \mathbb{T}(s', a, s)b(s)\mathrm{d}s. \tag{2.5}$$

For finite $\mathcal{S}$, the integration is replaced by summation. The term $p(z' \mid b, a)$ in (2.4) is a normalisation term equal to the prior probability of observing z', obtained by

$$p(z' \mid b, a) = \int_{\mathcal{S}} \mathbb{O}(z', s', a)p(s' \mid b, a)\mathrm{d}s', \tag{2.6}$$

replacing integration by summation for finite $\mathcal{S}$.

The steps presented above outline a recursive procedure by which the agent's belief state may be tracked over histories of past actions and observations. A belief state b_t is a sufficient statistic for the history h_t. The belief state and history may be used interchangeably as representations of the agent's knowledge.

With the addition of observations, the MDP is a partially observable MDP, or a POMDP. As the state is not observable, the sets of allowed actions must instead depend on the history, or equivalently the belief state instead of the state. Furthermore, with the introduction of belief states we can allow reward functions that are also dependent on the belief state as opposed to the true underlying state of the system. With these modifications, the following definition of a POMDP is given.

[3]For $t = 0$, $b_0(s_0) = p(s_0)$.

Definition 2.5 (Partially observable Markov decision process (POMDP))**.** *A partially observable Markov decision process (POMDP) is a tuple* $\langle \mathcal{T}, \mathcal{S}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}, \mathbb{O}, R \rangle$*, where* $\mathcal{T}$ *is the set of decision epochs,* $\mathcal{S}$ *is the state space,* $\mathcal{A}_b$ *is the set of actions allowed in belief state* $b \in \mathcal{B}$ *such that* $\mathcal{A} = \bigcup_{b \in \mathcal{B}} \mathcal{A}_b$*,* $\mathcal{Z}$ *is the observation space,* $\mathbb{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ *is the state transition model,* $\mathbb{O} : \mathcal{Z} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ *is the observation model, and* $R : \mathcal{B} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is a real-valued reward function.*

Since the belief states of the POMDP are fully observed by the agent, a POMDP is equivalent to a MDP over belief states.

Lemma 2.6 (Belief MDP)**.** *A POMDP* $\langle \mathcal{T}, \mathcal{S}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}, \mathbb{O}, R \rangle$ *is equivalent to a MDP* $\langle T, \mathcal{B}, \{\mathcal{A}_b\}, \mathbb{T}_b, \rho \rangle$*, known as the belief MDP, where* $\mathbb{T}_b : \mathcal{B} \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}^+$ *is a state transition model for belief states defined*

$$\mathbb{T}_b(b', a, b) = \begin{cases} p(z' \mid b, a), & \text{for } b' = \tau(b, a, z') \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$

and $\rho : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$ *is the reward function defined as the expectation* $\rho(b, a) = \mathbb{E}_{s \sim b}[R(b, s, a)]$*.*

The state space in the belief MDP is the belief space $\mathcal{B}$ of the POMDP. Recall that $\mathcal{B} = \mathcal{P}(\mathcal{S})$. If the state space is finite, e.g. $|\mathcal{S}| = n \in \mathbb{N}$, the belief space is the $(n-1)$-dimensional unit simplex $\mathcal{B} = \left\{ v \in \mathbb{R}^n \mid \sum_{i=1}^{n} v_i = 1, v_i \geqslant 0 \right\} \subset \mathbb{R}^n$, which contains all PDFs over $\mathcal{S}$ (Lovejoy, 1991a). In case the state space is uncountable, e.g. an interval of $\mathbb{R}$, the belief space is instead a function space. Detailed discussion and proofs for Lemma 2.6 may be found e.g. in Bertsekas (1995, Ch. 5) for the case of finite $\mathcal{S}$, and in Bertsekas and Shreve (1996, Ch. 10) for the case of general $\mathcal{S}$.

2.1.3 Decision rules, policies, and optimality criteria

The MDP and POMDP definitions up to this point have avoided the discussion of how the agent actually should select the actions it applies. In general, the best course of action is such that it maximises some function of the rewards over the decision epochs $\mathcal{T}$ in the problem. To facilitate further discussion, we formalise the agent's decision-making procedure in general terms. The discussion of this subsection is in terms of the belief MDP (Lemma 2.6), since this allows a unified treatment of both fully observable and partially observable MDPs.

A *decision rule* δ_t describes the procedure of action selection at decision epoch $t \in \mathcal{T}$. The history contains all information available to the agent to select its next action. As seen earlier, in a POMDP the belief state summarises all knowledge contained in the history. In this thesis, all decision rules considered are deterministic. This means that decision rules are functions $\delta_t : \mathcal{B} \to \mathcal{A}$. A *policy* $\pi = (\delta_k)_{k \in \mathcal{T}}$ is a sequence of decision rules that specify how the agent acts at any decision epoch. For finite horizon problems with $\mathcal{T} = \{0, 1, \ldots, d\}$, $\pi = (\delta_0, \ldots, \delta_{d-1})$ as no decision is made at the last epoch. From now on we only consider policies that consist of admissible decision

rules that fulfil the technical conditions $\delta(b) \in \mathcal{A}_b \neq \varnothing, \forall b \in \mathcal{B}$. Define Π as the space of admissible policies fulfilling these conditions.

We define the value of a policy $\pi \in \Pi$ as the expected utility of the reward sequences obtained while acting according to the policy. Let $(r_t)_{t\in\mathcal{T}}$ denote a realisation of the random reward process in a MDP or POMDP. The discounted linear additive utility

$$\Psi(r_0, r_1, \ldots) = \sum_{t\in\mathcal{T}} \gamma^t r_t \tag{2.8}$$

is applied to quantify the utility of the reward sequence. The term $\gamma \geqslant 0$ is a discount factor that determines the time-preference for the rewards. For $0 \leqslant \gamma < 1$ the agent prefers immediate rewards to those obtained later, for $\gamma = 1$ there is no preference between immediate and future rewards, and $\gamma > 1$ indicates that future rewards are preferred. We remark that $\gamma \geqslant 1$ is only applicable for finite horizon problems, and lead to convergence issues in the infinite horizon case as the expected sum of rewards is not finite, even if the reward function is bounded. The discounted linear additive utility as defined in (2.8) is a suitable representation for the preferences of a risk neutral decision maker with the aforementioned possible attitudes toward timing of the rewards (Puterman, 1994).

Let $\rho : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$ be the reward function, and define $V_d^\pi : \mathcal{B} \to \mathbb{R}$ as the *value function* of a policy π, giving the expected value of following π for d decision epochs starting from a given belief state b_0. Based on the discounted linear additive utility (2.8) we write for a finite horizon

$$V_d^\pi(b_0) = \mathbb{E}\left[\sum_{t=0}^{d-1} \gamma^t \rho(b_t, \delta_t(b_t))\right], \tag{2.9}$$

where b_t evolves according to the belief update equation (Equation (2.4)). For a bounded reward function, the sum is finite. For a finite horizon, we allow $\gamma \geqslant 0$. For an infinite horizon, (2.9) is interpreted as a limit when $d \to \infty$. The limit exists when the reward function is bounded, and to guarantee that the value is finite we require $0 \leqslant \gamma < 1$. As the limit exists, we write the infinite horizon value function as V^π, defined as above but taking the sum up to ∞.

Besides finite and infinite horizon problems, *indefinite horizon* formulations are sometimes distinguished as well. The indefinite horizon formulation contains a stopping action that results in a transition to a state where all subsequent rewards are zero. This formulation avoids the use of the discount factor where it is merely a mathematical convenience and not otherwise justified in the problem (Hansen, 2007).

2.1.4 Optimal value functions and policies

A policy is optimal when its value function for any belief state is at least as great as for any other policy.

Definition 2.7 (Optimal policy)**.** *A policy $\pi^* = (\delta_0, \delta_1, \ldots, \delta_{d-1}) \in \Pi$ for d decision epochs is optimal if*

$$V_d^{\pi^*}(b) \geqslant V_d^\pi(b) \quad \forall b \in \mathcal{B} \text{ and } \forall \pi \in \Pi. \tag{2.10}$$

The value function $V_d^{\pi^*}$ is the finite horizon optimal value function. A similar definition is easily made for the infinite horizon value function V^{π^*}.

For the expected discounted reward formulation of the value function (Equation (2.9)), it can be shown that deterministic policies where decision rules are of the form $\delta_t : \mathcal{B} \to \mathcal{A}$ are optimal in the finite horizon case (Puterman, 1994), and in the infinite horizon case a stationary policy $\pi = (\delta, \delta, \delta, \ldots)$ with $\delta : \mathcal{B} \to \mathcal{A}$ is optimal under mild conditions, e.g. measurable state, action, and observation spaces (see for instance Bertsekas and Shreve, 1996, Ch. 10, Feinberg, 2002 and Feinberg et al., 2013).

Optimal value functions and optimal policies may be found via the Bellman equations, also known as the optimality equations (Puterman, 1994). The equations derive their name from Bellman's (1957) principle of optimality. The principle of optimality enables a backward in time recursive procedure, by which the optimal value function and optimal policy for any finite horizon may be extracted. The principle of optimality states that "whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

Suppose the agent is in belief state b at decision epoch $(d-1)$. As the agent is at the last decision epoch and no decisions remain beyond the current one, an optimal action is one that maximises the immediate reward $\rho(b, a)$. The decision rule for this decision epoch is $\delta_{d-1}(b) = \underset{a \in \mathcal{A}_b}{\operatorname{argmax}} \rho(b, a)$, and the optimal value function is $V_1^{\pi^*}(b) = \rho(b, \delta_{d-1}(b))$. We now apply the principle of optimality to derive the optimal value function and policy for decision epoch $(d-2)$. The optimal value function must maximise the sum of immediate rewards and the expected rewards at the next decision epoch $(d-1)$:

$$V_2^{\pi^*}(b) = \max_{a \in \mathcal{A}_b} \left(\rho(b, a) + \gamma \mathbb{E} \left[V_1^{\pi^*}(\tau(b, a, z')) \right] \right), \tag{2.11}$$

where $\tau(b, a, z')$ is the belief update equation (2.4), and the expectation is taken with respect to the state transition model $\mathbb{T}_b$. As seen from Lemma 2.6, for POMDPs the expectation is taken under the prior PDF of the measurements (2.6). It is therefore a function of both the state transition model $\mathbb{T}$ and the observation model $\mathbb{O}$. For MDPs with a fully observable state, the expectation is taken w.r.t. $\mathbb{T}$.

Equation (2.11) states that the optimal action at decision epoch $(d-2)$ is the one which maximises the sum of immediate rewards and the discounted expected rewards at decision epoch $(d-1)$. The optimal policy is augmented with a decision rule $\delta_{d-2}(b)$ that is found by selecting the argument $a \in \mathcal{A}_b$ maximising (2.11).

The procedure presented above can be extended for an arbitrary finite horizon d. The procedure is a backwards in time recursion called *value iteration*. To avoid confusion with the decision epochs, we define $Q_k : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$ as the expected value of executing action a in belief state b with k decision epochs remaining, when an optimal policy is followed for the rest of the $(k-1)$ decisions. The function $Q_k(\cdot, a)$ is also called the Q-value function of an action $a \in \mathcal{A}$. The value iteration is initialised for $k = 1$ by setting $Q_1(b, a) = \rho(b, a)$. We then proceed for $k = 2, \ldots, d$ by iterating the following

equations:

$$Q_k(b,a) = \rho(b,a) + \gamma \mathbb{E}\left[V_{k-1}^{\pi^*}(\tau(b,a,z'))\right] \tag{2.12}$$

$$\delta_{d-k}(b) = \underset{a \in \mathcal{A}_b}{\operatorname{argmax}}\, Q_k(b,a) \tag{2.13}$$

$$V_k^{\pi^*}(b) = \max_{a \in \mathcal{A}_b} Q_k(b,a). \tag{2.14}$$

Value iteration can be used to extract the optimal value function $V_k^{\pi^*}$ and an optimal policy $\pi^* = (\delta_0, \ldots, \delta_{d-2}, \delta_{d-1})$ for any finite horizon. The decision rule index is $(d-k)$ to relate it to the true decision epochs in $\mathcal{T}$.

For the infinite horizon case, we may define a dynamic programming operator L mapping $V_{k-1}^{\pi^*}$ to $V_k^{\pi^*}$ as in Equations (2.12)-(2.14). For $0 \leqslant \gamma < 1$, L is a contraction mapping and the optimal infinite horizon value function V^{π^*} is a unique fixed point of L satisfying $V^{\pi^*} = LV^{\pi^*}$. The optimal policy is stationary. Furthermore, approximations for V^{π^*} may also be found by value iteration as when $k \to \infty$, $V_k^{\pi^*} \to V^{\pi^*}$. We refer the reader to Puterman (1994) and Feinberg et al. (2013) for further details.

2.1.5 Classification of Markovian decision processes

A POMDP with just one action is effectively uncontrollable, and corresponds to a hidden Markov model (HMM). HMMs are often applied as models for sequential signals e.g. in speech recognition or handwriting detection (Bishop, 2006). A POMDP with a single action and fully observable state is in fact a Markov process. The special case of a POMDP where $\mathcal{Z} = \mathcal{S}$ and the observation model is an identity mapping such that each observation uniquely identifies the true underlying state, the state is effectively fully observable and the POMDP reduces to a MDP. Table 2.1 summarises this classification of Markov decision processes.

Table 2.1: Classification of Markovian decision processes according to the cardinality of the action space $\mathcal{A}$ and observability of the state.

	State known	State hidden
$\lvert\mathcal{A}\rvert = 1$	Markov process	HMM
$\lvert\mathcal{A}\rvert > 1$	MDP	POMDP

Decision processes containing a single decision making agent may be viewed as a two-player "game against nature" (Papadimitriou, 1985). In a game against nature, the agent is attempting to maximise some performance measure against an opponent that is disinterested, following a random strategy. The opponent's moves in this game correspond to selecting next states according to the state transition model $\mathbb{T}$.

If multiple decision-makers are involved, the situation becomes more complex. We direct the interested reader to Oliehoek (2010); Goldman and Zilberstein (2004) for cooperative multi-agent decision making, and to Emery-Montemerlo (2005); Hansen et al. (2004) for competitive multi-agent decision making.

2.2 Solving POMDPs

From the point of view of computational complexity, computing an optimal policy for a finite horizon POMDPs is known to be PSPACE-hard while the special case without observations is NP-hard[4] (Papadimitriou and Tsitsiklis, 1987). It was also conjectured by Papadimitriou and Tsitsiklis and later shown by Madani et al. (1999, 2003) that finding an optimal policy in the infinite horizon case is undecidable[5]. Madani et al. concluded their analysis with a suggestion that it may be promising to study restrictions on the state transition model and observation model among other model parameters, that are useful in realistic problem domains and simultaneously easier to tackle by algorithms finding optimal or approximately optimal policies. An overview of such algorithms and the principles they are based on is our goal in this section.

The value iteration procedure presented in Subsection 2.1.4 is directly applicable only to MDPs with finite state spaces, where the value iteration equations can be solved for each state (Puterman, 1994). In contrast, POMDPs are equivalent to belief MDPs with an uncountable state space (see Lemma 2.6). Therefore, it is not possible to compute the optimal policy and value function via exhaustive enumeration of belief states. In this section, we review practical methods of finding optimal value functions and policies for POMDPs.

For the majority of the section, we limit the discussion to POMDP models with finite state, action, and observation spaces. General POMDPs with possibly uncountable state, action, and observation spaces are briefly covered in Subsection 2.2.6. Unless otherwise specified, we assume that the action space is not restricted and all actions are always applicable, i.e. $\forall b \in \mathcal{B} : \mathcal{A}_b = \mathcal{A}$.

In sensor management problems, information theoretic quantities such as mutual information or entropy of the belief state are interesting reward functions. However, these functions are non-linear in the belief state. Hence, we note for each algorithm whether it is applicable to such non-linear rewards.

A typical distinction is to divide the algorithms into offline and online algorithms. Exact offline algorithms find the optimal value function $V^{\pi*}$ and an optimal policy π^* for all belief states before beginning task execution. While executing the task, the agent simply uses the optimal policy like a look-up table to select actions conditional on the belief state. Exact online algorithms interleave task execution and finding optimal actions. In each belief state, the agent finds only the currently optimal action and executes

[4]Decision problems in NP are such that there is some algorithm that can guess an answer and then verify its correctness in polynomial time. A problem is called NP-hard if any other decision problem in NP can be reduced to it in polynomial time. Decision problems in PSPACE are solvable using an amount of memory polynomial in the size of the input instance. A problem is PSPACE-hard if any other decision problem in PSPACE can be reduced to it in polynomial time. It is widely conjectured that the complexity classes NP and PSPACE are not equal, and problems in PSPACE are harder than those in NP (Russell and Norvig, 2010).

[5]There exists no single algorithm always giving the correct yes-or-no answer to the question "Is there a policy for a POMDP with an expected value greater than some given threshold?"

it, repeating the procedure at each subsequent decision epoch. Approximate versions of both types of algorithms act in a similar manner, but instead merely provide approximately optimal solutions, possibly with a bounded error compared to an optimal solution.

The algorithms in Subsections 2.2.1-2.2.3 are offline, while algorithms in Subsection 2.2.4 are online. In all cases, the solution algorithms may be either exact or approximate. In Subsection 2.2.5 some special cases of POMDPs are reviewed. Subsection 2.2.6 concludes the discussion by reviewing suitable approaches to general POMDPs with uncountable state, action, and observation spaces.

2.2.1 Linear programming and policy iteration

Consider a POMDP with a finite state space $\mathcal{S}$, action space $\mathcal{A}$, and observation space $\mathcal{Z}$. Suppose $|\mathcal{S}| = n$, resulting in a belief space $\mathcal{B}$ which is a finite-dimensional subset of $\mathbb{R}^n$. Åström (1969) determined that in such a POMDP the optimal finite horizon value function $V_k^{\pi *}$ for any k is convex. In the early 1970s, Sondik (1971) and Smallwood and Sondik (1973) showed that the value function is piecewise-linear and convex (PWLC), and derived a linear programming method for computing it. The idea is that the value function has a finite representation by a set Γ_k of so-called α-vectors, each of which represents a $|\mathcal{S}|$-dimensional hyperplane. The value function is the convex hull of the set Γ_k:

$$V_k^{\pi *}(b) = \max_{\alpha \in \Gamma_k} \sum_{s \in \mathcal{S}} \alpha(s) b(s). \tag{2.15}$$

Another way to view (2.15) is that Γ_k induces a partition of $\mathcal{B}$ into regions where in each region a single α-vector dominates the others. Later, Sondik (1978) showed that the value function in an infinite horizon discounted case may also be approximated arbitrarily closely by a PWLC function. These results by Sondik and Smallwood have since been exploited to derive a myriad of solution algorithms for POMDPs, both exact and approximate.

Smallwood and Sondik applied in their studies a reward function that only depends on the true state and action, i.e. $R(s, a)$. In this case according to Equations (2.12)-(2.14), Definition 2.5, and Lemma 2.6, $\Gamma_1 = \{R(\cdot, a)\}_{a \in \mathcal{A}}$. The sets Γ_k are then computed applying the representation of Equation (2.15) of the value function to Equation (2.14), see e.g. Smallwood and Sondik (1973) for details.

Given Γ_k, the linear programming algorithm applied e.g. by Smallwood and Sondik (1973) generates $|\mathcal{A}||\Gamma_k|^{|\mathcal{Z}|}$ new α-vectors that may be part of Γ_{k+1}. As implied by (2.15), it may well be that some of the linear functions $\alpha \in \Gamma_{k+1}$ are dominated by others, and removing them from the set Γ_{k+1} does not change the PWLC function $V_{k+1}^{\pi *}$. The identification and removal of such redundant α-vectors helps improve the computational efficiency of the algorithm.

A related but complementary approach is to attempt to identify so-called witness beliefs, which are belief states that support a given α-vector and thus provide evidence for the existence of the region where the α-vector dominates. Instead of enumerating all α-vectors and then pruning them,

only the necessary ones as indicated by the witness beliefs are computed. Examples of these types of methods include e.g. Cheng (1988), Cassandra et al. (1997), and Kaelbling et al. (1998), while Monahan (1982) and Lovejoy (1991a) review several other approaches as well. However, the exponential growth in the number of α-vectors and the difficulty of identifying redundant α-vectors usually limits the applicability of such methods to problems with at most in the order of a few dozen states, actions, and observations (Hauskrecht, 2000; Pineau et al., 2006).

The crucial requirement for (2.15) to hold is that the expectation of the reward function must be linear in $b \in \mathcal{B}$. Only then the value function is PWLC. This clearly holds for reward functions of the form $R(s, a)$. Consider then the belief MDP reward function $\rho(b, a) : \mathcal{B} \times \mathcal{A} \to \mathbb{R}$. Now $\rho(b, a)$ is linear in $b \in B$ if and only if it is of the form $\sum_{s \in \mathcal{S}} f(s, a)b(s) + g(a)$, where $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $b(s)$ is the belief state and $g : \mathcal{A} \to \mathbb{R}$ is an action reward or cost. Since $\sum b(s) = 1$, we have $\sum cb(s) = c$ for any $c \in \mathbb{R}$, and

$$\sum_{s \in \mathcal{S}} f(s, a)b(s) + g(a) = \sum_{s \in \mathcal{S}} (f(s, a) + g(a))b(s) = \mathbb{E}\left[f(s, a) + g(a)\right]. \quad (2.16)$$

We conclude that the representation (2.15) is only valid for POMDPs where the reward is dependent on the hidden state and action only.

An alternative to value iteration is to iterate over the space of admissible policies. This policy iteration procedure (Howard, 1960) consists of two steps: policy evaluation and policy improvement. In the policy evaluation step, the value of the current policy is determined. The policy improvement step solves (2.13), thus updating the policy. The iteration is started with an arbitrary policy. Sondik (1978) presented an policy iteration algorithm for discounted infinite horizon POMDPs, representing the policy as a mapping from a finite number of polyhedral regions of $\mathcal{B}$ to $\mathcal{A}$. Hansen (1998a,b) proposed representing the policy as a finite state controller, making the policy evaluation step easier to implement.

2.2.2 Point-based methods

A natural idea for approximating the optimal value function is to determine it only over a finite subset $B_R \subset \mathcal{B}$ instead of the complete belief space. The value iteration is only performed on the subset B_R instead of the complete belief space. A method for generalising the value function at belief states in $\mathcal{B} \backslash B_R$ may be applied, based e.g. on interpolation. Such methods are collectively termed *point-based* POMDP solution methods. In the following, we review some of the point-based POMDP solvers proposed in the literature. A recent, thorough survey on the topic is provided by Shani et al. (2013).

Some point-based approaches apply a discretisation of the belief space. For instance, Lovejoy (1991b) generated a fixed grid over $\mathcal{B}$ via triangulation and approximated the value function on the grid. A variable resolution grid that is denser in some parts of $\mathcal{B}$ was considered by Zhou and Hansen (2001). If an α-vector representation is applied, Equation (2.15) provides a recipe for generalising to beliefs not in B_R. Grid-based approaches that rely purely on interpolation-extrapolation rules for generalisation and do not make use

of the α-vector representation may be applied even to problems where the expected reward is non-linear in $\mathcal{B}$.

In approaches that do apply the α-vector representation of the value function, exactly one α-vector is maximal at a given belief state. Thus, the value function is represented by at most $|B_R|$ vectors, reducing computational demands. The value function is exact for $b \in B_R$, and by (2.15), an approximation of the value function for belief states in $\mathcal{B} \backslash B_R$ may be found.

Several point-based methods are based on the idea of setting B_R equal to the set of reachable belief states in the POMDP. Initial information about the state in a POMDP is summarised by an initial belief state b_0. Given b_0, reachable belief states are those that can be reached by executing an admissible policy.

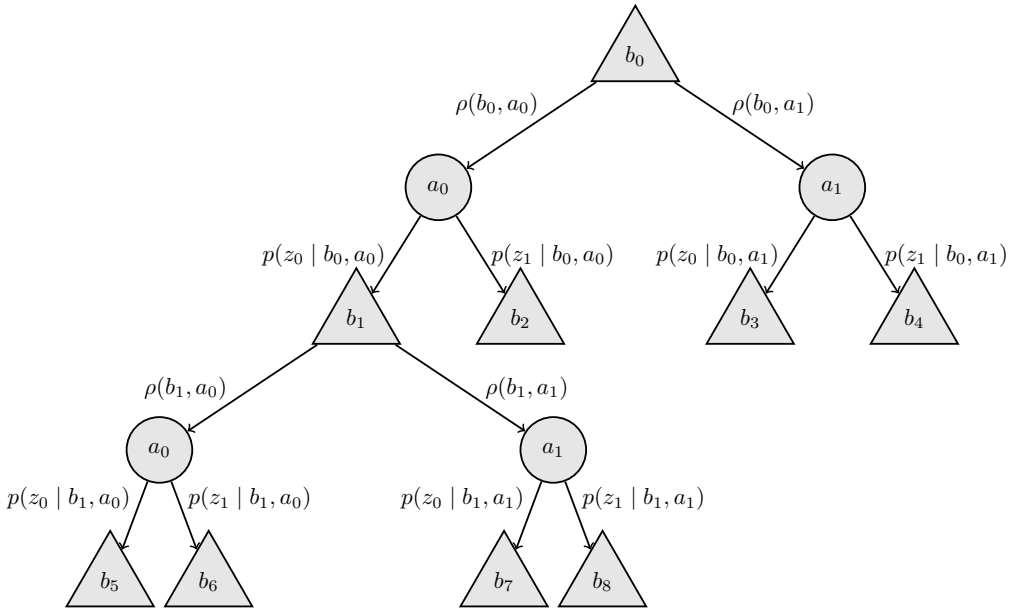


Figure 2.1: A partial belief tree. Belief states are depicted as triangular nodes, and actions are depicted by circular nodes. Each edge is labelled, from a belief node to an action by the expected reward of executing the action in the belief state, and from actions to belief nodes by the probability of reaching each possible successor belief state.

The concept is further illustrated by considering a tree graph representation of reachable belief states. Consider a POMDP with action space $\mathcal{A} = \{a_0, a_1\}$ and observation space $\mathcal{Z} = \{z_0, z_1\}$. An example of a partial tree graph of reachable belief states over 2 decisions for such a POMDP is shown in Figure 2.1. Belief states in the tree are depicted by triangular nodes. The initial belief state is b_0, shown at the root of the tree. A belief node has child nodes labelled by actions. An edge from a belief node to an action node is labelled with the expected reward of the action in the parent belief state, for example $\rho(b_0, a_0)$ at the upper left hand side of the figure. The child nodes of action nodes are again belief nodes. There are $|\mathcal{Z}|$ out-edges from an action node, one for each possible observation. Each edge from an action node to a belief node is labelled with the prior probability of perceiving that observation, for instance $p(z_0 \mid b_0, a_0)$ between nodes labelled a_0 and b_1. A pair of edges from a belief node via an action node to a child belief node determines a belief state. For example, the leftmost belief node in the third

layer of the tree corresponds to action a_0 at belief b_0 and observation z_0. Hence, $b_1 = \tau(b_0, a_0, z_0)$ as defined in (2.4). Based on the tree view presented, we conclude that if the agent executes d actions, there are $(|\mathcal{A}||\mathcal{Z}|)^d$ possible reachable belief states.

Beyond the third layer of the tree, only the children of b_1 have been expanded and drawn in the tree. In a fully expanded belief tree, the belief states corresponding to $(d + 1)$ topmost layers of belief nodes in the tree form the set of belief states that can be reached by any combination of up to d decisions and observations starting from the initial belief state b_0. For example for $d = 1$ this set in the case of Figure 2.1 is $\{b_0, b_1, b_2, b_3, b_4\}$.

As it is rarely feasible to consider all of the reachable belief states, alternative ways to select a suitable subset of them as B_R have been proposed instead.

Pineau et al. (2006) suggest interleaving point-based value iteration steps with inclusion of more belief states into B_R. Their point-based value iteration (PBVI) algorithm starts with an initial set $B_R = \{b_0\}$ of beliefs, performs a finite number of value iteration steps, then inserts new belief states into B_R and repeats the procedure. They define a density for the belief set B_R as

$$\Delta_{B_R} = \max_{b' \in \mathcal{B}} \min_{b \in B_R} ||b - b'||_1. \tag{2.17}$$

This density determines how well B_R covers $\mathcal{B}$. When selecting which belief point $\tau(b, a, z')$ to insert to B_R between value iteration stages, Pineau et al. take into account 1) how likely it is to reach the belief state $\tau(b, a, z')$ from $b \in B_R$, i.e. $p(z' \mid b, a)$, 2) to minimise Equation (2.17), how far the belief state $\tau(b, a, z')$ is from other beliefs already in B_R, and 3) what is the current approximate value at $\tau(b, a, z')$.

A related method based on randomising the value iteration stages was suggested by Spaan and Vlassis (2005). The set B_R is generated by sampling random trajectories of reachable beliefs. The algorithm is given an initial value function V_1 in terms of α-vectors. At iteration k, instead of computing the value iteration step for each belief state in B_R, the algorithm randomly chooses a belief state $b \in B_R$ and implements the value iteration step for that belief. The related α-vector is added to the value function estimate V_{k+1}. The new α-vector may also improve the value function also at belief points other than b. Every $b \in B_R$ is checked for whether their value was improved, obtaining a set $\tilde{B} = \{b \in B_R \mid V_{k+1}(b) < V_k(b)\} \subset B_R$ of beliefs whose value has not yet been improved. A new belief is chosen randomly from $\tilde{B}$, the value iteration step is computed for it, the α-vector is inserted to V_{k+1}, and the beliefs are checked again. This process is repeated until $\tilde{B} = \varnothing$, and the value of every belief state has been improved.

The algorithm presented above, called Perseus, features an asynchronous value iteration stage since the beliefs in B_R are processed in a random order, and not necessarily an equal number of times. Regrettably, as a result the concept of a fixed planning horizon is obfuscated: performing the Perseus value iteration stage k times will not consider policies k steps into the future, but less by some amount (Spaan and Vlassis, 2005). The algorithm however eventually converges to the optimal value function V^{π^*}, and is thus suited for discounted infinite horizon problems.

Further examples of point-based methods include SARSOP (Kurniawati et al., 2008), which attempts to further reduce the computation of the value function to the set of belief states reachable under an *optimal* policy instead of an arbitrary policy, or a random policy as in Perseus. Heuristic search value iteration (HSVI) (Smith and Simmons, 2004, 2005) follows the simple rule of updating the successors of a belief state, i.e. the children of any belief state in the tree of Figure 2.1 before the belief itself, accelerating the convergence of value iteration. Additionally, HSVI applies a heuristic function to select the most relevant belief points to be included into B_R. The heuristic is computed by maintaining lower and upper bound for the value function, and new beliefs to include in B_R are selected based on maximising the distance between the bounds. In other words, belief points are included in areas of $\mathcal{B}$ where the uncertainty about the value function is the greatest.

Araya-López et al. (2010) suggested a variant of POMDP, called ρPOMDP, with a reward function that is convex in the belief state. The convex reward function may be approximated by a PWLC function, and regular α-vector updates or point-based value iteration steps may be applied to obtain a bounded-error approximation of the value function. Ji et al. (2007) derived a policy iteration algorithm replacing Hansen's policy improvement step by point based value iteration.

2.2.3 Problem compression

Roy et al. (2005) note that while much of the difficulty related to finding policies for POMDPs via value iteration is due to computing the value function over the complete belief space, in many problems typical belief states encountered lie on a low-dimensional subspace. They suggest applying exponential family principal component analysis (E-PCA) to find a mapping from the belief space to a low-dimensional subspace, reducing belief dimensionality. The mapping is found minimising a generalised Bregman divergence between the low- and high-dimensional representations of a collected sample of belief states. As the transformation of belief states is non-linear, convexity properties of the value function are lost in the low-dimensional subspace and α-vector representations cannot be used. Instead, Roy et al. (2005) approximate the problem as a MDP with a finite number of states, discretising the low-dimensional but still continuous belief space to a finite number of states. They learn a new state transition model and reward function that correspond to the discretisation, and solve the resulting MDP by value iteration.

The E-PCA method of belief compression exploits the sparsity in the belief space. A complementary idea is to use a value-directed belief compression method that distinguishes only between belief states that have a different value. An approach introduced by Poupart and Boutilier (2002) and Poupart, 2005, Ch. 4 learns a low-dimensional representation of a POMDP directly from the parameters $\mathbb{T}$, $\mathbb{O}$, and R. The method finds the smallest subspace of $\mathcal{B}$ which contains the immediate reward vector and is closed under the state transition and observation models, resulting in a lossless compression that distinguishes belief states that have a different value.

When the conditions required by such belief space compression methods are fulfilled, they have been shown to outperform other approaches (Roy et al., 2005; Poupart, 2005). However, if typical beliefs lie on multiple distinct low-dimensional surfaces on $\mathcal{B}$, a higher apparent dimensionality than actually exists is detected by the methods. It is also not always possible to use standard value iteration methods to find the solution of the compressed problem, as is the case e.g. in (Roy et al., 2005), or the compression itself may be computationally infeasible for large enough POMDPs (Poupart, 2005).

Large POMDPs are often represented in a factored form using Bayes networks (Pearl, 1988; Bishop, 2006) to model interactions between state variables, actions and observations. Structural properties of such representations, such as conditional independence, context-specific independence, or additive separability, may be taken advantage of to effectively find solutions. For instance, (Poupart, 2005, Ch. 5) explored the use of automatic methods to detect and exploit such properties.

As the α-vector representation is central to all of the problem compression methods reviewed here, they are not suitable for problems with non-linear belief dependent reward functions.

2.2.4 Online methods

The algorithms reviewed so far are offline. Such an offline planning approach is feasible when there is abundant time to spend computing the policy. A drawback of offline planning is that if a change occurs in the POMDP, e.g. in the state transition model, the policy must be computed again and task execution possibly has to be stopped during that time.

In case of partially unknown models or models undergoing non-deterministic variations, the use of offline algorithms is questionable. The issue may sometimes be mitigated by introducing a prior on the unknown parameters (Ross et al., 2007) that may be updated based on observed data, but also *online* algorithms may be applied in such situations.

Online algorithms interleave planning and plan execution. The idea is to reason about possible future outcomes only for the current belief state, instead of all possible belief states. This results in a local search in the belief space starting from the current belief state, and consequently online algorithms are sometimes also called agent-centered search algorithms (Koenig, 2001). Since planning is interleaved with execution, online planners must usually be capable of meeting strict real-time constraints, or at least have anytime qualities. An anytime algorithm can trade off computation time for quality of results, and can provide an answer at any desired time (Zilberstein, 1996). We remark that time in this paragraph refers to the runtime of the algorithm measured e.g. in seconds, and not the amount of decision epochs.

In such a local search it is not necessary to maintain a closed form representation of neither the value function nor the optimal policy. As such, many online algorithms are well suited to POMDPs where a closed form representation of the value function e.g. via α-vectors is not available. Nevertheless, some online planning approaches not based on look-ahead search learn from

past experience and agglomerate a database of optimal values and policies for the belief states encountered thus far, see e.g. Geffner and Bonet (1998); Bonet and Geffner (2009) for examples.

Most online planning approaches applied for POMDPs are based on a finite-depth look-ahead search (Ross et al., 2008). A look-ahead search algorithm is run starting from the current belief state b up to a depth of $d \in \mathbb{N}$ decision epochs. The search is over the tree of possible future belief states, as shown in Figure 2.1. Besides the immediate reward $\rho(b, a)$, it is useful to maintain estimates of the Q-values $Q_k(b, a)$, $k = 1, \ldots, d$, of each action in a belief state in the search tree. These values may be computed while constructing the tree by applying Equations (2.12)-(2.14) together with the observation probabilities.

As mentioned above, online methods typically suffer no penalties from changing the POMDP parameters during task execution, compared to offline methods that require re-computation of the policy. Complex constraints on the applicable actions based e.g. on the history of actions induce an exponential increase in the size of the state space. For instance, if each action can be executed at most n times, the state space must be increased by a factor $|\mathcal{A}|^n$ to keep track of the actions executed thus far. This increases complexity of computing a policy offline, whereas an online algorithm can handle the constraints only to the extent they are encountered during the tree search.

For a finite horizon problem the search tree is finite, and an algorithm that is guaranteed to return the optimal solution may be designed. Even for infinite horizon problems, bounded error approximations for the value function are obtained by searching up to a finite depth in the tree (Hauskrecht, 2000).

A generic online algorithm as adapted from Ross et al. (2008) is shown in Algorithm 2.1. The algorithm is divided to a planning phase (Lines 5-9) and an execution phase (Lines 10-14). The algorithm is given as input the current or initial belief state b_0 and a search depth $d \in \mathbb{N}$. The algorithm repeatedly finds the best action at the current belief state, executes the action, perceives an observation, and revises the current belief state, until execution is terminated. Execution is terminated when the function ExecutionTerminated (Line 14) returns a true value. An event leading to the termination of plan execution may be e.g. reaching some goal belief state (see Bonet and Geffner (2009) for discussion on goal-oriented POMDPs) or reaching a specified maximum number of actions executed, depending on how the problem is set up.

The algorithm is initiated by setting the initial belief as the active belief state b (Line 2) and initialising the search tree to containing only b as its root node (Line 3, see Figure 2.1 for an example illustration of a search tree).

The planning phase is executed until the condition given by the function PlanningTerminated returns a true value. The condition for stopping planning may be e.g. completing an exhaustive search of the reachable beliefs until the search depth d, reaching a specified real-time constraint that limits the maximum time allotted for planning, or finding a solution with a lower error bound than specified. The planning phase consists of three steps.

At first, a leaf node in the current search tree is selected under which the algorithm should perform a forward search (Line 6). Denote this leaf node by the belief state b^* related to it. The EXPAND function then extends the search tree by including reachable belief states starting from b^* (Line 7) by means of Equation (2.4), until the desired search depth d. By applying Equation (2.14), the value of b^* is determined while expanding, and the function UPDATEANCESTORS finally propagates the newly obtained value to update the ancestors of b^* in the tree. More concretely, this step updates the Q-values of ancestor nodes of b^* to take into account the reward information obtained while expanding the tree.

Algorithm 2.1 A generic online POMDP algorithm (Ross et al., 2008).

Input: The initial belief state b_0, the search depth d.

```
1: function OnlinePOMDPSolver(b_0, d)
2:     b ← b_0
3:     Initialise a search tree T to contain only b at the root
4:     repeat
5:         while not PlanningTerminated( ) do
6:             b* ← ChooseNextNodeToExpand( )
7:             Expand(b*, d)
8:             UpdateAncestors(b*)
9:         end while
10:        Execute best action â found for b
11:        Observe z
12:        b ← τ(b, â, z)
13:        Update T such that b is the new root
14:    until ExecutionTerminated( )
15: end function
```

Once the planning phase terminates, the best action $\hat{a}$ is executed (Line 10). The best action is found as in Equation (2.13), by $\hat{a} = \text{argmax}_{a \in \mathcal{A}} Q_d(b, a)$, where Q_d is the Q-value function for d remaining decision epochs. As the system transitions to a new state, an observation is emitted and perceived (Line 11). The belief state is revised accordingly (Line 12), and the search tree is updated such that it now contains the new belief state b at the root. If the POMDP model remains the same, the remainder of the search tree under b remains valid and may be preserved. Subsequent expansions may take advantage of this already computed part of the search tree.

The online algorithms for POMDPs differ in how the functions CHOOSENEXTNODETOEXPAND, EXPAND, and UPDATEANCESTORS are implemented. In the following, we first discuss briefly methods of obtaining upper and lower bounds for the optimal value function in a POMDP. Upper and lower bounds may be applied by online algorithms to guide the search process. We then review some online methods in more detail. For a comprehensive survey of online algorithms for POMDPs, we refer the reader to Ross et al. (2008).

2.2.4.1 Bounds for the optimal value function

Let us denote a lower and upper bound for $V^{\pi^*}(b)$ by $L(b)$ and $U(b)$, respectively. Alternative strategies to compute such bounds are surveyed in detail by Hauskrecht (2000).

To obtain lower bounds, one can e.g. compute the value function following a blind policy that always executes the same action, or a greedy policy that always chooses the action maximising the expected immediate reward. It is also possible to apply the offline methods presented in the previous subsections to find a lower bound for the optimal value function. Hauskrecht notes that an approximation via an unobservable Markov decision process (UMDP) provides a lower bound. An UMDP is obtained when the observations are removed from a belief MDP. This also changes the nature of the solution: as there are no observations, the policies for UMDPs are simply sequences of actions, as no observation contingencies need to be considered. In control theory, such a case is known as an open loop control problem. The policy-based bounds are applicable to problems with a non-linear belief-dependent reward, and bounds based on offline methods are applicable under constraints outlined in the preceding subsections.

Littman et al. (1995) introduced two methods for finding upper bounds for POMDPs. The first method completely ignores uncertainty in state estimation solving instead the underlying fully observable MDP by value iteration, obtaining an MDP value function. The upper bound $U(b)$ for the POMDP is then found by taking the expectation of the MDP value function under b. The second method, called QMDP, assumes that state uncertainty disappears after a single decision epoch and computes the bound accordingly. As these bounds ignore the effect of the observation model, they do not take into account any information gathering actions. Hauskrecht (2000) suggested the fast informed bound (FIB) based on an α-vector update procedure that uses the observation model as well. Due to the reliance on the underlying MDP, a state-dependent reward-function, and the α-vector representation, none of these bounds is applicable for problems with non-linear belief-dependent rewards.

Value functions obtained from other solution methods, e.g. point based solvers, can also be applied as lower bounds. For example, one could obtain a lower bound by sampling a set of belief points and executing a point-based solver on this set. An online method can then use this bound as a starting point for improving value estimates.

2.2.4.2 Open loop feedback control

Applying the information provided by observations helps to achieve a greater discounted total reward in a POMDP. However, as seen from the value iteration procedure (Equations (2.14)-(2.13)) and the branching of the belief tree (Figure 2.1) due to the observations, the use of this information increases the computational demands for computing an optimal policy. A lower bound for the optimal value is obtained by ignoring the availability of the information provided by observations. This leads to an open loop control problem, as mentioned above in the context of the UMDP. As observations are ignored, the optimal solution is a sequence of actions instead of a policy contingent on observations.

According to Bertsekas (2005), the open loop feedback control (OLFC) scheme was first proposed by Dreyfus (1965). OLFC does not consider the effect that possible future observations may have in determining the current action a_t. At decision epoch t, it proceeds by solving a finite horizon open

loop control problem of finding an action sequence $a^*_{t:t+d-1}$ for the next $d \in \mathbb{N}$ decision epochs, which maximises

$$J_d(b_t, a_{t:t+d-1}) = \mathbb{E}\left[\sum_{i=t}^{t+d-1} \gamma^{(i-t)}\rho(b_i, a_i) \mid b_t\right], \tag{2.18}$$

i.e. the expected reward of an action sequence $a_{t:t+d-1}$ given the current belief state b_t. The best action a^*_t found is then applied, an observation z_{t+1} from the process is observed and the belief state is updated to $b_{t+1} = \tau(b_t, a^*_t, z_{t+1})$. The process is then repeated at decision epoch $(t+1)$.

For infinite horizon problems, it is not feasible to find the infinite optimal open loop action sequence. Instead, the receding horizon control (RHC) principle is applied: at each decision epoch, a finite look-ahead depth of d decision epoch is employed as in Equation (2.18). The RHC principle is also applied in model predictive control (MPC) (Maciejowski, 2002), originally motivated by the desire to introduce non-linearities and constraints into the linear-quadratic control framework (Bertsekas, 2005).

In the context of the generic online algorithm (Algorithm 2.1), OLFC always chooses the current belief node to expand, implements the Expand function by Equation (2.18) and performs no operation while updating the ancestors of the current belief.

The OLFC control scheme has been shown to perform at least as well as simply executing the optimal open loop action sequence $a^*_{t:t+d-1}$ (Bertsekas, 2005).

Recently, Kantas et al. (2009) proposed a sequential Monte Carlo (SMC) variant of OLFC to deal with continuous action and observation spaces. A set of particles representing action sequences is maintained, and Monte Carlo approximations are applied to evaluate the expected reward of these sequences. Any type of reward function may be applied, including those non-linear in the belief state.

The approach is related to the Bayesian interpretation of the simulated annealing solution to maximum likelihood estimation (Johansen et al., 2008). The key idea is to construct a sequence of distributions

$$\lambda_\eta(b_t, a_{t:t+d-1}) \propto p(b_t, a_{t:t+d-1}) J_d(b_t, a_{t:t+d-1})^\eta, \tag{2.19}$$

where $p(b_t, a_{t:t+d-1})$ is an arbitrary prior that is non-zero at the maximisers of Equation (2.18)[6]. As $\eta \to \infty$, $\lambda_\eta(b_t, a_{t:t+d-1})$ becomes concentrated on the set of maximisers of J_d. In practice, the algorithm evaluates a set of particles via the objective function and based on the evaluation result particles are either discarded or carried forward to the next evaluation round. Eventually the particle set converges to represent an action sequence maximising (2.18).

The ChooseNextNodeToExpand-subroutine of the SMC-OLFC algorithm always returns the current belief state b_t. The Expand-subroutine of the algorithm is shown in Algorithm 2.2. The function takes as input parameters a belief state b and a search depth d.

[6]Note that b_t is a fixed parameter, so the maximisation is over the action sequences.

Additionally, the algorithm requires as static parameters a generative model $(s', z', r) \sim \mathcal{G}(s, b, a)$ for the POMDP[7] that produces samples of the next state, observation and reward, the number of particles M, the number of iterations $l_{\max}$, an increasing integer sequence $\{\eta_l\}_{l=1}^{l_{\max}}$, and a set of sampling kernels $\{q_l\}_{l=1}^{l_{\max}}$.

The number of iterations $l_{\max}$ is chosen according to the accuracy and runtime requirements. The algorithm has been shown to converge to the optimal solution for logarithmic sequences of integers η_l, although in practice faster increasing linear or geometric sequences are used (Johansen et al., 2008; Kantas et al., 2009). This integer sequence is analogous to the temperature cooling schedule in simulated annealing.

Algorithm 2.2 The expansion subroutine of the SMC-OLFC algorithm of (Kantas et al., 2009).

Input: The current belief state b, the search depth d.

1: **function** EXPAND(b, d)
2: **for** $l = 1, \ldots, l_{\max}$ **do**
3: **for** $i = 1, \ldots, M$ **do**
4: $a^{(i)}_{t:t+d-1,l} \sim q_l(\cdot \mid a_{t:t+d-1,l-1}, \theta)$
5: **for** $j = 1, \ldots, \eta_l$ **do**
6: $b^{(i)}_{t,j} \leftarrow b$
7: $s^{(i)}_{t,j} \sim b^{(i)}_{t,j}$
8: **for** $k = t, \ldots, t + d - 1$ **do**
9: $(s^{(i)}_{k+1,j}, z^{(i)}_{k+1,j}, r^{(i)}_{k,j}) \sim \mathcal{G}(s^{(i)}_{k,j}, b^{(i)}_{k,j}, a^{(i)}_{k,j})$
10: $b^{(i)}_{k+1,j} \leftarrow \tau(b^{(i)}_{k,j}, a^{(i)}_{k,j}, z^{(i)}_{k+1,j})$
11: **end for**
12: **end for**
13: $w^{(i)}_l = w^{(i)}_{l-1} \prod_{j=1}^{\eta_l} \sum_{k=t}^{t+d-1} \gamma^{(k-t)} r^{(i)}_{k,j}$
14: **end for**
15: Normalise weights: $w^{(i)}_l = \frac{w^{(i)}_l}{\sum_{i=1}^{M} w^{(i)}_l}$
16: **if** resampling required **then**
17: Resample particles and set equal weights: $w^{(i)}_l = 1/M \quad \forall i$
18: **end if**
19: **end for**
20: **end function**

The expansion proceeds in three phases: sampling new particles, weighting particles, and resampling (if necessary).

The algorithm iterates over $l = 1, \ldots, l_{\max}$ (Line 2), processing each particle i separately (Line 3).

Sampling new particles is started by updating the action sequence related to the particle. This is achieved by sampling from the current kernel q_l (Line 4. The sampling kernel may either sample from a prior distribution over action sequences (e.g. at $l = 1$), or take as input some additional parameters,

[7] We include the belief state b as input to the generative model to be able to generate r from a general belief-dependent reward function $\rho(b, s, a)$ not necessarily linear in b.

denoted here by θ, to guide the sampling process. These parameters may include e.g. the reward information from past simulations. In general, the choice of q_l depends on the problem specifics, the crucial point being that the initial sample of action sequences at $l = 1$ should have non-zero probability at the maximisers of J_d. For further discussion on kernel selection in SMC algorithms, we direct the reader to Del Moral et al. (2006).

Once the action sequence is sampled, a simulation procedure is repeated for the particle η_l times on Lines 5-12. The simulation procedure takes the action sequence related to the particle and samples applying the generative model a sequence of states, observations, and rewards (Line 9). The corresponding belief states are also computed (Line 10).

Weighting of the particles is done on Line 13 according to the reward information obtained during the simulation phases. This is the step that effectively weights the particles according to the η'th power of J_d as in Eq. (2.19).

Resampling is performed if the effective sample size (Liu and Chen, 1998) $N_{\text{eff}} = 1/\sum_{i=1}^{M} \left(w_l^{(i)}\right)^2$ falls below a threshold value (Lines 16-18).

The Expand subroutine is executed once per planning phase, and the function UpdateAncestors does not perform any operation.

For extracting the best action on Line 10 of the generic algorithm, the best particle found by the algorithm is selected by $i_m = \underset{i=1,\ldots,M}{\operatorname{argmax}}\, w_l^{(i)}$. Then, the best action to execute is the first action of the sequence related to particle i_m, $a_{t,l}^{(i_m)}$.

The method is flexible considering it can handle general state, action, and observation spaces and reward functions. The main disadvantages include the lack of general rules for selecting the sampling kernels q_l and the usual drawback of sampling-based algorithms, namely that worst-case performance guarantees cannot be given.

2.2.4.3 Branch-and-bound pruning

This subsection considers online methods based on branch-and-bound pruning of the search tree. Branch-and-bounding means the process of removing from the tree branches that are known to be sub-optimal. Removing sub-optimal branches prevents expanding the search tree unnecessarily. Sub-optimality can be detected by applying lower and upper bounds for the optimal value function.

In an online algorithm applying branch-and-bound pruning, a lower and upper bound $L_T(b, a)$ and $U_T(b, a)$ are maintained for each belief state node b and related possible action a in the search tree T. The bounds are derived

from $L(b)$ and $U(b)$ via the equations (Ross et al., 2008)

$$L_T(b) = \begin{cases} L(b) & \text{if } b \text{ is a leaf node of } T \\ \max\limits_{a \in \mathcal{A}} L_T(b, a) & \text{otherwise} \end{cases} \tag{2.20}$$

$$L_T(b, a) = \rho(b, a) + \gamma \mathbb{E}\left[L_T(\tau(b, a, z))\right] \tag{2.21}$$

$$U_T(b) = \begin{cases} U(b) & \text{if } b \text{ is a leaf node of } T \\ \max\limits_{a \in \mathcal{A}} U_T(b, a) & \text{otherwise} \end{cases} \tag{2.22}$$

$$U_T(b, a) = \rho(b, a) + \gamma \mathbb{E}\left[U_T(\tau(b, a, z))\right], \tag{2.23}$$

taking all expectations under the prior probability of observations (2.6). If for a fixed $a_i \in \mathcal{A}$ it holds that $U_T(b, a_i) < L_T(b, a_j)$ for all $a_j \neq a_i$, the branch of the search tree corresponding to the action a_i can be pruned from the search tree as it is known to be sub-optimal.

The real-time belief space search (RTBSS) algorithm (Paquet et al., 2006) is an example of an online POMDP algorithm employing branch-and-bounding. In terms of Algorithm 2.1, the function ChooseNextNodeToExpand always returns the current belief state b. The function Expand expands the search tree up to the search depth d under b. The procedure is outlined in Algorithm 2.3. RTBSS sorts actions greedily according to their upper bound to maximise possibilities to prune sub-optimal branches from the search tree (Line 5). The possible actions are processed in this order until no actions remain or all the rest of the actions may be pruned (Line 8).

Algorithm 2.3 The expansion subroutine of RTBSS.

Input: The current belief state b, the search depth d.

```
1:  function Expand(b, d)
2:      if d = 0 then
3:          L_T(b) ← L(b)
4:      else
5:          Sort actions {a_1, ..., a_|A|} so that U_T(b, a_i) ⩾ U_T(b, a_j) if i ⩽ j
6:          i ← 1
7:          L_T(b) ← −∞
8:          while i ⩽ |A| and U_T(b, a_i) > L_T(b) do
9:              L_T(b) ← ρ(b, a_i) + γ Σ_{z'∈Z} p(z' | b, a_i) Expand(τ(b, a_i, z'), d − 1)
10:             L_T(b) ← max{L_T(b), L_T(b, a_i)}
11:         end while
12:     end if
13:     return L_T(b)
14: end function
```

The lower bounds are recursively improved (lines 9-10) by implementing equations (2.20)-(2.21). The function finally returns the best lower bound found for the current belief state (Line 13).

The function UpdateAncestors of Algorithm 2.1 does not need to perform any operations, as the current belief state has no ancestors in T. After a single iteration of the planning loop, all reachable belief states up to the search depth d have been explored, and the planning phase terminates.

As actions need to be sorted, RTBSS is applicable to finite action spaces only. For any branch-and-bound algorithm to be effective, the savings by pruning the search tree must be greater than the additional cost of computing and maintaining the bounds. Ross et al. (2008) also note that RTBSS may not be a good choice for problems where there is a large number of possible observations. As RTBSS explores all observations equally, it will also explore parts of the tree that have a small probability of occurring and thus have a small effect on the value function. A large number of observations thus implies that RTBSS may be limited to exploring over short search depths d. We note that RTBSS is suitable for non-linear belief-dependent rewards if the lower and upper bounds can be computed for such rewards as well.

2.2.4.4 Heuristic search

Heuristic search methods prioritise the order in which nodes are expanded in an online search. In practice, besides the lower and upper bounds, values for a heuristic function are maintained for the leaf nodes in the search tree T. In Algorithm 2.1, the function ChooseNextNodeToExpand selects the leaf node that has the greatest heuristic value. Typically, heuristic search methods also employ branch-and-bound pruning to avoid expanding sub-optimal branches from the search tree. The computational overhead compared to pure branch-and-bounding is somewhat greater since the heuristic function needs to be maintained as well. However, practical experiments have shown that the heuristic-guided search is still often more effective in finding optimal actions (Ross et al., 2008).

Satia and Lave (1973) proposed a heuristic that attempts to guide the search toward nodes b which are likely to reach, i.e. the action-observation sequence required to reach them has a high total probability, and where the upper bound $U_T(b) - L_T(b)$ on the error $V^{\pi^*}(b) - L_T(b)$ of the value function is large. Washington (1997) introduced a heuristic that prefers exploring branches corresponding so-called promising actions, which have a large upper bound $U_T(b, a)$, especially when the successor belief states have loose bounds on their value, i.e. $U_T(b) - L_T(b)$ is large. Unlike Satia and Lave, Washington's heuristic does not take into account the effect of the discount factor nor does it consider the likelihood of reaching a certain belief state, i.e. the probability of observations. Later, Ross and Chaib-draa (2007) proposed a heuristic function that combined the strengths of both of the above mentioned heuristics.

For problems with a non-linear belief-dependent reward, heuristic search is usually applicable as the heuristic values are maintained separately for each node in the search tree. Of course, if lower and upper bounds are additionally used, they must be applicable for such rewards as well.

2.2.4.5 Monte Carlo methods

A third major class of online POMDP algorithms takes advantage of sampling-based Monte Carlo (MC) methods to evaluate the values of the search tree nodes. Branch-and-bound pruning is ruled out as correctness of the bounds cannot be guaranteed via MC evaluation (Ross et al., 2008). A typical requirement for MC planning is that a generative model for the POMDP

is available. A generative model denoted $(s', z', r) \sim \mathcal{G}(s, b, a)$ produces a sample of the immediate reward r, the next state s' and observation z', given the current state s and action a. The generative model is applied to evaluate the relative value of possible policies.

McAllester and Singh (1999) implemented a MC algorithm that samples a fixed number of observations in the EXPAND subroutine of the generic Algorithm 2.1, expanding the search tree T up to a fixed depth. After the execution phase, the search tree T is reset to contain only the current belief at the root.

The rollout method proposed by Bertsekas and Castañon (1999) has been employed in a variety of MC algorithms for POMDPs. The method starts with an initial policy, and estimates the utility of each action assuming that after executing the action the initial policy is followed. The process is repeated multiple times, sampling possible future observations. These repetitions are called rollouts with the initial policy, hence giving the method its name. Bertsekas and Castañon (1999) showed that the rollout method will always at least maintain the quality of the initial policy, and may improve it. However, the basic rollout algorithm can only change the initial action in the plan, and may fail to improve in cases where for instance a long sequence of actions are required to improve over the initial policy. The parallel rollout algorithm proposed by Chang et al. (2004) alleviates this issue by considering more than a single initial policy, maximising over them. The rollout methods' EXPAND subroutine only considers different actions at the root of the search tree, while relying on the initial policies on subsequent steps.

In a similar vein, the PEGASUS solver of Ng and Jordan (2000) evaluates the value of a policy by executing the policy for a fixed number of time steps starting from a sampled initial state. Although the emphasis is on evaluating the usefulness of a set of policies offline before execution, conceptually the method is similar to the online algorithms presented here. In sufficiently small problems, all policies were exhaustively evaluated. For larger problems, hand picking policies to evaluate was suggested. Notably, PEGASUS can directly handle continuous action spaces as well. Likewise, there is no barrier to applying a reward non-linear in the belief state, although Ng and Jordan did not explore this.

The concept of repeated rollouts has been combined with an iterative approach to constructing and maintaining a search tree in so-called Monte Carlo tree search (MCTS) algorithms. MCTS is applicable to a wide range of search and game-playing problems, and has been successfully applied in domains such as computer Go or real-time games. We refer the reader to Browne et al. (2012) for a survey of principles and variants of MCTS with a discussion on applications.

The initial policy in the context of MCTS is typically called a rollout policy. Compared to the rollout algorithms presented above, the idea in MCTS is to iteratively deepen the search tree and apply rollouts starting from the fringe nodes of the tree. Such an approach allows improvement over the rollout policy even if multiple actions are required.

The basic iterative process that underlies MCTS illustrated in Figure 2.2.

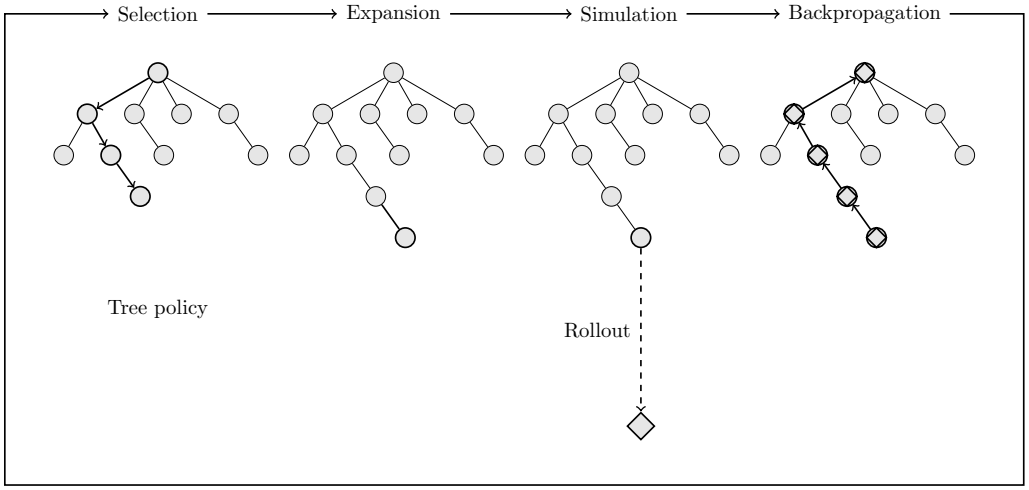


Figure 2.2: The basic scheme of Monte Carlo tree search (adapted from Browne et al. (2012)).

The algorithm repeatedly executes four phases shown from left to right: the selection, expansion, simulation and backpropagation phase. At each iteration the search tree is updated using the information gained by simulating the problem. A generative model is applied to carry out the simulations. In the selection phase, the information in the current search tree is applied to find a promising path to a leaf node that can be expanded, shown by a thicker line with arrowheads. Promising paths are those that seem to yield a high reward according to current information. Once an expandable leaf node is reached, the expansion phase is entered. In the expansion phase at least one child node is added to the tree, with the possible choices given by the action space $\mathcal{A}$. The simulation phase is then entered, where a rollout is executed starting from the state in the previously added child node. The actions to execute in the simulation phase are determined by the rollout policy. The generative model is applied to sample next states and rewards. The simulation procedure is continued e.g. until a desired depth has been reached, and the discounted sum of rewards related to the simulation is recorded. Finally, in the backpropagation phase the reward is applied to revise the information in the search tree along the path traversed in the selection phase.

Algorithm 2.4 The expansion subroutine of POMCP.

Input: The current belief state b, the search depth d under b.

1: **function** EXPAND(b, d)
2: $s \sim b$
3: SIMULATE(s, b, $\varnothing$, d)
4: **end function**

To date the most successful implementation of MCTS for POMDPs is the partially observable Monte Carlo planning (POMCP) algorithm of Silver and Veness (2010). POMCP is a variant of the popular upper confidence bounds for trees (UCT) algorithm of Kocsis and Szepesvári (2006). As seen in Section 2.1.2, a history may be used interchangeably with the corresponding belief state. POMCP represents the search tree T as a set of nodes that

are indexed by possible action-observation histories h in the POMDP. For a cleaner presentation, we use a shorthand notation to represent sequences of histories after the initial belief state. An empty set symbol refers to the initial belief state itself, and notation such as ha denotes a sequence (h, a) consisting of a history h of actions and observations followed by an action a. These sequences are then used to index nodes in the search tree, for instance the root node corresponding to the current belief state is referred to as $T(\varnothing)$. For each node $T(h)$ in the search tree, an estimate $V(h)$ of the discounted sum of rewards attainable from the node and a count $N(h)$ of the number of times the node has been visited are maintained.

In the context of the generic online algorithm (Algorithm 2.1), the EXPAND-subroutine of POMCP is simple, as shown in Algorithm 2.4. Each call to the EXPAND-function corresponds to a single iteration of the general MCTS procedure of Figure 2.2. The expansion starts by drawing a state sample from the current belief state (Line 2). After that, the function SIMULATE is called, initiating the selection phase (Line 3).

The function SIMULATE shown in Algorithm 2.5 takes as input a state sample s, a belief state b, a history h of actions and observations, and the remaining search depth d. The function returns a sample of the sum of discounted rewards attainable starting from state s and belief state b, while d decisions are executed. If the maximum search depth has been reached, the function returns the value zero.

If the history h does not correspond to a leaf node in the current search tree T, the algorithm initiates the selection phase. On Line 10, the UCT method is applied to select the most promising action $\hat{a}$ in the current belief state. The UCT method selects the action by maximising the sum of the current value estimate $V(ha)$ of the action and an exploration bonus that encourages trying actions that have not been tried yet. The exploration bonus is equal to $e \cdot \sqrt{\log N(h)/N(ha)}$, where e is an exploration coefficient, typically set to a value on the same range as typical sums of discounted reward in the problem (Silver and Veness, 2010). However, if there exists any a such that $N(ha) = 0$, the action is sampled uniformly at random from the set $\{a \in \mathcal{A} \mid N(ha) = 0\}$ of actions with count zero. Implementing the selection stage in this manner will ensure that all actions are first tried once, and then as the ratio $\log N(h)/N(ha)$ increases, actions other than the one with the currently highest value estimates will be selected again. Further details and discussion on the UCT method are given in Auer et al. (2002); Kocsis and Szepesvári (2006).

The generative model is applied to sample the next state s', observation z' and immediate reward r after executing action $\hat{a}$ (Line 11). A recursive call to the function SIMULATE is made on Line 12, giving as input arguments the next state s', the next belief state computed via $\tau(b, \hat{a}, z')$, the updated history sequence $(h, \hat{a}, z')$, and decreasing the remaining expansion depth to $(d - 1)$. Finally, the backpropagation phase (lines 13-15) updates the visitation counts and value estimates of nodes.

If $T(h)$ is a leaf node, the possible children $T(ha)$, $a \in \mathcal{A}$, are inserted to the search tree in the expansion phase (lines 5-7). Typically, the values $V(ha)$ and $N(ha)$ are both initialised to zero, but if heuristic or domain knowledge is available this can be modelled by functions $V_i(ha)$ and $N_i(ha)$,

respectively. After expansion, a sample of the attainable reward obtained by calling the function ROLLOUT is returned.

Algorithm 2.5 The simulation phase of POMCP.

Input: State sample s, belief state b, history h, search depth d.

1: **function** SIMULATE(s, b, h, d)
2: **if** $d = 0$ **then**
3: $R \leftarrow 0$
4: **else if** ISLEAF(h) **then**
5: **for all** $a \in \mathcal{A}$ **do**
6: $T(ha) \leftarrow (N_i(ha), V_i(ha))$
7: **end for**
8: $R \leftarrow$ ROLLOUT(s, b, d)
9: **else**
10: $\hat{a} \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, V(ha) + e\sqrt{\frac{\log N(h)}{N(ha)}}$
11: $(s', z', r) \sim \mathcal{G}(s, b, \hat{a})$
12: $R \leftarrow r + \gamma \cdot$SIMULATE($s'$, $\tau(b, \hat{a}, z')$, $(h, \hat{a}, z')$, $d - 1$)
13: $N(h) \leftarrow N(h) + 1$
14: $N(h\hat{a}) \leftarrow N(h\hat{a}) + 1$
15: $V(h\hat{a}) \leftarrow V(h\hat{a}) + \frac{R - V(h\hat{a})}{N(h\hat{a})}$
16: **end if**
17: **return** R
18: **end function**

The rollout phase is implemented by Algorithm 2.6, and it returns a sample of the sum of discounted rewards over d decisions from a given state s and belief state b when following a given rollout policy. Actions are selected according to a rollout policy $\pi_{\text{rollout}} : \mathcal{B} \to \mathcal{A}$ (Line 5), and the generative model is applied to sample the next state, observation and immediate reward (Line 6). The rollout policy in the simplest case selects actions uniformly at random from $\mathcal{A}$, although other choices such as heuristically guided policies are possible as well (Browne et al., 2012). A recursive call to the function ROLLOUT is made on Line 7, with arguments revised as in the function SIMULATE above.

Algorithm 2.6 The rollout phase of POMCP.

Input: State sample s, belief state b, search depth d.

1: **function** ROLLOUT(s, b, d)
2: **if** $d = 0$ **then**
3: $R \leftarrow 0$
4: **else**
5: $a \leftarrow \pi_{\text{rollout}}(b)$
6: $(s', z', r) \sim \mathcal{G}(s, b, a)$
7: $R \leftarrow r + \gamma \cdot$ ROLLOUT(s', $\tau(b, a, z')$, $d - 1$)
8: **end if**
9: **return** R
10: **end function**

The POMCP algorithm has been shown to converge to the optimal horizon d value function as the number of EXPAND-function calls tends towards

infinity (Silver and Veness, 2010). As seen from the algorithm listings 2.5 and 2.6, applying non-linear belief dependent rewards does not constitute any additional difficulty. The tree expansion in POMCP is non-uniform, meaning that some branches may be explored to a great depth, while others are quickly considered non-preferable and not expanded further. Thus it has a potential advantage over uniformly-expanding algorithms in domains requiring long sequences of actions to reach a high reward. As usual, the general weaknesses of Monte Carlo algorithms apply: worst-case performance guarantees usually cannot be given, nor can branch-and-bound pruning be applied. Other approaches to POMDP planning employing Monte Carlo methods include the DESPOT algorithm of Somani et al. (2013), or the particle-based monotonic value improvement technique of Pajarinen and Kyrki (2015).

2.2.5 Other approaches

The approaches to finding exactly or approximately optimal policies for POMDPs presented in the previous subsections by no means give an exhaustive treatment on the topic. For instance, Monahan (1982); Lovejoy (1991a) survey exact solution methods. More comprehensive surveys of approximate value function (Hauskrecht, 2000), point-based (Shani et al., 2013) and online methods (Ross et al., 2008) are also avaiable. We next present a brief overview of POMDP solution methods that do not fit under the topics covered in the preceding subsections.

2.2.5.1 Special cases of POMDPs

Other special cases of POMDPs can be distinguished e.g. by the properties of the state transition function.

In his thesis, (Littman, 1996, Sect. 6.3.2) studied the subclass of deterministic POMDPs. A deterministic POMDP is characterised by the fact that both state transitions and observations are deterministic, as seen in the following definition.

Definition 2.8 (Deterministic POMDP)**.** *A POMDP* $\langle \mathcal{T}, \mathcal{S}, \mathcal{A}_b, \mathcal{Z}, \mathbb{T}, \mathbb{O}, R \rangle$ *is deterministic if for all* $s, s' \in \mathcal{S}$*,* $a \in \mathcal{A}$*,* $z \in \mathcal{Z}$

$$\mathbb{T}(s', a, s) = \begin{cases} 1 & \text{if } s' = f(s, a) \\ 0 & \text{otherwise} \end{cases}, \tag{2.24}$$

where $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ *is a deterministic state transition function, and*

$$\mathbb{O}(z', s', a) = \begin{cases} 1 & \text{if } z' = h(s', a) \\ 0 & \text{otherwise} \end{cases}, \tag{2.25}$$

where $h : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$ *is a deterministic observation function.*

Littman determined that as there is a single successor state to any action, and each state-action pair emits a single observation, the reachable belief states in a finite horizon problem can be enumerated. Hence, a conversion to a finite MDP over the set of reachable belief states exists, and e.g. value or policy

iteration may be applied to solve the deterministic POMDP. Littman showed that the finite horizon deterministic POMDP problem is NP-complete.

Bonet (2009) notes a relation between policies and so-called AND/OR graphs, allowing for potentially more effective solutions in practice than via the MDP mapping suggested by Littman.

Besse and Chaib-draa (2009) considered quasi-deterministic POMDPs, where state transitions are deterministic, but with the constraint that for any state and action the probability of perceiving one of the observations is lower bounded by at least one half. They show that these quasi-deterministic problems are easier than general problems by bounding the length of the history sequence needed to identify almost surely the underlying state.

Warnquist et al. (2013) studied the case where *some* actions have deterministic effects. The deterministic actions are abstracted into macro actions, which improves practical convergence speed of a solver.

In some cases, POMDP problems have mixed observability. This means that some components of the state space in a factored representation may be fully observable. Such cases are particularly encountered e.g. in robotics domains, for which specialised POMDP solvers exist (Ong et al., 2010) that leverage the lower-dimensional representation of the belief space together with a point-based solver to compute approximate solutions. Computational savings are achieved as it suffices to maintain a collection of sets of low-dimensional α-vectors, one for each member variable of the fully observable part of the state space, to represent the value function.

2.2.5.2 Submodularity

A *submodular function* (Fujishige, 2005) is a set function which satisfies a "diminishing returns" property.

Definition 2.9 (Submodular function)**.** *Let E denote a nonempty finite set, and 2^E the power set of E. A function $f : 2^E \to \mathbb{R}$ is submodular if for any two subsets $X, Y \subset E$ such that $X \subseteq Y$ and any $x \in E \backslash Y$*

$$f(X \cup \{x\}) - f(X) \geqslant f(Y \cup \{x\}) - f(Y). \tag{2.26}$$

Submodularity indicates that adding an element x to a smaller set X is increases the value of f more than adding the same element to the larger set Y. Additionally, a submodular function is monotone if $f(X \cup \{x\}) \geqslant f(X)$.

Krause et al. (2008) considered monotone submodular objective functions in a sensor placement problem. The problem was one of optimising prediction quality in a Gaussian process by selecting sampling locations. Concretely, a finite number k of sensors was to be placed at a set of locations X of interest. The sensor locations were chosen from the set $E \supset X$ of all possible locations, and the objective was to maximise a submodular function quantifying the informativeness of sensor locations and prediction quality over the whole of E. According to Nemhauser et al. (1978), in this case a greedy algorithm that sequentially adds elements to X such that they maximise the expected immediate improvement of the objective function finds a solution that is at most a factor $(1 - 1/\exp(1))$ lower than the optimal solution.

Krause et al. (2008) considered a stationary case where the process was fixed and merely a set of sampling locations were selected to maximise a performance measure. As such, the approach is not directly suited to a case where contingencies arise: for example, as a result of deploying a sensor, feedback is received indicating whether the deployment succeeded or not and further actions are taken contingent on this information. POMDPs are examples of contingent planning as well; a different course of action may be taken at a subsequent decision epoch conditional on the observations perceived.

Golovin and Krause (2011) extend the idea to contingent planning by introducing the concept of adaptive submodularity. The extension introduces a realisation function $\phi : E \to O$ mapping possible sensor locations to a state O. The problems considered proceed sequentially: a location $e \in E$ is selected, its state $\phi(e)$ is (perfectly) observed, the next location is selected, its state is observed, and so on. The state may e.g. correspond to the event of a failure or success in sensor deployment. A partial realisation ψ is a function from a subset of E to their states. Thus, ψ can be a description of locations selected thus far and whether the deployment in each location failed or not. The policies in Golovin and Krause's approach are mappings from a set of partial realisations to E; specifying which location to select next given a particular history of deployments and their successes or failures. As such, the partial realisations are equivalent to belief states of a POMDP. Adaptive submodular monotone objective functions are considered, defining submodularity now via the *expected* marginal improvement of selecting an item given the current partial realisation (belief state). A constant factor approximation to the optimal solution is shown to be found by a greedy policy.

Golovin and Krause's decision-making model is equivalent to the deterministic POMDP (Definition 2.8). If k locations are to be selected, the problem is finite horizon with k decisions. Let $c = \{c_e\}_{e \in E}$ indicate for each location $e \in E$ whether it has been selected ($c_e = 1$) or not ($c_e = 0$). Furthermore, let $d = \{d_e\}_{e \in E}$ where $d_e \in O$ denote the (a priori unknown) state of location $e \in E$. The state is thus described by a pair $s = (c, d) \in \mathcal{S}$. The set of applicable actions is recoverable from the belief state since c_e are fully observable; the applicable actions $\mathcal{A}_b$ are simply equal to the set of hitherto unselected locations. The observation space is $\mathcal{Z} = O$. The conditions of Definition 2.8 are now fulfilled by making the following two definitions. Define $f(s, a) = (c', d)$, where $c'_a = 1$ and $c'_i = c_i$ for all $i \neq a$ and d remains unchanged. Finally, define $h(s', a) = d_a$.

The key insight to seeing the equivalence is to note that in the sensor selection problem the fact whether a sensor deployment at a particular location will succeed or fail remains fixed; it is only our information regarding the relative probability of these two events that may vary as a consequence of deploying sensors at other locations. Even the information regarding failure probabilities remains fixed if the failures are independent as suggested by Golovin and Krause. As the event of failure or success is perfectly observed, the conditions for a deterministic observation model are fulfilled.

Both Krause et al. (2008) and Golovin and Krause (2011) considered submodular reward functions that are non-linear in the belief state. The

conditions under which these reward functions are (adaptive) submodular and monotone are somewhat restrictive; for instance a general stochastic state transition model or a non-perfect observation model cannot be applied. As the problems considered fit into the general framework of POMDPs, they nevertheless expand the class of POMDPs that can be efficiently approximated. In recent work by Satsangi et al. (2015), submodularity of value functions is shown for reward function equal to the negative belief entropy, while extending the aforementioned results to the full POMDP setting that allows non-deterministic state transitions.

2.2.6 General POMDPs

A feature common to most of the methods reviewed above was that the state, action and observation spaces were assumed to be finite. As continuous spaces are realistic models for real-world problems, such POMDP representations have also been studied. We already briefly covered some algorithms able to deal with continuous spaces, for instance the SMC-OLFC algorithm presented in Subsection 2.2.4.2. This subsection gives a more detailed overview of the work on general POMDPs.

Arguably the most famous special case of a POMDP is the linear-quadratic-Gaussian (LQG) control problem. In the LQG case, the state space is a n-dimensional real space $\mathbb{R}^n$, while the action space is m-dimensional, $\mathbb{R}^m$. The state transition model and observation model are linear with additive Gaussian noise, and the reward (cost) function is a quadratic function of the state and action. If the initial belief state is Gaussian over $\mathbb{R}^n$, all subsequent beliefs are Gaussian as well. In this case, it is well known the optimal control policy may solved in closed form, see e.g. Åström (1970); Athans (1971).

In LQG control, the principle of separation of estimation and control holds. An optimal state estimator, in this case a Kalman filter, can be designed separately for the system. The optimal control is a deterministic function of the state estimate from the optimal estimator.

Meier et al. (1967) studied the case when one additionally has a choice between a finite number of measurement channels in a LQG control problem, i.e. a sensor management problem is considered alongside the control problem. They showed that the separation principle allows one to solve separately the control problem and the problem of sequentially selecting the most useful measurement channels. The optimal policy for operating the sensing subsystem was found independent of the actual measurement data obtained: such a result does not hold for non-linear cases.

The optimal control in the LQG case may be stated as a function of a parametric representation of the belief state. A similar idea has been applied to POMDPs where the state transition and observation models are non-linear or where the noise is non-Gaussian. In general, the belief states can be arbitrary PDFs over the state space. Some approximate methods constrain belief states to some parametric family of PDFs represented by a finite-dimensional vector of sufficient statistics. For instance, Brooks et al. (2006) apply a Gaussian parametrisation of the belief state. They then apply value iteration over the finite-dimensional parameter space. A weakness of the approach is that a single-Gaussian representation is not sufficient

to represent multi-modal beliefs. An extension projecting belief states to a family of parametrised PDFs instead of just Gaussians was introduced by Zhou et al. (2010). Particle filtering was applied to track belief states in the projected parameter space. Platt et al. (2010) apply LQG control in the belief space while assuming maximum likelihood observations to simplify the belief state dynamics.

Porta et al. (2006) generalise the notion of α-vectors to continuous state spaces via *α-functions*. They show that for expected rewards linear in the belief state the value function is convex, and for discrete action and observation sets it is PWLC. They tackle the problem of belief state representation by using Gaussian mixture models (GMMs) to represent both the beliefs and the state transition and observation models. They apply point-based value iteration over this parametric representation of beliefs to solve the POMDP, representing the value function as the supremum of a set of α-functions. However, the number of components in the GMMs representing posterior belief states and value functions increases exponentially, and the number of components is reduced by approximating the GMMs with a mixture with fewer components.

An alternative way to handle continuous spaces is via discretisation. The discretised problem can then be handled by any solver suitable for a finite POMDP. However, as noted e.g. by Brooks et al. (2006), finite POMDP methods have problems when discretisations are fine, as the dimensionality of the belief space increases with the number of states leading to rapidly increasing computational demands.

Sampling-based methods have also been widely applied for general POMDPs. Thrun (2000) represents a belief state over a continuous state space by a particle approximation, and applies a function approximation based on k-nearest-neighbours to evaluate the value function. A policy search method was applied by Martinez-Cantin et al. (2009) in a continuous-state, continuous-action POMDP. Monte Carlo sampling was applied to evaluate the quality of policies, iteratively improving the policies through Bayesian optimisation. Likewise, the PEGASUS solver (Ng and Jordan, 2000) applied sampled simulation trajectories to evaluate policies, and can handle continuous action spaces. Kantas et al. (2009) present a SMC method combined with the receding horizon control principle that is able to handle continuous state, action and observation spaces.

Hoey and Poupart (2005) studied continuous observation spaces, and noted that from the decision maker's point of view only observations that ultimately lead to a different action must be distinguished. Regions of the observation space leading to the same optimal action are aggregated into a single meta-observation, ultimately leading to a finite representation of the observation space.

Some online planning methods presented in Subsection 2.2.4 can be applied to POMDPs with a continuous state space and finite action and observation space. Specifically, the online methods that build a search tree over the reachable belief states (see Figure 2.1) require finite action and observation spaces for the tree to have a finite branching factor. If beliefs over the continuous state space can be propagated via the belief update equation (Equation (2.4)), no additional difficulties arise from the uncountable state

space. Furthermore, sampling-based approaches remain applicable in such cases.

2.2.7 Summary of POMDP solution methods

An overview of the key properties of the algorithms surveyed in this section is presented in Table 2.2. The algorithms are grouped by type in the leftmost column, and by the specific algorithm in the second column. Algorithms with similar properties are shown on a single line to save space.

The table summarises three properties of the algorithms. In the third column, algorithms that are suitable for POMDP problems with a non-linear belief-dependent reward function are marked by the symbol "+", and unsuitable with a symbol "-". Possible additional conditions are indicated and explained by footnotes in the table. The fourth column determines whether the algorithm can handle uncountable state, action and/or observation spaces, labelled by the respective mathematical symbol for the space in the affirmative case, and with a symbol "-" if the algorithm is limited to finite state, action and observation spaces. The fifth column indicates with a symbol "+" if a bounded error compared to the optimal solution can be guaranteed for the solution provided by the algorithm, and with a symbol "-" if not. The sixth column provides additional notes about the applicability of the algorithm where pertinent.

All of the online algorithms listed in the table can handle non-linear belief-dependent rewards. In addition, many of them are suitable for POMDPs with uncountable spaces.

A drawback of the two most flexible online algorithms, MCTS and SMC-OLFC, is that they cannot give a guarantee as to the quality of the solution they provide – it has merely been shown that they asymptotically converge to the optimal solution as the number of Monte Carlo simulations tends toward infinity. If more than one algorithm can be applied to solve a problem, the ability to guarantee an error bound may be the deciding factor in preferring one method over another. For instance, safety-critical applications cannot be operated under a policy of unbounded worst-case performance, as featured in the Monte Carlo methods.

Table 2.2: Summary of algorithms for finding policies for POMDPs.

Type (Section)	Algorithm	Non-linear R	Uncountable $\mathcal{S}, \mathcal{A}, \mathcal{Z}$	Bounded error	Notes
Exact (2.2.1)	LP, Policy iteration	-	-	+	
	ρPOMDP	+[8]	-	+	Can use with other methods
Point-based (2.2.2)	Grid	+	-	+[9]	
	PBVI	-	-	+	
	Perseus	-	$\mathcal{S}, \mathcal{A}, \mathcal{Z}$[10]	+	
	SARSOP, HSVI	-	-	+	
Compression (2.2.3)	E-PCA, VDC	-	-	+[11]	
Online (2.2.4)	RTBSS, Heuristic search	+[12]	$\mathcal{S}$	+	
	Monte Carlo Tree Search	+	$\mathcal{S}$	-	
	SMC-OLFC	+	$\mathcal{S}, \mathcal{A}, \mathcal{Z}$	-	
	Adaptive submodularity	+[13]	-	+[14]	Deterministic POMDPs

[8] Convex R.
[9] Lovejoy (1991b) bounded the error for linear R.
[10] Continuous variant of Perseus supports POMDP parameters defined as GMMs, see subsection 2.2.6 and (Porta et al., 2006).
[11] For lossless belief compression.
[12] Provided neither the bounds nor the heuristics rely on the α-vector representation of value functions.
[13] Submodular R.
[14] Greedy policy has bounded error.

2.3 Markovian multi-armed bandits

In this section, we give the definition of a subclass of stochastic decision making problems known as bandit problems, or multi-armed bandits (MABs). The MAB problem has been applied as a model for sensor management problems, for instance in target tracking domains (Washburn, 2008). The key property of a MAB is that it admits an optimal solution by a priority index rule that can be computed by forward induction. Such forward induction solutions are usually easier to find than the backward value iteration procedure required for general POMDPs (Mahajan and Teneketzis, 2008). In this section, necessary background information to facilitate further discussion on MAB models for sensor management problems is presented.

Informally, in a MAB the decision-maker is faced with a choice of playing a single arm of a set of n one-armed bandit slot machines. More generally, selecting an arm may represent e.g. allocation of resources to a particular project. When an arm is played a reward is accumulated, and the arm that was played transitions to a new state. The decision-maker is concerned with sequential selection of arms to maximise the discounted sum of the collected rewards.

The classical MAB problem is distinguished among general stochastic decision problems by four properties as follows (see Bertsekas, 2001, Sect. 1.5 and Mahajan and Teneketzis, 2008):

Property 1:	exactly one machine is operated at each decision epoch, and the evolution of the state of that machine is uncontrolled; the decision-maker chooses which machine to operate but not how to operate it,
Property 2:	machines that are not operated remain in their current state,
Property 3:	machine states are independent, and
Property 4:	machines that are not operated contribute no reward.

We consider the special case where the states of the machines evolve satisfying the Markov assumption (Definition 2.2). In this case, a MAB is a special case of a MDP (Definition 2.4). For a general treatment of bandit problems, we refer the reader to Berry and Fristedt (1985); Gittins et al. (2011).

The following definition based on Puterman (1994, Sect. 3.6) satisfies the properties listed above.

Definition 2.10 (Markovian multi-armed bandit)**.** *A Markovian multi-armed bandit with n machines is a MDP $\langle \mathcal{T}, \mathcal{S}, \{\mathcal{A}_s\}, \mathbb{T}, R \rangle$, where $\mathcal{T} = \{0, 1, \ldots\}$ is the uncountable set of decision epochs, the state space is the Cartesian product of the state spaces of the individual machines, $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$, the applicable actions are $\mathcal{A}_s = \mathcal{A} = \{1, 2, \ldots, n\}$ $\forall s \in \mathcal{S}$, the state transition model is factorised into independent transition models $\mathbb{T}_i : \mathcal{S}_i \times \mathcal{A} \times \mathcal{S}_i \to \mathbb{R}^+$, $1 \leqslant i \leqslant n$, such that*

$$\mathbb{T}(s', a, s) = \prod_{i=1}^{n} \mathbb{T}_i(s'_i, a, s_i), \tag{2.27}$$

and furthermore each transition model is uncontrollable, i.e.

$$\mathbb{T}_i(s_i', a, s_i) = \begin{cases} p_i(s_i' \mid s_i) & \text{if } a = i \\ \mathbb{1}_{\mathcal{S}_i}(s_i', s_i) & \text{if } a \neq i \end{cases}, \tag{2.28}$$

where $p_i(s_i' \mid s_i)$ denotes the state transition probabilities of a Markov process on $\mathcal{S}_i$ (Definition 2.2), $\mathbb{1}_{\mathcal{S}_i} : \mathcal{S}_i \times \mathcal{S}_i \to [0, 1]$ is an identity mapping such that for $s_i', s_i \in \mathcal{S}_i$ $\mathbb{1}_{\mathcal{S}_i}(s_i', s_i) = 1$ if $s_i' = s_i$ and 0 otherwise, and the reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ satisfies

$$R(s, a) = R(s_1, s_2, \ldots, s_n, a) = R_i(s_i) \quad \text{if } a = i, \tag{2.29}$$

where $R_i : \mathcal{S}_i \to \mathbb{R}$ is the reward function for the ith machine.

Compare the definition to the four properties of MABs listed above. The requirement of operating one machine in an uncontrolled manner (Property 1) is satisfied through the definition of the applicable action sets and (2.28). The arms that are not played are static (Property 2), as indicated by the second case of (2.28). The requirement for independence of the arms (Property 3) is satisfied through (2.27). Finally, machines that are not operated contribute no reward (Property 4), as indicated by (2.29).

2.3.1 Partially observable MABs

The Markovian MAB (Definition 2.10) is stated in terms of the fully observable MDP. If the states of the bandit arms are not directly observable, one can instead define a partially observable version of the problem, similarly to the case of MDPs and POMDPs. The states of the arms are replaced by belief states, and the state transition function is replaced by a probabilistic transition function between belief states, as in the belief MDP (Lemma 2.6).

The observation model in a partially observable MAB must satisfy

$$\mathbb{O}(z', s', a) = \mathbb{O}_a(z', s_a'), \tag{2.30}$$

where $s_a \in \mathcal{S}_a$, i.e. the observation following an action a is only conditional on the following state of the corresponding machine, s_a'. Consider now the belief update equation for $\mathbb{O}$ satisfying (2.30) and $\mathbb{T}$ and $\mathcal{S}$ satisfying the properties in Definition 2.10. Suppose the current information regarding the state of each machine is given by a vector $b = [b_1, \ldots, b_n]^{\mathrm{T}}$ of belief states, where each b_i is a belief over the state s_i of the ith machine, with sum equal to one.

Now for a fixed $a \in \mathcal{A}$, the belief update equation only affects the part b_a of the belief state. The state prediction is

$$p(s_a' \mid b_a, a) = \int_{\mathcal{S}_a} \mathbb{T}_a(s_a', a, s_a) b_a(s_a) \mathrm{d}s_a, \tag{2.31}$$

and the observation z' only depends on s_a;

$$p(z' \mid b_a, a) = \int_{\mathcal{S}_a} \mathbb{O}_a(z', s_a') p(s_a' \mid b_a, a) \mathrm{d}s_a', \tag{2.32}$$

and finally the belief update equation for the part b_a is

$$b'_a = \tau_a(b_a, a, z') = \frac{\mathbb{O}_a(z', s'_a)p(s'_a \mid b_a, a)}{p(z' \mid b_a, a)}. \tag{2.33}$$

Hence the complete belief update equation is

$$\tau(b, a, z') = [b_1, \ldots, b_{a-1}, \tau_a(b_a, a, z'), b_{a+1}, \ldots, b_n]^{\mathrm{T}}. \tag{2.34}$$

Thus when we start with a Markovian MAB $\langle \mathcal{T}, \mathcal{S}, \mathcal{A}_s, \mathbb{T}, R \rangle$, hide the state while adding an observation model $\mathbb{O}$ satisfying (2.30), the resulting bandit problem is referred to as a POMDP MAB. The following definition states the problem as a POMDP.

Definition 2.11 (POMDP multi-armed bandit)**.** *A POMDP multi-armed bandit with n machines is a POMDP* $\langle \mathcal{T}, \mathcal{S}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}, \mathbb{O}, R \rangle$*, where* $\mathcal{T} = \{0, 1, \ldots\}$*,* $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$*, and correspondingly* $\mathcal{B} = \mathcal{B}_1 \times \cdots \times \mathcal{B}_n$ *where* $\mathcal{B}_i = \mathcal{P}(\mathcal{S}_i)$*,* $\mathcal{A}_b = \mathcal{A} = \{1, 2, \ldots, n\} \forall b \in \mathcal{B}$*, the state transition model is factorised into independent transition models* $\mathbb{T}_i : \mathcal{S}_i \times \mathcal{A} \times \mathcal{S}_i \to \mathbb{R}^+$ *as in Definition 2.10, the observation model satisfies* (2.30)*, and the reward function* $R : \mathcal{B} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *satisfies*

$$R(b, s, a) = R(b_1, s_1, b_2, s_2, \ldots, b_n, s_n, a) = R_i(b_i, s_i) \quad \text{if } a = i, \tag{2.35}$$

where $R_i : \mathcal{B}_i \times \mathcal{S}_i \to \mathbb{R}$ *is the reward function for the* i*th machine.*

An equivalent version via the belief MDP (Lemma 2.6) is straightforward to derive.

2.3.2 Index policies for MABs

The optimality criterion for MABs is the infinite horizon discounted total reward (Bertsekas, 2001; Mahajan and Teneketzis, 2008). By the arguments presented in Subsection 2.1.3, the optimal policy in a Markovian MAB is a stationary policy mapping from states to arms to play. In Subsection 2.1.4, the optimal policy was found by a backward recursive method. This can guarantee that an optimal policy is found in a general stochastic decision problem. However, in a MAB an action that is currently available can be chosen at any later decision epoch attaining the same reward, excluding the effect of discounting: any arm not played currently can be played at a later stage.

The MAB can be decomposed into a set of n single-armed bandit problems, and the optimal policy is to play an arm with the greatest dynamic allocation index, known as the Gittins index (Gittins, 1979). This property is related to the irrevocability of actions in a MAB. Irrevocability means that an action currently available can be chosen at any later decision epoch attaining the same reward, excluding the effect of discounting. The infinite optimisation horizon is a necessary condition for the index rule property to hold (Gittins et al., 2011).

We next define the Gittins index in terms of the POMDP MAB, and note that the definition carries to the Markovian MAB by replacing belief states

with states. Let $b_{i,t}$ denote the belief over the state of arm i at decision epoch t. The Gittins index for arm i is given by (Gittins, 1979; Varaiya et al., 1985; Gittins et al., 2011)

$$v_i(b_{i,t}) = \max_{\tau > t} \frac{\mathbb{E}\left[\sum_{k=t}^{\tau-1} \gamma^k \rho_i(b_{i,k})\right]}{\mathbb{E}\left[\sum_{k=t}^{\tau-1} \gamma^k\right]}, \tag{2.36}$$

where $\rho_i(b_{i,k}) = \mathbb{E}\left[R_i(b_{i,k}, s_{i,k})\right]$, taking the expectation under $b_{i,k}$. The expectations in (2.36) are taken given the initial state $b_{i,t}$, and the maximisation is over the stopping time $\tau > t$. The index value may be interpreted as the maximum expected reward per unit of expected discounted time (Gittins, 1979). The optimal policy in a MAB is to play arm $\operatorname{argmax}_{i \in \mathcal{A}} v_i(b_{i,t})$ until the stopping time τ related to the arm, and repeat.

The Gittins indices for a MAB can be computed for each arm offline for all possible states, before starting the task execution. For algorithms for computing the Gittins indices for the fully observable Markovian MAB, see (Gittins et al., 2011).

The computation of the Gittins index for the POMDP MAB is somewhat more involved, due to the fact that the index must be computed over the uncountable belief space. The Gittins index of Equation (2.36) may alternatively be defined as (Krishnamurthy and Wahlberg, 2009)

$$v_i(b_{i,t}) = \min\left\{M \in \mathbb{R} : V^i(b_{i,t}, M) = M\right\}, \tag{2.37}$$

where $V^i(b_{i,t}, M)$ satisfies the functional Bellman recursion

$$V^i(b_{i,t}, M) = \max\left\{\rho_i(b_{i,t}) + \gamma \sum_{z' \in \mathcal{Z}} p(z' \mid b_{i,t}, i) V^i(\tau_i(b_{i,t}, i, z')), M\right\}. \tag{2.38}$$

Here, M is a retirement reward for which the decision maker is indifferent between continuing to play the arm or retiring immediately and obtaining a one-time reward of M.

For rewards linear in the belief state, Krishnamurthy and Evans (2001); Krishnamurthy and Wahlberg (2009) showed that an arbitrarily close approximation for the Gittins index in this case can be found by value iteration. Furthermore, leveraging the results of Smallwood and Sondik (1973) they showed that a finite representation of the approximate Gittins index can be given in a similarly to the α-vectors in a POMDP. The approximate Gittins indices were then solved by standard POMDP value iteration.

In the case the right hand side of (2.36) is maximised for $\tau = t + 1$, the Gittins index is equal to $\mathbb{E}\left[\rho_i(b_{i,t})\right]$, the expected immediate reward. Then the index policy is the same as a myopic policy, or greedy policy, that selects actions such that the expected immediate reward is maximised. Conditions when the myopic policy is optimal are identified e.g. in (Gittins et al., 2011, Prop. 2.5, Prop. 2.7) and (Bertsekas, 2001, Sect. 1.5). The requirement is that the Gittins index after selecting a bandit arm is almost surely lower than or equal to the Gittins index at the current decision epoch. The result is summarised in the following proposition due to (Bertsekas, 2001, Sect. 1.5).

Lemma 2.12 (Optimality of a myopic policy)**.** *If in a POMDP MAB for all* $t \in \mathcal{T}$*,* $i \in \{1, 2, \ldots, n\}$*,* $b_{i,t} \in \mathcal{B}_i$*,* $a_t = i$*,* $z_{t+1} \in \mathcal{Z}$ *the Gittins index satisfies*

$$v_i(b_{i,t}) \geqslant v_i(\tau_i(b_{i,t}, a_t, z_{t+1})), \tag{2.39}$$

then the optimal policy is to select arms to play that greedily maximise the expected reward, i.e. $\pi^* = \operatorname{argmax}_i \rho_i(b_{i,t})$.

Bertsekas (2001, Sect. 1.5) identifies this case as the deteriorating case, since arms become less profitable as they are selected.

3

Canonical sensor management problems in mobile robotics

In this chapter, four sensor management problems in mobile robotics domains are defined. These canonical problems are examined throughout the remainder of the thesis. In each problem, an agent has degrees of freedom in the operation of a sensing subsystem.

Necessary background on the information-theoretic concepts of entropy and mutual information is presented in Appendix A. Motivation for why information-theoretic reward functions are a useful and justified choice for sensor management problems is given in Section 3.1.

Section 3.2 gives a brief overview of real robot control applying non-myopic policies for POMDPs. Relevant applications and existing solution approaches are identified. An overview of the four canonical problems addressing similar applications is given in Section 3.3. Problems with mixed observability and deterministic robot motion are defined in Section 3.4, while problems with partial observability and stochastic robot motion are defined in Section 3.5. Further categorisation in both sections is given by the type of reward function, which is either state or belief state dependent.

3.1 Motivation for information-theoretic reward functions

The role of the reward function in a POMDP is to encode the task by defining the utility of states and actions. Typically, the reward function is dependent on the state and action, i.e. $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ (see e.g. Kaelbling et al. (1998)). Negative rewards can be interpreted as costs. This type of reward function has applications in control tasks. Examples include regulation of the state to a given set point, controlling the system in order to follow a given state trajectory, or encouraging reaching "good" states (high positive reward) and discouraging reaching adverse or dangerous states (high negative reward). Additionally, through the dependency on the action this formulation can capture the cost of different possible control actions. A classical example

of the use of such encoding of the task is the LQG control problem with a quadratic cost function (Athans, 1971).

Sensor management problems with partially observable state that can be solved with a reward linear in the belief state include e.g. adaptive classification (Chong et al., 2009) and active hypothesis testing (Atanasov et al., 2014b). The common feature in these problems is that the objective can be expressed via a state and action dependent reward function: choose a correct class from a given set of classes, or choose the correct hypothesis from a given set of hypotheses, respectively.

However, there are problems where the objective can not naturally be encoded in terms of a state and action dependent reward. Two examples are surveillance and exploration. Both have no pre-specified end condition or state. Additionally, we can make the reasonable assumption that we are essentially unaware of what to expect in either task, and do not know beforehand how to react to any new discoveries (Araya-López et al., 2010). The problems seem to be ill-suited for modelling with state-dependent rewards.

The separation of *gathering* and *exploitation* of information is useful from a practical viewpoint as well. An exploratory mission typically has a large or infinite amount of possible information outcomes. Even if the possible outcomes were known in advance, it is not realistic to plan a strategy for their exploitation in advance. If the possible outcomes cannot be described beforehand, the situation is even worse, equivalent to planning contingent on realisations of a random variable whose sample space is unknown.

In some cases it might be possible to circumvent issues such as those illustrated above by clever manipulations of e.g. the state and action spaces of the problem. For instance, Spaan et al. (2015) manipulate the action space of a factored-state POMDP to include a binary action for each state factor. The reward function is then modified accordingly to define a range of beliefs where the agent will execute either of the binary actions; allowing modelling information gathering problems. An advantage of the approach is that a variety of known results and approximate algorithms can be exploited by remaining in the standard POMDP framework. However, this has the drawback of increasing the size of the action space exponentially, possibly leading to computational difficulties. Careful design of the reward function w.r.t. the POMDP problem at hand is also required to ensure appropriate information gathering behaviour.

A natural choice often is to modify the reward function so that it is explicitly dependent on the belief state, and can hence model information gathering objectives directly. For example, information-theoretic quantities can be applied as reward functions in information gathering problems. These quantities can of course be combined with state-dependent rewards e.g. as a linear combination to model varying attractiveness of true underlying states or varying costs of actions.

Specifically, in the following sections we will apply the mutual information (MI) of the state S and observation Z on the following decision epoch conditional on the action a at the current decision epoch, i.e.

$$I(S; Z \mid a) = H(S \mid a) - \mathbb{E}_Z \left[H(S \mid Z = z, a) \right]. \tag{3.1}$$

The terms in the above equation can be related to the filtering distributions from Subsection 2.1.2 as follows. The term $H(S \mid a)$ is the entropy of the predictive PDF (Equation (2.5)), $H(S \mid Z = z, a)$ is the entropy of the posterior distribution $\tau(b, a, z)$ (Equation (2.4)), and the expectation is taken under the prior PDF of the measurements (Equation (2.6)). Equivalence with Definition A.5[1] is established by basic probability theory.

Mutual information and entropy are non-linear in the belief state. A limitation imposed by choosing a reward function that is non-linear in the belief state is that it limits the set of applicable POMDP solvers, see Table 2.2 on page 47.

3.2 POMDPs applied to problems in robotics domains

Autonomy, meaning the ability to execute tasks independently and to modify behaviour appropriately according to changing objectives, is one of the key properties a robot control system should support (Alami et al., 1998). Given the general nature of the POMDP decision-making framework, and its ability to handle uncertainty in both action effects and sensing outcomes, it is an attractive choice when designing control systems for autonomous robots. This section provides a brief review of some existing approaches to applying POMDPs in robotics domains. The purpose of the discussion is to identify robotic planning problems that are relevant and have attracted attention in the literature, and to review features of such problems. The focus is on work where the applicability of the approach has been demonstrated using real robotic hardware. Furthermore, although a variety of work exists on applying myopic or one-step greedy policies in robotics, e.g. Stachniss et al. (2005); Charrow et al. (2015), we concentrate on approaches finding a closed-loop policy contingent on observations over multiple decision epochs.

A summary of the literature review is presented in Table 3.1. The table indicates in the leftmost column the type of the robot applied; a robotic arm, a wheeled robot, or a humanoid robot. The second column from the left indicates the targeted application. The table further indicates whether the state transitions in the applied POMDP model were stochastic (S) or deterministic (D), if a non-linear-in-the-belief reward function was used (+) or not (-), and whether the state, action, and observation spaces $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{Z}$, respectively, were finite (F) or uncountable (U). In the rightmost column, the type of the POMDP solver applied is indicated, along with references to the related publications.

We remark that most applications consider stochastic state transitions. In the object recognition application (Atanasov et al., 2014b) on the second row of the table, the robot is equipped with a wrist-mounted camera for which viewpoints are selected. Arm motion is considered deterministic, and the task itself is object recognition in a *static* environment. We also note that this application features mixed observability (Subsection 2.2.5.1), as the camera pose is always known.

[1] With the distributions in the definition also conditioned on a.

Table 3.1: Overview of POMDP applications with real robot hardware.

Robot type	Application	State transitions	Non-linear reward	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{Z}$	Solver
Arm	Object manipulation, grasping	S	-	F	F	F	Online sampling-based (Pajarinen and Kyrki, 2015), point-based (Monso et al., 2012)
	Object recognition	D	-	F	F	F	Point-based (Atanasov et al., 2014b)
Wheeled	Exploration	S	+	U	U	U	Policy search (Martinez-Cantin et al., 2009; Kollar and Roy, 2008)
	Target tracking	S	-	F	F	F	Point-based (Spaan et al., 2010)
	Human assistance	S	-	F	F	F	Hierarchical decomposition (Pineau et al., 2003)
	Path planning	S	-	F	F	F	Hierarchical decomposition (Foka and Trahanias, 2007), with online sampling-based (Bai et al., 2015)
Humanoid	Walking	S	-	U	U	U	Policy gradient (Endo et al., 2008)

As indicated by the table, reward functions non-linear in the belief states have been considered for robotic exploration. For example, Martinez-Cantin et al. (2009); Kollar and Roy (2008) apply the negative squared error of the robot's map estimate as the reward. One factor explaining the greater popularity of reward functions linear in the belief is the availability of mature, ready-made software implementations for the related solution algorithms. For instance, Spaan et al. (2010) apply a variant of the Perseus algorithm (Spaan and Vlassis, 2005), the SARSOP algorithm (Kurniawati et al., 2008) is applied by Monso et al. (2012); Atanasov et al. (2014b), while Bai et al. (2015) apply the sampling-based DESPOT algorithm (Somani et al., 2013).

The hieararchical decomposition approach adopted by Pineau et al. (2003); Foka and Trahanias (2007); Bai et al. (2015) refers to dividing the overall POMDP for the whole control task into a hierarchy of smaller layered POMDPs. The overall control action is then determined by reasoning on the solutions of the layered POMDPs. For POMDPs with uncountable state, action, and observation spaces, methods such as direct policy search or policy gradients related to reinforcement learning have been applied.

To obtain a finite representation of the state, action, and observation spaces, some kind of discretization is typically applied. For example, Spaan et al. (2010) discretize the locations of the targets to a finite set of prototype locations, Atanasov et al. (2014b) define a finite set of possible camera locations around the scene where objects are recognised, and Foka and Trahanias (2007) discretize e.g. the robot heading angle.

If uncountable representation of the aforementioned spaces are applied, approximators for the value function and/or policy have to be defined. For example, Martinez-Cantin et al. (2009) represent the value of a parametrised policy by a Gaussian process (GP), and apply a Bayesian optimization scheme to iteratively find an improved policy. In a similar vein, Kollar and Roy (2008) apply training data in reinforcement learning with support vector machines (SVMs) to learn parameters characterising policies. For further discussion on function approximators and reinforcement learning for dynamic programming, we refer the reader to Buşoniu et al. (2010). For a recent survey of reinforcement learning for robotics, see Kober et al. (2013).

3.3 Overview of the canonical sensor management problems

In this section, we define the canonical sensor management problems examined throughout the rest of the thesis. Each of the problems is defined as a POMDP (Definition 2.5). The problems can each be thought of as instances of deploying autonomous robotic agents to execute a task. Depending on the problem, the capabilities and the objective of the agent vary. In defining the canonical problems, the objective is on one hand to exploit information about relevant applications presented in the previous subsection. On the other hand, we are also interested in exploring types of problems so far not covered in the literature, with a specific focus on sensor management.

The problems considered are path planning, environment monitoring, task support, and exploration. In path planning, the robot is attempting to

navigate from a start location to a given goal location. In the task support problem, the robot is executing some more general task where the objective is not related to uncertainty reduction or accuracy of sensing per se. In both of these problems, the sensor subsystem is operated in a way that assists in completing the primary task. In the environment monitoring and exploration problems, the robot is monitoring or exploring an a priori partially known environment. The robot must choose a trajectory along which it may obtain the maximum amount of information about the environment.

The key features of the problems are summarised in Table 3.2. The leftmost column indicates the problem and between parentheses the subsection defining the problem. The second column indicates whether the robot motion in the problem is stochastic (S) or deterministic (D). Robot motion at the level of physical pose and orientation is almost always stochastic (see e.g. (Thrun et al., 2006, Ch. 5)). At a more abstract level, deterministic motion models may be justified: for example in the monitoring problem the robot pose is a distinct spatial location or area rather than an exact pose within the environment, and transitions between the spatial locations are assumed deterministic.

Table 3.2: Summary of key features of the canonical sensor management problems.

	Robot motion	Observability	Non-linear reward	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{Z}$
Path planning (3.4.3)	D	mixed	-	F	F	F
Monitoring (3.4.4)	D	mixed	+	F	F	F
Task support (3.5.1)	S	partial	-	U	F	F
Exploration (3.5.2)	S	partial	+	U	U	F

The third column of the table indicates the degree of observability in the problem, either partial or mixed. Partial observability refers to the typical situation in a stochastic domain, where a belief state describes knowledge about the state without additional assumptions. In a mixed observability case, a part of the state is fully observable, leading to a factored state and belief representation. Mixed observability frequently arises in robotic domains (Ong et al., 2010), for instance when some parts of the robot state are accurately sensed. An example is a robot equipped with a compass but without a global positioning system (GPS) sensor – while its orientation is sensed accurately, its position is not. In the mixed observability cases considered here, robot motion is deterministic and robot position is fully observable.

The fourth column indicates whether the problem features a reward function that is non-linear in the belief state (+) or not (-). While the task support and path planning problems apply a typical POMDP reward function that

depends on the underlying state and action, the monitoring and exploration problems are formulated with an information-theoretic reward function that is non-linear in the belief state.

Finally, the final three columns on the right hand side of the table indicate whether the state space $\mathcal{S}$, action space $\mathcal{A}$, and observation space $\mathcal{Z}$ in the problem are finite (F) or uncountable (U).

Additional details on problem features are given in the following sections. Section 3.4 defines the problems with deterministic robot motion, and Section 3.5 the problems with stochastic robot motion.

3.4 Deterministic robot motion and mixed observability

In this section, we define the path planning and environment monitoring problems. Both of these problems feature deterministic robot motion on a graph. Besides the robot location, the state consists of a set of environmental variables modelling e.g. the traversability of the terrain. The robot's location on the graph is always fully observable, while the states of the environmental variables are only partially observed, leading to mixed observability problems.

Subsections 3.4.1 and 3.4.2 define the robot motion model and the Markovian model for the environmental variables, respectively. Having defined the specific models applied in this context, Subsections 3.4.3 and 3.4.4 formally state the path planning and environment monitoring problems.

3.4.1 Robot motion model

Robot movement is abstracted such that the robot can travel between distinct spatial locations $x \in \mathcal{X}$. Spatial connectivity between the locations is modelled by an undirected graph $G = (\mathcal{X}, E)$. The graph G is an ordered pair comprising a finite non-empty set $\mathcal{X}$ of vertices and a set E of edges which are unordered pairs of vertices. If the robot is currently located at $x \in \mathcal{X}$, it can choose to move to any of the neighbouring locations $N(x) = \{x' \in \mathcal{X} \mid (x, x') \in E\} \subset \mathcal{X}$ that are connected to x by an edge in E.

Figure 3.1 shows an illustration of a simple undirected graph with $\mathcal{X} = \{x_0, x_1, x_2, x_3, x_4, x_5\}$ and $E = \{(x_0, x_1), (x_1, x_4), (x_1, x_5), (x_2, x_3), (x_3, x_4), (x_4, x_5)\}$. The vertices in set $N(x_1)$ are shown shaded in gray colour.

The set of movement actions $\mathcal{A}_x$ available to the robot at x is equal to $N(x)$. Given that the robot motion is deterministic, the robot state transition model $\mathbb{T}_x : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is defined

$$\mathbb{T}_x(x', a, x) = \begin{cases} 1 & \text{if } x' = a \\ 0 & \text{otherwise} \end{cases}. \tag{3.2}$$

In other words, the robot transitions with probability 1 to $x' \in N(x)$.

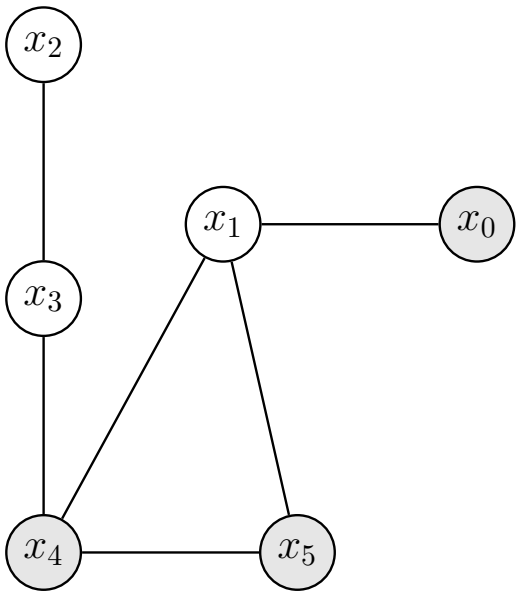


Figure 3.1: An undirected graph with the set $N(x_1)$ of neighbours of x_1 shaded.

3.4.2 Markovian model for environmental variables

Conditional independence relationships between random variables can be modelled as a dynamic Bayesian network (DBN)[2] (Pearl, 1988; Neapolitan, 2004). The model is based on using a directed acyclic graph (DAG) to describe the independence relations between variables.

A directed graph $G = (V, A)$ is an ordered pair comprising a finite non-empty set V of vertices and a set A of directed edges, which are ordered pairs of vertices. If $(v_i, v_j) \in A$, it is said that there is an edge from v_i to v_j and that v_i and v_j are adjacent. Given $v_i, v_j \in V$ and if $(v_i, v_j) \in A$, then v_i is called a parent of v_j.

Suppose $\{v_1, v_2, \ldots, v_k\} \subset V$, and we have $(v_{i-1}, v_i) \in A$, $2 \leqslant i \leqslant k$, i.e. each v_{i-1} is a parent of v_i. A sequence of edges connecting v_1 to v_k is said to be a directed path from v_1 to v_k. A directed cycle is a directed path from any $v \in V$ to itself. A directed graph is *acyclic* if it contains no directed cycles.

We now state the definition of a Bayesian network, adapting from Neapolitan (2004).

Definition 3.1 (Bayesian network)**.** *Let $G = (V, A)$ be a DAG and let $Y = \{Y_v\}_{v \in V}$ be a set of random variables indexed by V. The pair (G, Y) is a Bayesian network if the joint probability of Y satisfies*

$$p(y) = \prod_{v \in V} p(y_v \mid y_{\mathrm{pa}(v)}), \tag{3.3}$$

where $y_{\mathrm{pa}(v)}$ is the set of parent vertices of v.

In a DBN, we have a set of n random variables indexed by the decision epochs $t \in \mathcal{T}$. Following to (Neapolitan, 2004, p. 266), a DBN is defined as follows.

[2]The term *dynamic* Bayesian network is used here to emphasise the time-series nature.

Definition 3.2 (Simple dynamic Bayesian network)**.** *Given a set* $\mathcal{T} = \{0,1, \ldots, d\}$ *of decision epochs and random variables* $\{Y_t\}_{t\in\mathcal{T}}$ *where* $Y_t = \{Y_t^1, Y_t^2, \ldots, Y_t^n\}$*, define:*

1. *an initial Bayesian network with a DAG* G_0 *containing the variables in* Y_0 *and an initial PDF* $p(y_0)$ *and*
2. *a transition Bayesian network which is a template containing a transition DAG* $G_{\rightarrow}$ *containing variables in* $Y_t \cup Y_{t+1}$ *and a conditional transition PDF* $p(y_{t+1} \mid y_t)$*.*

A simple dynamic Bayesian network containing the variables $\{Y_t\}_{t\in\mathcal{T}}$ *consists of the DAG composed of* G_0 *and for* $0 \leqslant t \leqslant d-1$ *the DAG* $G_{\rightarrow}$ *evaluated at* t*; and the joint PDF*

$$p(y_{0:d}) = p(y_0) \prod_{t=0}^{d-1} p(y_{t+1} \mid y_t). \tag{3.4}$$

For the case where the random variables evolve conditional on actions, we include multiple transition templates instead of just one.

Definition 3.3 (Dynamic Bayesian network with actions)**.** *A dynamic Bayesian network with action set* $\mathcal{A}$ *is a simple dynamic Bayesian network where the transition template is replaced by a set of transition templates indexed by* $a \in \mathcal{A}$*, i.e. a set* $\{G_{\rightarrow}^a\}_{a\in\mathcal{A}}$ *of transition DAGs with related conditional transition PDFs* $p(y_{t+1} \mid y_t, a_t)$*.*

Each $G_{\rightarrow}^a = (\{Y_t \cup Y_{t+1}\}, A_a)$, i.e. as before it contains the variables in $Y_t \cup Y_{t+1}$ but has action-dependent connections between the variables determined in the edge set A_a.

As a simple example with $n = 1$, consider a robot acting sequentially in a partially observable environment. Let $\mathcal{T} = \{0, 1, \ldots, d\}$, $d \in \mathbb{N}$ and denote the robot's actions as a_t. A random environmental variable Y_t describes the state of the environment at decision epoch $t \in \mathcal{T}$. The environmental variable evolves according to a Markovian model such that the joint PDF of $Y_{0:d}$ given the robot's actions $a_{0:d-1}$ is

$$p(y_{0:d} \mid a_{0:d-1}) = p(y_0)p(y_1 \mid y_0, a_0) \cdots p(y_d \mid y_{d-1}, a_{d-1}). \tag{3.5}$$

It is straightforward to verify that for any action sequence $a_{0:d-1}$, $Y_{0:d}$ together with the DAG shown in Figure 3.2 is a DBN with actions.

In Figure 3.2, the conditional independence relationships evident in (3.5) are given a graphical representation. Each vertex corresponds to a realisation of a random or deterministic variable, and is associated with a conditional PDF. The shaded circle nodes correspond to realisations of the random environmental variables. The diamond nodes correspond to deterministic variables, in this case the actions a_t. The edges represent causal dependence relationships. Each variable is conditionally independent of the other variables in the graph given its parent variables. For example, the edges toward y_1 imply that it is conditionally independent given y_0 and a_0. The

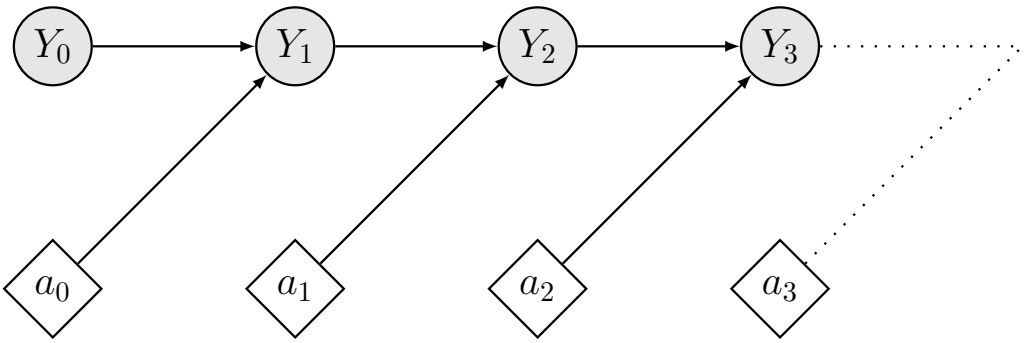


Figure 3.2: A dynamic Bayesian network with actions corresponding to Eq. (3.5).

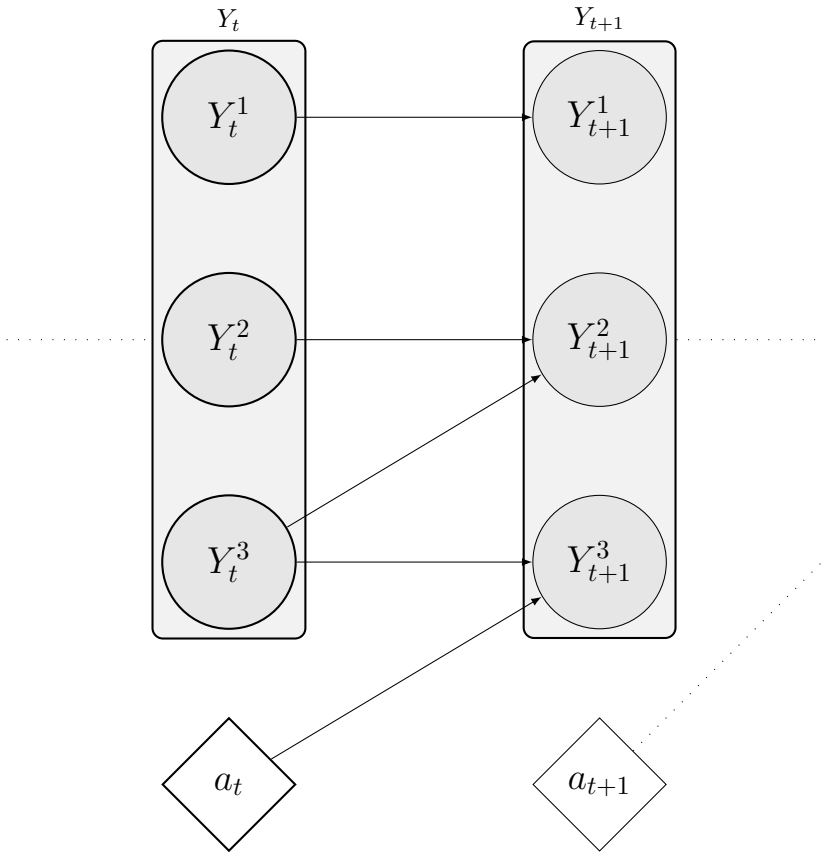


Figure 3.3: A dynamic Bayesian network with actions showing conditional independence relationships between three environmental variables.

drawing continues towards the right in a similar pattern until y_d and a_{d-1}, respectively.

Consider then a more complicated example with $n = 3$ such that $Y_t = \{Y_t^1, Y_t^2, Y_t^3\}$. Each environmental variable assumes values in a specific domain, such that $y_t^i \in \mathcal{Y}_i$. We define a vector $y_t = [y_t^1, y_t^2, y_t^3]^{\mathrm{T}}$ containing all the components. It is possible that there are more intricate dependencies between the individual components of the vectors y_t and y_{t+1} than shown in Figure 3.2. A possible DAG forming a DBN in this case is depicted in Figure 3.3. The vectors y_t and y_{t+1} are contained inside the two shaded boxes, with their components $y_{t,t+1}^i$ drawn inside as circular nodes. In this particular case, y_{t+1}^1 only depends on y_t^1. In contrast, y_{t+1}^2 depends on both y_t^2 and y_t^3, and finally y_{t+1}^3 depends on y_t^3 and the agent's action a_t.

Let $\mathcal{Y} = \bigtimes_{i=1}^{n} \mathcal{Y}_i$ denote the space of realisations for the environmental variables for arbitrary n. By the definitions given in this section, the state transition model $\mathbb{T}_y : \mathcal{Y} \times \mathcal{A} \times \mathcal{Y} \to [0, 1]$ for the environmental variables may be given a concise, factored form by a DBN with actions.

3.4.3 Path planning

Consider a robot traversing an undirected graph $G = (\mathcal{X}, E)$. The robot is navigating towards a goal vertex $g \in \mathcal{X}$. A random environmental variable Y_x is related to each vertex $x \in \mathcal{X}$ in the graph.

The state space in the path planning problem is the cross product of possible robot locations $\mathcal{X}$ and possible environment variable values $\mathcal{Y} = \bigtimes_{x \in \mathcal{X}} \mathcal{Y}_x$. States are pairs $s = (x, y)$ of robot location $x \in \mathcal{X}$ and environment variables $y \in \mathcal{Y}$. As robot locations are fully observable, the belief states are pairs $b = (x, p(y))$ consisting of the robot location and a PDF $p(y) \in \mathcal{P}(\mathcal{Y})$ over environment variables.

At belief state $b = (x, p(y))$, the robot chooses a movement action $a_p \in N(x)$ from the set $N(x)$ of vertices neighbouring x. Let $x' \in N(x)$ denote the successor vertex after the robot executes the movement. In addition, the robot chooses a sensing action $a_m \in N(x')$ to target a vertex neighbouring the successor. Let $\mathcal{A}_b \equiv \mathcal{A}_x$ denote the set of allowed actions at belief state $b = (x, p(y))$, defined as the set of all possible combinations of movement and sensing actions when the robot is at $x \in \mathcal{X}$. Let $\mathcal{A} = \bigcup_{x \in \mathcal{X}} \mathcal{A}_x$. The actions in the problem are ordered pairs $a = (a_p, a_m) \in \mathcal{A}$ of movement and sensing actions. A special case is the goal vertex g, where the action space is a singleton $A_g = \{g, g\}$ to indicate termination of the task.

The state transition model $\mathbb{T} = \mathbb{T}_x \times \mathbb{T}_y$ is factorised into a robot state transition model $\mathbb{T}_x$ and the environmental variable transition model $\mathbb{T}_y$. Robot motion is deterministic, with $\mathbb{T}_x$ defined as in (3.2). The environmental variables evolve independently from the robot's actions, i.e. the robot cannot affect the environment state. The state transition model $\mathbb{T}_y$ is defined by an appropriate DBN model.

After executing an action, the robot obtains a noisy observation $z' \in \mathcal{Z}$ about the environmental variable in the targeted node, according to a stochastic observation model $\mathbb{O}$.

The reward function is state and action dependent and reflects the cost of robot movement and sensing activities. We assume that all sensing actions have the same cost, but note that varying costs are easily incorporated by adding a term dependent on the sensing action to the reward function. When the robot has reached the goal vertex g the reward is equal to zero to indicate termination of the task, i.e. for all states $s = (x, y)$ where $x = g$, we have $R(s, a) = 0 \, \forall a \in \mathcal{A}_g$. There may be a positive reward for entering the goal vertex for the first time. In other states, the reward is strictly negative and dependent on the value of the environmental variable $y_{x'}$ in the successor vertex x' after the movement action.

In the path planning problem considered here, the state, action, and observation spaces $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{Z}$, respectively, are finite.

Given the undirected graph $G = (\mathcal{X}, E)$ and a goal vertex $g \in \mathcal{X}$, the path planning problem is stated as follows.

Problem 3.4 (Path planning)**.** *The path planning problem is a POMDP* $\langle \mathcal{T}, \mathcal{X} \times \mathcal{Y}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}_x \times \mathbb{T}_y, \mathbb{O}, R \rangle$*, where* $\mathcal{T}$ *is the set of decision epochs, and* $\mathcal{X} \times \mathcal{Y}$ *is the finite state space. A state is an ordered pair* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*. The part* $x \in \mathcal{X}$ *is fully observable and* $y \in \mathcal{Y}$ *is partially observable. Thus belief states are ordered pairs of the form* $b = (x, p(y)) \in \mathcal{B}$*, where* $p(y)$ *is a PDF over* $\mathcal{Y}$*. At belief state* $b = (x, p(y))$ *the applicable actions are* $\mathcal{A}_b = \mathcal{A}_x \subset \mathcal{X} \times \mathcal{X}$*, corresponding to possible combinations of movement to* $x' \in N(x)$ *and sensing* $x'' \in N(x')$*. In the goal vertex* $g \in \mathcal{X}, \mathcal{A}_g = \{g, g\}$*. The state transition model is factorised* $\mathbb{T} = \mathbb{T}_x \times T_y$ *with* $\mathbb{T}_x : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ *deterministic as in* (3.2) *and* $\mathbb{T}_y : \mathcal{Y} \times \mathcal{A} \times \mathcal{Y} \to [0, 1]$ *is independent of* $a \in \mathcal{A}$*, i.e.* $T_y(y', i, y) = \mathbb{T}_y(y', j, y)\, \forall i, j \in \mathcal{A}$*. The observation set is* $\mathcal{Z}$*, and the observation model is* $\mathbb{O} : \mathcal{Z} \times \mathcal{X} \times \mathcal{Y} \times \mathcal{A} \to [0, 1]$*. The real-valued reward function is* $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$*, such that for* $s = (x, y)$ *where* $x = g$*,* $R(s, a) = 0\, \forall a \in \mathcal{A}$*.*

The optimal solution of the problem specifies a policy for how to traverse the graph and concurrently operate the sensing subsystem in order to best assist in reaching the goal. Knowledge regarding the goal vertex g is incorporated into the action space and reward function and does not need to be explicitly stated in the problem definition.

The factorised state transition model leads to factorised belief update function $\tau : \mathcal{B} \times \mathcal{A} \times \mathcal{Z} \to \mathcal{B}$ as well. Given a belief state $b = (x, p(y))$, an action a and the resulting observation z', the updated belief is $\tau(b, a, z') = (x', p(y' \mid b, z', a))$, where x' is determined through Eq. (3.2), and $p(y' \mid b, z', a)$ is obtained by Bayesian filtering from $p(y)$ applying $\mathbb{T}_y$ and $\mathbb{O}$. The Bayesian filtering operation is as in Equations (2.4)-(2.6), replacing s,s' by y,y', $b(s)$ by $p(y)$ and $\mathbb{T}$ by $\mathbb{T}_y$ along with the appropriate changes in function arguments.

The path planning problem as stated here is closely related to the Canadian traveller's problem (CTP) (Papadimitriou and Yannakakis, 1991). In the CTP, the cost of traversing an edge is revealed only when a vertex incident to the edge is visited. The cost assumes either a finite or infinite value, denoting traversable and non-traversable edges, respectively. The prior probability of each edge being traversable is known, and the objective is to minimise the expected total cost of traversal.

The remote sensing variant where the agent may sense the cost of any edge regardless whether it is adjacent to the current vertex or not was studied by Bnaya et al. (2009). Psaraftis and Tsitsiklis (1993) considered the case where traversal costs are described by a known function dependent on an environmental variable. The environmental variables are fully observable and independent and evolve according to finite state Markov chains. Problem 3.4 combines characteristics of these variants to yield a case where edge costs can be observed by a noisy sensor and depend on dynamic, partially observable environment variables.

We remark that the formulation of path planning presented here is specific to the sensor management context. An arguably more typical definition of a path planning problem in mobile robotics considers finding paths with

metric map representations satisfying kinematic and dynamic constraints of robot motion (LaValle, 2006).

3.4.4 Environment monitoring

In the environment monitoring problem, the robot is restricted to local sensing only. In other words, the robot can sense the environmental variables at its current location only, with no remote sensing capabilities for adjacent or other locations. The objective is to design a policy for traversing the underlying graph to maximise the total amount of information collected about the partially observable environmental variables. Compared to the path planning problem a goal vertex does not exist, as the task is pure information gathering. Additionally, the environmental variables are no longer assumed to be independent of the robot's actions.

In the environment monitoring problem, the state, action, and observation spaces $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{Z}$, respectively, are finite.

Given the undirected graph $G = (\mathcal{X}, E)$ that the robot is traversing, the environment monitoring problem can be stated as a POMDP as follows.

Problem 3.5 (Environment monitoring)**.** *The environment monitoring problem is a POMDP* $\langle \mathcal{T}, \mathcal{X} \times \mathcal{Y}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}_x \times \mathbb{T}_y, \mathbb{O}, R \rangle$*, where* $\mathcal{T}$ *is the set of decision epochs, and* $\mathcal{X} \times \mathcal{Y}$ *is the finite state space. A state is an ordered pair* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*. The part* $x \in \mathcal{X}$ *is fully observable and* $y \in \mathcal{Y}$ *is partially observable. Thus belief states are ordered pairs of the form* $b = (x, p(y)) \in \mathcal{B}$*, where* $p(y)$ *is a PDF over* $\mathcal{Y}$*. The applicable actions at* $b = (x, p(y))$ *are* $\mathcal{A}_b = N(x) \subset \mathcal{X}$*. The state transition model is factorised* $\mathbb{T} = \mathbb{T}_x \times T_y$ *with* $\mathbb{T}_x : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ *deterministic as in* (3.2) *and* $\mathbb{T}_y : \mathcal{Y} \times \mathcal{A} \times \mathcal{Y} \to [0, 1]$*. The observation set is* $\mathcal{Z}$*, and the observation model is* $\mathbb{O} : \mathcal{Z} \times \mathcal{X} \times \mathcal{Y} \times \mathcal{A} \to [0, 1]$*. The real-valued reward function* $R : \mathcal{B} \times \mathcal{X} \times \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$ *is the MI of the state and observation.*

As in the path planning problem, the belief update equations are factorised. The environmental variables' evolution may be modelled as a DBN.

From (3.1), and noting that the robot location is fully observable, we derive that the MI of the state and observation is

$$R(b, x, y, a) = I(Y; Z \mid a) = H(Y \mid a) - \mathbb{E}_Z \left[H(Y \mid Z = z', a) \right]. \tag{3.6}$$

The interpretation for the reward function is that the more information the robot is expected to gain about the environmental variables by moving to a new location $a = x' \in N(x) \subset \mathcal{X}$, the more useful that movement is.

3.5 Stochastic robot motion and partial observability

The key difference of the problems defined in this section compared to those presented in Section 3.4 is that the robot motion is no longer considered deterministic. We also remove the assumption of mixed observability, leading to a partially observable problem formulation.

The task support problem with a reward function linear in the belief state is considered in Subsection 3.5.1, while a robotic exploration task with an information-theoretic reward function is defined in Subsection 3.5.2.

3.5.1 Task support

As in the path planning problem (Problem 3.4), in the task support problem the objective is not related to information gathering per se. Instead, a robotic agent is executing a task and using the degrees of freedom in its sensing subsystem to optimally support the completion of the task, minding also the cost of the sensing activity. In this sense the problem is similar to the path planning problem of Subsection 3.4.3. However, the problems differ in two important ways. First, the state space in the task support problem is uncountable and the state is not mixed observable. Secondly, in the task support problem there is no clear separation between robot and environment state and hence no independence properties assumed between them.

At each decision epoch, the agent chooses an action $a_p \in \mathcal{A}_p$ on the process or system itself, and an action $a_m \in \mathcal{A}_m$ on the sensing subsystem or sensor selector (c.f. Figure 1.2). The joint action for the POMDP as a whole is an ordered pair $a = (a_p, a_m) \in \mathcal{A}_p \times \mathcal{A}_m = \mathcal{A}$.

In the task support problem, the state space $\mathcal{S}$ is uncountable, while action and observation spaces $\mathcal{A}$ and $\mathcal{Z}$, respectively, are finite.

Problem 3.6 (Task support)**.** *The task support problem is a POMDP* $\langle \mathcal{T}, \mathcal{S}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}, \mathbb{O}, R \rangle$*, where* $\mathcal{T}$ *is the set of decision epochs and* $\mathcal{S}$ *is the uncountable state space. The space* $\mathcal{A}_b = \mathcal{A}_p(b) \times \mathcal{A}_m(b)$ *of applicable actions in belief state* $b \in \mathcal{B}$ *consists of allowed actions* $a_p \in \mathcal{A}_p(b)$ *on the process and allowed actions* $a_m \in \mathcal{A}_m(b)$ *on the sensing subsystem.* $\mathcal{Z}$ *is the observation space,* $\mathbb{T} : \mathcal{S} \times \mathcal{A}_p \times \mathcal{S} \to \mathbb{R}^+$ *is the process state transition model,* $\mathbb{O} : \mathcal{Z} \times \mathcal{S} \times \mathcal{A}_m \to [0, 1]$ *is the observation model for the controllable sensing subsystem, and* $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is a real-valued reward function.*

The observation process is controlled by selection actions $a_m \in \mathcal{A}_m$, which do not affect the evolution of the state process. The reward function encodes the task and is dependent on the state and the joint action (a_p, a_m). The reward determines the preferences between system states, and accounts for differing costs between actions both on the process and on the sensing subsystem.

Although an equivalent formulation can be obtained by choosing $\mathcal{A}$ equal to the set of all possible pairs of process and sensing subsystem actions, the explicit distinction between the two types of actions clarifies the operable nature of the sensing subsystems. Such a choice is natural e.g. in a robot navigation task with an independent machine vision system: a robot can traverse in any desired direction, while the machine vision system can likewise be freely directed to any direction of interest, independent of the direction of movement.

The state space $\mathcal{S}$ could also be composed of the process state and sensing subsystem state. Consider the machine vision system as an example of the sensing subsystem. The state of the machine vision system may be defined as the direction its attention is currently focused on. The presence of the

state allows constraints on the movement of the vision system. For instance, it may be possible to change the focus of attention of the vision system only to nearby directions until the next decision epoch. Then the state transition model decomposes as well; containing a part for the process and a part for the sensing subsystem.

Another example is the case of multiple agents with a single central coordinating agent that has complete authority over other agents. Instead of the multi-agent frameworks discussed in Subsection 2.1.5, the central control allows treatment of this problem as a POMDP. The state space decomposes into the states of each agent, and the state of the process. A concrete example of this type of scenario would be a robot assisted by an additional sensor platform that is commanded by the robot (Melin et al., 2015).

3.5.2 Robotic exploration

Efficient deployment of autonomous robots requires that they are able to explore a priori unknown environments. Exploration and mapping of the environment is crucial for successful completion of further operations, for instance navigation and manipulation tasks. To model the exploration task, we adopt a factorised state representation as in the environment monitoring problem presented in the previous subsection. The objective is to collect the maximum amount of information about the state of the environment.

It may be possible to represent the exploration problem on an abstract level with a finite set of locations that the robot can visit, with actions corresponding to movements between the locations. However, such topological representations of the environment are rarely sufficient alone, and a metric map representation which explicitly describes the geometry of the environment is required for real robot operation (Kuipers et al., 2004). This has important consequences for the task at hand.

The environment state, which we shall denote by a random variable M for "map", is typically much larger than the environment state in the monitoring problem. Even if the map admits a finite representation, such as an occupancy grid map (Moravec, 1988), the robot state may have to be represented as a continuous variable. Motion planning, collision avoidance and object manipulation are examples of tasks where a finite state representation is not sufficient, especially if the robot is operating under kinematic and dynamic constraints (see e.g. Khatib, 1986 and LaValle, 2006, Ch. 13). The state may consist e.g. of the position and orientation of the robot and the states of its manipulator or joints. An immediate consequence is that a natural representation for the robot's action space is also continuous, describing e.g. the effort applied to the joints or wheels of the robot; or in a more abstract manner, e.g. the desired velocity and heading of the robot.

A natural, straightforward representation of the observation space often corresponds to a continuous space: for instance, a single distance measurement in a laser range finder (LRF) scan might be represented on $[0, D_{\max}] \subset \mathbb{R}$, where $D_{\max}$ is the maximum distance measurable. If a finite representation is chosen instead, it often has a large number of observations. For example, each of the N distance measurements in a LRF scan might be discretized

to belong to one of M distinct distance values, leading to NM possible finite observations. For high angular-resolution LRFs, N is in the order of hundreds, and depending on the required accuracy in distincting distance values, an appropriate value of M might be of the same or even greater order. A more detailed discussion on LRF sensing models is provided e.g. by Thrun et al., 2006, Ch. 6.

The robotic exploration problem thus contains challenging features, including

- a factorised state space defined as a cross product of a continuous space representing the robot state and a finite space representing map realisations,
- a continuous action space, and a continuous or large finite observation space, and
- dynamic and kinematic constraints on applicable actions.

The state space in a robotic exploration problem is $\mathcal{X} \times \mathcal{M}$, where $\mathcal{X}$ is the uncountable robot state space and $\mathcal{M}$ is the space of possible realisations of the random variable depicting the map.

The robot motion model $\mathbb{T}_x$ is stochastic. The exact form of the motion model depends on the physical construction of the robot among other variables. For examples of stochastic motion models for mobile robots, we refer the reader to (Thrun et al., 2006, Chap. 5)

For generality, we assume that the map is dynamic. A common feature of robotic exploration is that the robot cannot affect the map by its actions. This leads to a simplification in the map state transition model $\mathbb{T}_m$, as shown in the DBN in Figure 3.4. The next robot state x_{t+1} is conditionally independent given the current state x_t and action a_t, and the next map state m_{t+1} is conditionally independent given m_t. The observations z_{t+1} are affected by both the previous action a_t, as well as the robot state x_{t+1} and map m_t, as indicated by the arrows from the shaded box containing X_{t+1} and M_{t+1}.

The state is a pair (x_t, m_t), and the transition PDF for the DBN describing the robot and map state transitions is

$$p(x_{t+1}, m_{t+1} \mid x_t, m_t, a_t) = p(x_{t+1} \mid x_t, a_t)p(m_{t+1} \mid m_t), \tag{3.7}$$

with $p(x_{t+1} \mid x_t, a_t) \equiv \mathbb{T}_x$ and $p(m_{t+1} \mid m_t) \equiv \mathbb{T}_m$.

In the robotic exploration problem considered here, the state and action spaces $\mathcal{S}$ and $\mathcal{A}$, respectively, are uncountable, and the observation space $\mathcal{Z}$ is finite. The problem is represented as a POMDP as follows.

Problem 3.7 (Robotic exploration)**.** *The robotic exploration problem is a POMDP* $\langle \mathcal{T}, \mathcal{X} \times \mathcal{M}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}_x \times \mathbb{T}_m, \mathbb{O}, R \rangle$*, where* $\mathcal{T}$ *is the set of decision epochs, and* $\mathcal{X} \times \mathcal{M}$ *is the state space consisting of the uncountable set* $\mathcal{X}$ *of possible robot states and a finite set* $\mathcal{M}$ *of possible map realisations. The state is an ordered pair* (x, m) *and belief states are joint distributions* $p(x, m)$ *over the robot state and map. The uncountable sets* $\mathcal{A}_b$ *of applicable actions is determined according to the dynamic and kinematic constraints*

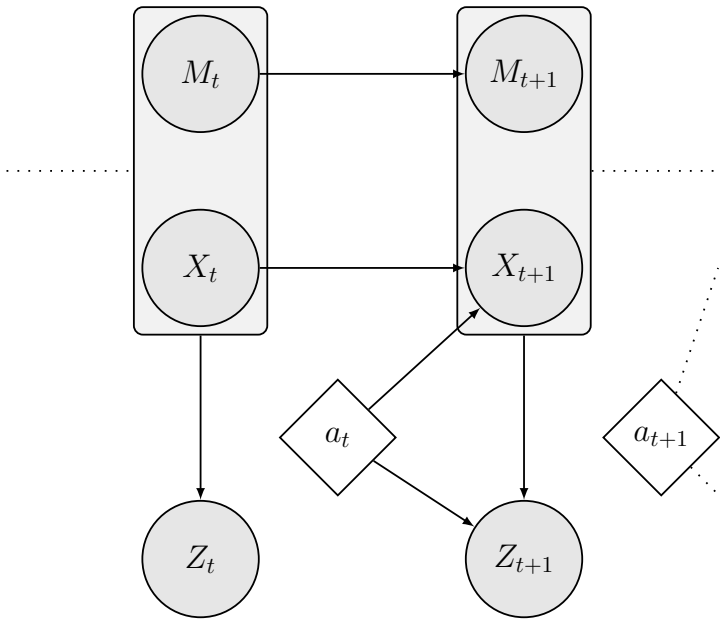


Figure 3.4: A dynamic Bayesian network with actions of the conditional independence relationship between the robot state X_t, environmental state M_t, the observations Z_t, and actions a_t.

of the robot. The state transition model is factorised $\mathbb{T} = \mathbb{T}_x \times \mathbb{T}_m$ *with* $\mathbb{T}_x : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}^+$ *and* $\mathbb{T}_m : \mathcal{M} \times \mathcal{A} \times \mathcal{M} \to [0, 1]$ *which is independent of the action, i.e. for all* $a_i, a_j \in \mathcal{A}$, $T_m(m', a_i, m) = \mathbb{T}_m(m', a_j, m)$. *The observation set is* $\mathcal{Z}$, *and the observation model* $\mathbb{O} : \mathcal{Z} \times \mathcal{X} \times \mathcal{M} \times \mathcal{A} \to [0, 1]$ *is determined according to the robot's sensing capabilities. The real-valued reward function* $R : \mathcal{B} \times \mathcal{X} \times \mathcal{M} \times \mathcal{A} \to \mathbb{R}$ *is the MI of the state and observation.*

Setting the reward function equal to the MI is useful as it selects actions leading to the greatest decrease in the uncertainty (i.e. entropy) of the state estimate. Information-theoretic quantities such as entropy or MI have been proposed as reward functions for robotic information gathering problems also e.g. by Stachniss et al. (2005); Hitz et al. (2014); Atanasov et al. (2014a); Charrow et al. (2014); Nikandrova et al. (2014).

The belief update equation (Equation (2.4)) for the robotic exploration problem requires the joint estimation of the robot's state and map. In mobile robotics, this estimation problem is known as simultaneous localization and mapping (SLAM) (see e.g. Thrun et al., 2006, Chs. 10-13, or Durrant-Whyte and Bailey (2006)), and has been the subject of intense research in the past decades.

If $\mathbb{T}_m = \mathbb{1}_{\mathcal{M}}$, i.e. an identity mapping, the map is stationary. The problem then corresponds to an active SLAM task of selecting robot trajectories to maximise information collected about the map, see e.g. Stachniss et al. (2005); Blanco et al. (2008); Carlone et al. (2010).

4

Solving sensor management problems

This chapter is concerned with the application of the algorithms presented in Chapter 2 to finding policies for the canonical problems defined in Chapter 3. In Section 4.1 we discuss the features of the canonical problems and how they affect the choice of a solution algorithm. A summary of factors to consider when choosing an appropriate algorithm is given.

Based on the analysis of suitable solvers, we identify areas where the effectiveness of existing algorithms for solving POMDP applied to sensor management problems can be improved. Our contribution in these areas is presented in Sections 4.2-4.5. All of the contributions are related to exploiting the structural properties of e.g. the problems' state transition and observation models or other input parameters. These structural properties can be applied e.g. to compute bounds for the optimal value function or to apply domain-specific Monte Carlo approximations.

The effectiveness of heuristics and tightness of upper and lower bounds are crucial for online methods. In Section 4.2 we introduce new heuristics and ways to compute bounds for the path planning problem.

In Section 4.3, we identify through constraint relaxation upper bounds for the environmental monitoring problem. The relaxed problem is shown to be equivalent to a POMDP MAB. Subsection 4.3.3 proves a monotonicity property for the Gittins index that leads to a simple characterisation of the optimal policy in the relaxed problem.

Section 4.4 derives a new bound for a point-based POMDP algorithm's approximation error, based on the L^2-norm of belief states. Based on the error bound, an algorithm for sampling belief states for use in a point-based value iteration algorithm is presented.

In Section 4.5 we present a new Monte Carlo approximation for mutual information in a robotic exploration problem. This approximation can be integrated with sampling-based planners such as the SMC-OLFC method.

4.1 Selection of POMDP solvers for sensor management problems

Table 3.2 on page 58 indicated the key features of the four canonical sensor management problems considered in this thesis. On the other hand, Table 2.2 on page 47 listed the properties of state-of-the-art algorithms for computing policies for POMDPs. By comparing the two tables, we identify the most suitable algorithms for each canonical problem.

4.1.1 Path planning

The reward function in the path planning problem is linear in the belief state. This allows most of the algorithms in Table 2.2 to be applied, although methods that leverage the mixed observability of the domain or the factorised model representation are likely to be more effective.

Exact methods based on linear programming are infeasible in this problem beyond toy problems with in the order of a few dozen states (Hauskrecht, 2000; Pineau et al., 2006). Point-based methods are suitable for the problem, especially if they take advantage of the mixed observability in the problem. The SARSOP algorithm of Kurniawati et al. (2008); Ong et al. (2010) is an example of a point based solver that exploits mixed observability among other features.

Algorithms exploiting problem compression based on structural properties may also be effective in this problem type. However, to the author's knowledge current work in the field does not consider mixed observability models and may hence fail to take into account this property. We conjecture that for this problem point-based methods explicitly taking mixed observability into account will be a better choice.

Among online algorithms (Algorithm 2.1), suitable methods include branch-and-bounding, heuristic search methods, and Monte Carlo methods. Branch-and-bound pruning such as implemented e.g. by RTBSS (Algorithm 2.3) is an appropriate choice if lower and upper bounds for the optimal value function are available. The looser these bounds are, however, the closer the performance is to that of an exhaustive finite-depth search of the belief tree. RTBSS may not be a good choice for problems with a high number of observations, since all observations are explored equally without regard to their relative likelihood. For heuristic search methods, the informativeness of the heuristic function, i.e. how accurately it reflects the differences in the values of belief states, affects the performance (Ross et al., 2008). For both branch-and-bounding and heuristic search, the overhead of computing the bounds or heuristic values is a factor affecting performance.

Monte Carlo methods such as MCTS and SMC-OLFC have few limitations restricting their applicability. For both methods, the ability to sample states, observations, and rewards using a POMDP model or a black box simulator is required. Monte Carlo methods cannot provide worst-case performance guarantees, which may be a limiting factor in some applications. The MCTS algorithm requires no parameter selection beyond choosing an appropriate exploration coefficient, and is well suited for the finite action space of the

Table 4.1: Solution algorithms best suited for the path planning problem from Subsection 3.4.3.

Algorithm	Limitations and performance considerations
Point-based	Methods leveraging mixed observability preferable
RTBSS	Tightness of bounds, computational overhead, number of observations
Heuristic search	Accuracy of heuristics, computational overhead
MCTS	Needs sampling, no worst-case guarantees

path planning problem. Due to the additional parameter tuning required by the SMC-OLFC algorithm, it is not advisable for this problem.

Table 4.1 summarises the suite of suitable solvers for the path planning problem, along with their application limitations and key factors affecting performance.

4.1.2 Environment monitoring

Since the reward function in the environment monitoring problem is non-linear in the belief state, most offline solution methods are not applicable. Point-based grid methods that do not rely on the α-vector representation of the value function and ρPOMDP remain useful for the problem. A factored representation of the POMDP model offers a compact way to represent large problems, and is applied in the environment monitoring problem as well. The properties of the factored representation can be taken advantage of in point-based solvers (see e.g. Shani et al., 2013 and references therein). The mixed observability and deterministic robot motion are examples of such properties in the environmental monitoring problem. A naive implementation that does not leverage any structural properties is unlikely to scale up to handle large problems, limiting the applicability of the point-based grid or ρPOMDP algorithms.

All online algorithms presented in Table 2.2 are applicable for the environment monitoring problem. For branch-and-bound pruning, heuristic search, and sampling based methods similar limitations as outlined in the previous subsection are relevant. The bounds and heuristic values may also be computed taking advantage of structural properties. Algorithms taking advantage of adaptive submodularity are applicable to the special case of the environment monitoring problem that is deterministic, with the only source of uncertainty related to the initial belief state.

Table 4.2 summarises the suite of suitable solvers for the environment monitoring problem, along with their application limitations and key factors affecting performance.

Table 4.2: Solution algorithms best suited for the environment monitoring problem from Subsection 3.4.4.

Algorithm	Limitations and performance considerations
Point-based grid, ρPOMDP	Limited model size if structural properties not exploited
RTBSS	Tightness of bounds, computational overhead, number of observations
Heuristic search	Accuracy of heuristics, computational overhead
MCTS	Needs sampling, no worst-case guarantees
SMC-OLFC	Needs sampling, no worst-case guarantees, needs parameter tuning
Adaptive submodularity	Underlying POMDP must be deterministic

4.1.3 Task support

In the task support problem, the reward function is linear in the belief state. The state space is uncountable, and action and observation spaces are finite. Among offline algorithms, the continuous variant of the Perseus algorithm is applicable provided that a suitable GMM based POMDP model can be obtained. The overhead of compressing the GMM-based representations of belief states and the value function may limit the size and complexity of models that can be handled. For instance, Porta et al. (2006) considered a problem with a one-dimensional state, a linear state transition model with additive Gaussian noise, and an observation model with less than a dozen GMM components.

Table 4.3: Solution algorithms best suited for the task support problem from Subsection 3.5.1.

Algorithm	Limitations and performance considerations
Continuous Perseus	GMM-based POMDP, limited model size
RTBSS	Tightness of bounds, computational overhead, number of observations
Heuristic search	Accuracy of heuristics, computational overhead
MCTS	Needs sampling, no worst-case guarantees
SMC-OLFC	Needs sampling, no worst-case guarantees, needs parameter tuning

Online branch-and-bound pruning and heuristic search methods are applicable with similar considerations as outlined above. Monte Carlo methods are suitable for continuous state spaces as well. In the task support problem, MCTS is applicable with similar limitation and performance considerations

as outlined for the path planning problem. SMC-OLFC requires designing appropriate sampling kernels and choosing a suitable number of particles and iterations to apply. The parameter selection can be expected to have a strong influence on the convergence speed of the algorithm (Kantas et al., 2009; Del Moral et al., 2006).

Table 4.3 summarises the suite of suitable solvers for the task support problem, along with their application limitations and key factors affecting performance.

4.1.4 Robotic exploration

The robotic exploration problem features a reward function non-linear in the belief state and uncountable state and action spaces.

From the algorithms listed in Table 2.2, only SMC-OLFC can handle this combination. If a discretisation of the action space can be made, other online algorithms could also be applied to the problem. However, the discretisation leads to a further loss in optimality, as the feasible space of solutions is restricted to a finite subset of actions. The limitations for these algorithms are as in earlier subsections.

Table 4.4: Solution algorithms best suited for the robotic exploration problem from Subsection 3.5.2.

Algorithm	Limitations and performance considerations
MCTS	Must discretise action space. Needs sampling, no worst-case guarantees
SMC-OLFC	Needs sampling, no worst-case guarantees, needs parameter tuning

Table 4.4 summarises the limitations and performance considerations for the SMC-OLFC algorithm for the robotic exploration problem.

4.2 Heuristics and bounds for path planning

In the path planning problem (Problem 3.4), the robot can operate its sensory subsystem independently from its movement. In this POMDP, the primary task is to reach a given goal, while the operation of the sensory subsystem should be planned such that it best assists completing this task.

The path planning problem is solved approximately by taking into account operating the sensing subsystem over a limited length look-ahead horizon, while ignoring the effect of observations at subsequent decision epochs (Lauri and Ritala, 2012, 2013a). The full belief tree with all action and observation choices is considered up to a given search depth of $d > 0$ decision epochs, and an upper bound or heuristic for the optimal value is used to estimate values of leaf nodes. Operating the sensing subsystem is considered only over

the first d decision epochs. In the following subsections, we describe how to derive an upper bound or a heuristic value for leaf nodes. This bound may be applied e.g. in the RTBSS algorithm (Algorithm 2.3) to find a policy for the POMDP.

4.2.1 Upper bound for leaf nodes

Consider the situation at decision epoch d beyond which the effect of observations is ignored. The basic idea for obtaining an upper bound on the optimal value in the problem until reaching the goal is found by solving a shortest path problem on an undirected graph. As shortest path queries often take the form of finding a path with the minimum cost, we will find a lower bound for the cost of reaching the goal. This is then equivalently an upper bound on the reward, or value, achievable.

Let $G = (\mathcal{X}, E)$ be the undirected graph the robot is traversing, and let $g \in \mathcal{X}$ denote the goal vertex. Let $C(b, g)$ denote the real-valued cost of a shortest path from $x \in \mathcal{X}$ determined by the current belief state $b = (x, p(y))$ to the goal vertex g. A lower bound for $C(b, g)$ is computed by assuming an optimistic attitude towards the values of environment variables (see e.g. Bnaya et al., 2009; Eyerich et al., 2010 for some related examples). We assume that all environment variables take the value that results in the lowest cost of traversal. Adoption of the optimistic attitude is related to the notion of planning with clear preferences investigated by Likhachev and Stentz (2009). Clear preferences mean that one can identify beforehand what is the best value of an unknown variable. However, Likhachev and Stentz assume perfect sensing of hidden variables and completely deterministic underlying problem dynamics differentiating the situation from the one considered here.

Solving the shortest path problem with optimistic costs gives a lower bound $\underline{C}(b, g)$ on $C(b, g)$. Now $-\underline{C}(b, g)$ is a valid upper bound for the optimal value in the problem since the lowest possible costs of traversal were assumed. The upper bound $U(b) = -\underline{C}(b, g)$ can be applied in a branch-and-bound scheme as in Equation (2.22).

4.2.2 Heuristic value for leaf nodes

We next describe how to compute a heuristic value for which the lower bound property cannot be guaranteed. This heuristic is computed by finding the optimal solution of an open-loop version of the shortest path problem using the expected costs of traversal.

The expected reward for any action a at any belief state b can be computed by taking the expectation of $R(s, a)$ under b. Assuming that at decision epochs $t \geqslant d$ no observations are obtained, the belief state after any finite sequence of actions starting from b_d is unique[1]. The expected reward of any action at any decision epoch $t \geqslant d$ can be calculated. For this the assumption that environmental variables evolve independently of the robot's actions is crucial.

[1]Consider the belief tree when there is a single non-informative observation. Then each belief state with a fixed action has a single successor belief state. There is no branching due to observations.

The open loop version of the shortest path problem is then a shortest path problem with deterministically time varying costs given by the negative expectations of the reward function under the belief states encountered at decisions epochs $t \geqslant d$. A shortest path problem on a graph with deterministically time-varying costs is equivalent to a shortest path problem in a time expanded network (Ahuja et al., 2003).

Given an undirected graph $G = (\mathcal{X}, E)$, the time expanded network over k decision epochs is denoted by $G^k = (\mathcal{X}^k, E^k)$. For each vertex $x_i \in \mathcal{X}$, there are $(k+1)$ copies $x_i^d, x_i^{d+1}, \ldots, x^{d+k}$ in $\mathcal{X}^k$, representing the vertex x_i at decision epochs from d to $(d+k)$. Correspondingly, for each edge $e \in E$ there are at most k edges in E^k representing the different possible decision epochs when e can be traversed. If there is an edge $(x_i, x_j) \in E$, edges (x_i^{d+m}, x_j^{d+m+1}) are in E^k for $1 \leqslant m \leqslant k-1$. The costs of the edges in E^k are set according to the expected values with respect to the belief $p(y_t)$ over the environmental variables that can be computed for any $t \geqslant d$. The cost of the shortest path to any copy $g^k \in \mathcal{X}^k$ of the goal vertex $g \in \mathcal{X}$ is then computed. If the time expanded network G^k is constructed implicitly during the search, deciding the number of epochs k is not necessary before starting the shortest path search algorithm.

Define the negative cost of the shortest path in the time expanded network as a heuristic value. This heuristic value is not a valid lower bound for the optimal value function as the effect of observations on the belief state and hence expected costs is ignored. There are two possibilities on how to apply this heuristic value. The first option is to apply it anyway as a lower bound in branch-and-bound search. More aggressive pruning can be achieved at the cost of making a greater sacrifice in the optimality of the solution besides the effect of considering only a limited look-ahead horizon. In this case, no general performance guarantees can be given, rendering the approach risky. The second option is to use the heuristic in an online heuristic search algorithm as explained in Subsection 2.2.4.4. Any performance guarantees of the online search algorithm are preserved, making this approach preferable over the aforementioned lower bound method.

4.3 Multi-armed bandit relaxations for environment monitoring

This section concentrates on the environment monitoring problem (Problem 3.5). The actions in the problem correspond to the robot's possible movements between spatial locations. The applicable actions are constrained to the set of *local* actions, i.e. the neighbouring locations. As discussed earlier, such constraints can be handled in a straightforward manner by applying online planning algorithms. Monte Carlo methods such as MCTS or SMC-OLFC may be applied. If worst case performance guarantees are required, methods such as RTBSS are suitable choices. These methods make use of lower and upper bounds for the optimal value function. In the following, we derive upper bounds for the environment monitoring problem via constraint relaxation and identify conditions when the relaxed problem is a partially observable MAB.

This section gives an expanded treatment of the results we presented in Lauri and Ritala (2015a).

4.3.1 Relaxed environment monitoring problems

By removing constraints on the actions the agent can apply, upper bounds for the value function may be derived. The fact that the optimal solution of such a relaxed problem is an upper bound on the optimal solution of the original problem is easily seen. Let $\mathcal{A}_b$ denote the set of applicable actions in the original problem, and let $\overline{\mathcal{A}}_b \supseteq \mathcal{A}_b$ denote the set of applicable actions in the relaxed problem. The subset relation is evident from the fact that removing constraints can only increase the set of applicable actions. Let $V_k^{\pi^*}$, $\overline{V}_k^{\pi^*}$ denote the optimal value functions for the original and relaxed problem, respectively, when k decision epochs are remaining. For $k = 1$, by Equation (2.14) we have

$$V_1^{\pi^*}(b) = \max_{a \in \mathcal{A}_b} \rho(b, a) \leqslant \max_{a \in \overline{\mathcal{A}}_b} \rho(b, a) = \overline{V}_1^{\pi^*} \tag{4.1}$$

as increasing the set of feasible set in an optimisation problem can only improve the optimal solution. Inductively via the value iteration equations (2.12)-(2.14) the relationship

$$V_k^{\pi^*}(b) \leqslant \overline{V}_k^{\pi^*}(b) \quad \forall b \in \mathcal{B} \tag{4.2}$$

can be shown to hold for all k.

The relaxations for the environment monitoring problem are derived by relaxing the locality constraints on the robot's movement. This is equivalent to adding new edges to the undirected graph $G = (\mathcal{X}, E)$ on which the robot is moving.

A straightforward relaxation is obtained by removing all constraints on the movement. This corresponds to replacing $G = (\mathcal{X}, E)$ by a *complete* graph over the vertices $\mathcal{X}$. A complete graph is one where there is a unique edge between every distinct pair of vertices, i.e. $\forall i, j \in \mathcal{X}, i \neq j : (i, j) \in E$. We can also allow the agent to stay in place by removing the condition $i \neq j$. As the robot can move from every vertex to every other vertex in such a graph, keeping track of the robot's current location becomes redundant and the part $\mathcal{X}$ may be removed from the state space. The state of the problem is simplified into the state of the environment variables only. The relaxation thus obtained is called the universal sensor relaxation.

Problem 4.1 (Universal sensor relaxation)**.** *Let $P = \langle \mathcal{T}, \mathcal{X} \times \mathcal{Y}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}_x \times \mathbb{T}_y, \mathbb{O}, R \rangle$ be an environment monitoring problem (Problem 3.5). The universal sensor relaxation to P is a POMDP $P_u = \langle \mathcal{T}, \mathcal{Y}, \mathcal{A}, \mathcal{Z}, \mathbb{T}_y, \mathbb{O}, R \rangle$, where the state space is $\mathcal{Y}$, and the set of applicable actions is $\mathcal{A} = \mathcal{X}$ for every belief state. The state transition model is the part $\mathbb{T}_y$ from the P. The other parameters are unchanged.*

Given the undirected graph $G = (\mathcal{X}, E)$ the robot is traversing, suppose the robot is at $x \in \mathcal{X}$ and $d > 0$ decisions epochs are remaining. It is obvious that only vertices that can be reached from x by executing at most d

decisions are relevant to consider. Define a function $\mathscr{N} : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ mapping $\mathbb{X} \subset \mathcal{X}$ to the subset of vertices that neighbour any vertex in $\mathbb{X}$, while also considering the possibility of staying at the current vertex:

$$\mathscr{N}(\mathbb{X}) = \bigcup_{\forall x \in \mathbb{X}} \{x\} \cup N(x). \tag{4.3}$$

Clearly, $\mathscr{N}(\{x\}) = \{x\} \cup N(x)$ is the set of vertices reachable from $x \in \mathcal{X}$ by executing a single action. Thus, the set

$$M(x, d) = \underbrace{\mathscr{N} \circ \ldots \circ \mathscr{N}(\{x\})}_{d \text{ times}} \tag{4.4}$$

defined through a composition of d functions $\mathscr{N}$ is the set of vertices reachable from x by executing at most d decisions.

A relaxation that takes into account only these reachable vertices is obtained by replacing $M(x, d)$ by a complete graph in G. The resulting relaxation is called the d-step sensor relaxation.

Problem 4.2 (d-step sensor relaxation)**.** *Let $P = \langle \mathcal{T}, \mathcal{X} \times \mathcal{Y}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}_x \times \mathbb{T}_y, \mathbb{O}, R \rangle$ be an environment monitoring problem (Problem 3.5). The d-step ($d > 0$) sensor relaxation to P at $x \in \mathcal{X}$ is a POMDP $P_d(x) = \langle \mathcal{T}, \mathcal{Y}, M(x, d), \mathcal{Z}, \mathbb{T}_y, \mathbb{O}, R \rangle$, where the state space is $\mathcal{Y}$, and the set of applicable actions is $\mathcal{A} = M(x, d)$ for every belief state. The state transition model is the part $\mathbb{T}_y$, the observation model $\mathbb{O}$ and the reward function R are as in the original problem, except restricted to $M(x, d) \subseteq \mathcal{X}$. The other parameters are unchanged.*

Let $b = (x, p(y))$ be the belief state. Denote by $\mathcal{A}_b$ the actions applicable in this belief state in the original problem. By the definitions of the action spaces of the relaxations, we have the relationship $\mathcal{A}_b \subseteq M(x, d) \subseteq \mathcal{X}$ for any $d > 0$. Denote by $V_k^{\pi^*}(b)$, $V_k^{d,\pi^*}(b)$, and $V_k^{u,\pi^*}(b)$ the optimal value at b when k decision epochs are remaining for P, P_u, and $P_d(x)$, respectively. By the arguments established for Equation (4.1), we can show inductively that for any $k \geqslant 1$,

$$V_k^{\pi^*}(b) \leqslant V_k^{d,\pi^*}(b) \leqslant V_k^{u,\pi^*}(b). \tag{4.5}$$

We finally remark that if there exists some $d \in \mathbb{N}$ such that $M(x, d) = \mathcal{X}$, then $P_d(x) = P_u$, which leads to $V_k^{d,\pi^*}(b) = V_k^{u,\pi^*}(b)$. In other words, when all vertices are reachable within d decisions, the d-step sensor relaxation $P_d(x)$ is equivalent to the universal sensor relaxation P_u.

4.3.2 MAB equivalence of problem relaxations

Having defined the relaxed problems P_u and $P_d(x)$ above, we note that they themselves are still POMDPs and thus hard to solve without additional conditions. The objective of this subsection is to identify when the relaxed problems are POMDP MABs (Definition 2.11). Identifying such conditions is useful since finding the optimal index policy solution of a MAB is easier than computing the backwards induction solution to a POMDP. If the optimal MAB solution may be easily found, it can be practically applied

as an upper bound e.g. in a branch-and-bound algorithm for the original, unrelaxed problem as seen from Equation (4.5).

Informally speaking, in the following we will show that when the environmental variables are partitioned into disjoint subsets such that the state transitions, observations and rewards within each subset do not depend on the variables in other subsets, the relaxed problem is a POMDP MAB. The result has applications in reactive target tracking, where the target's state changes as result of observation actions. The result also holds for the useful special case of stationary environmental variables.

Identifying the bandit arms. We first note that as required in MABs, all actions are applicable in every belief state in both relaxed problems. For any MAB problem, the arms of the bandit and their state must be identified. In the case of the environment monitoring problem, the arms are identified by a partition of the environmental variables according to the action space.

Definition 4.3 (Partition of the environmental variables)**.** *Let $F : \mathcal{A} \to 2^Y$ be a set-valued function mapping actions $a \in \mathcal{A}$ to subsets of $Y = \{Y^1, Y^2, \ldots, Y^n\}$. If*

1. *(Non-emptiness.) $\forall a \in \mathcal{A} : F(a) \neq \varnothing$,*
2. *(Completeness.) $\bigcup_{a \in \mathcal{A}} F(a) = Y$, and*
3. *(Disjointness.) $i, j \in \mathcal{A}, i \neq j : F(i) \cap F(j) = \varnothing$,*

then the family of sets $\{F(a)\}_{a \in \mathcal{A}}$ is a partition of Y.

The interpretation is that F relates the actions to disjoint, non-empty subsets of environmental variables, each of which constitute a single arm of a bandit with in total $|\mathcal{A}|$ arms. An example partition is illustrated in Figure 4.1. A set $Y = \{Y^1, \ldots, Y^6\}$ of environmental variables is partitioned by $F : \mathcal{A} \to 2^Y$, with $\mathcal{A} = \{a_1, a_2, a_3\}$. Each environmental variable Y^i is illustrated in the figure by a dot with a corresponding label. The partition from Definition 4.3 is illustrated as disjoint shaded regions labelled e.g. by $F(a_1)$. For example, according to the figure $F(a_1) = \{Y^1, Y^2, Y^3\}$, and similarly for the other spatial locations.

This type of situation could arise e.g. in an indoor environment quality monitoring task. Each spatial location is a room in the environment, which can be targeted by an action. The environmental variables related to each action correspond to a set of possibly correlated random quantities of interest at the particular room targeted by the action; e.g. carbon dioxide level, number of people present, the amount of ambient light, and so on.

Definition 4.3 can be modified for the d-step sensor relaxation by replacing $\mathcal{A}$ by $\bigcup_{x' \in M(x,d)} \mathcal{A}_{x'}$, i.e. the set of actions applicable in locations $M(x, d)$ in the original problem. Furthermore, the completeness condition 2 may be removed, as it is not necessary that the partition determined by $M(x, d)$ covers Y completely. The other two conditions of non-emptiness and disjointness are still required.

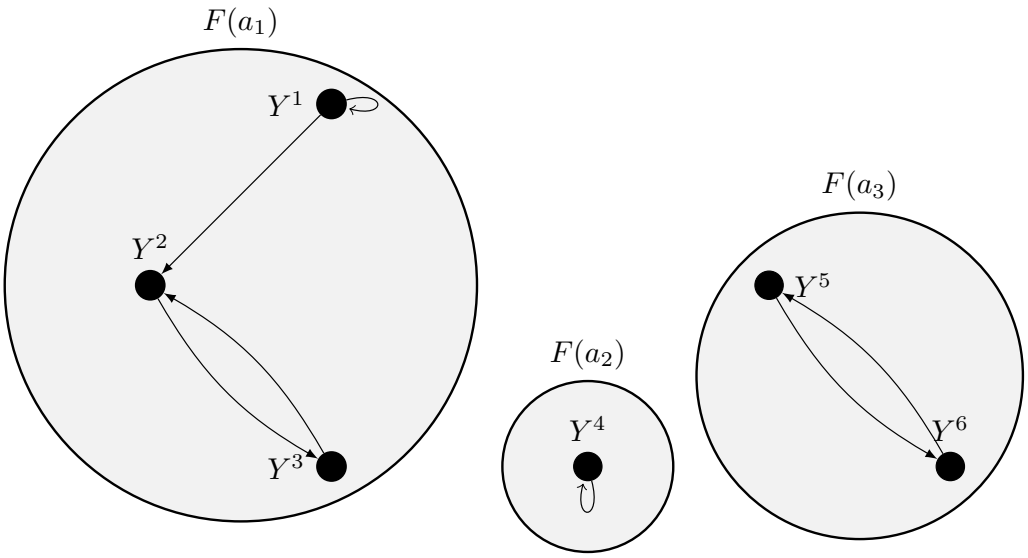


Figure 4.1: Partition of a set $Y = \{Y^1, \ldots, Y^6\}$ of environmental variables by a function $F : \mathcal{A} \to 2^Y$, $\mathcal{A} = \{a_1, a_2, a_3\}$ is shown by the shaded circular areas. The edges depict for a fixed $a \in \mathcal{A}$ a DBN transition template DAG.

Independence of arm states. The state transition model for the environmental variables may be expressed as a dynamic Bayes network with actions (Definition 3.3). The independence between the bandit arms is satisfied under the conditions stated in the following definition.

Definition 4.4 (Independence of state transitions)**.** *Let $\{G^a_{\rightarrow}\}_{a\in\mathcal{A}}$ with $G^a_{\rightarrow} = (\{Y_t \cup Y_{t+1}\}, A_a)$ denote the set of transition DAG templates in a DBN with actions, and let $F_t : \mathcal{A} \to 2^{Y_t}$ be a function defining a partition of $Y_t = \{Y_t^1, Y_t^2, \ldots, Y_t^n\}$ for any decision epoch $t \in \mathcal{T}$. If for any two actions $a, a' \in \mathcal{A}$ and every edge $(Y_t^i, Y_{t+1}^j) \in A_a$*

$$Y_t^i \in F_t(a') \Rightarrow Y_{t+1}^j \in F_{t+1}(a'), \tag{4.6}$$

then the state transitions according to the transition templates in the DBN with actions are independent in the partition defined by F_t.

The definition states that for any fixed action $a_t \in \mathcal{A}$, the environmental variables in $F_{t+1}(a')$ for any $a' \in \mathcal{A}$ are conditionally independent given the environmental variables in $F_t(a')$. In other words, the state transition model for the environmental variables is factorised into a product of conditional transition PDFs:

$$\mathbb{T}_y(y_{t+1}, a_t, y_t) \equiv p(y_{t+1} \mid y_t, a_t) = \prod_{a'\in\mathcal{A}} p(y_{t+1}^{a'} \mid y_t^{a'}, a_t), \tag{4.7}$$

with $Y_t^{a'} = F_t(a') \subset Y_t$ and $Y_{t+1}^{a'} = F_{t+1}(a') \subset Y_{t+1}$.

The independence of state transitions in Definition 4.4 is depicted in Figure 4.1 by the arrows drawn between the environmental variables. The arrows indicate the DBN structure that describes the conditional independence relations between the variables at consecutive decision epochs. The edge structure drawn corresponds to one particular $a \in \mathcal{A}$ and the related transition template DAG $G^a_{\rightarrow}$, and may vary for different actions. An example of such a DBN transition template DAG for a fixed action $a_t \in \mathcal{A}$ is shown

in Figure 4.2. The depicted situation corresponds to the same partition as shown in Figure 4.1. We note that directed arrows indicating conditional independence are only drawn between the action a_t and environmental variables fulfilling the independence conditions of Definition 4.4.

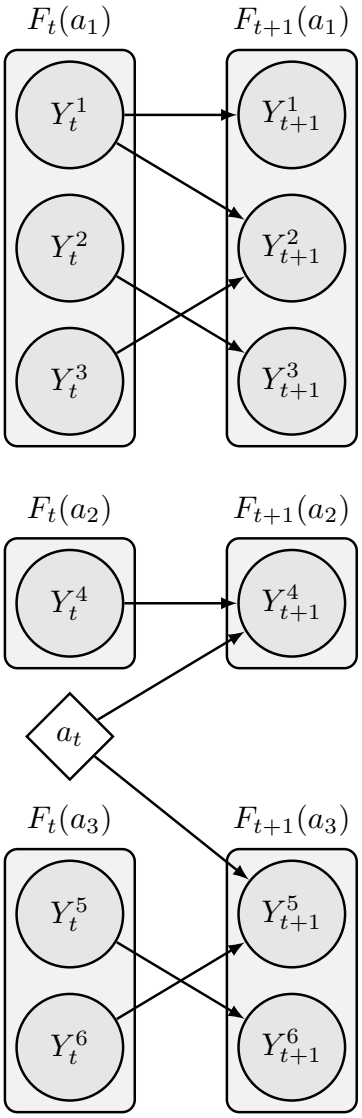


Figure 4.2: An example DBN transition template DAG with independent state transitions in the partition of $Y_t = \{Y_t^1, \ldots, Y_t^6\}$ by $F_t : \mathcal{A} \to 2^Y$, with $\mathcal{A} = \{a_1, a_2, a_3\}$.

With the notion of independent state transitions, we can state the following lemma that says that if the prior over Y_t is independent across the sets determined by F_t, the independence is maintained in the predictive PDF after a state transition.

Lemma 4.5 (Independence preserved in state transition)**.** *Let $F_t : \mathcal{A} \to 2^{Y_t}$ define a partition of $Y_t = \{Y_t^1, Y_t^2, \ldots, Y_t^n\}$ for any decision epoch $t \in \mathcal{T}$, and let $\{G_{\to}^a\}_{a \in \mathcal{A}}$ with PDFs $p(y_{t+1} \mid y_t, a)$ denote a transition template for a DBN with actions where state transitions are independent in the partition defined by F_t. If*

$$p(y_t) = \prod_{i \in \mathcal{A}} p(y_t^i) \tag{4.8}$$

with $Y_t^i = F(i)$, denoting $b \equiv p(y_t)$, then the predictive PDF for all $a \in \mathcal{A}$ preserves the independence:

$$p(y_{t+1} \mid b, a) = \prod_{i \in \mathcal{A}} p(y_{t+1}^i \mid b, a). \tag{4.9}$$

Proof. By Equation (2.5),

$$\begin{aligned} p(y_{t+1} \mid b, a) &= \int_{y_t \in \mathcal{Y}} p(y_{t+1} \mid y_t, a) p(y_t) \mathrm{d}y_t \\ &= \int_{y_t \in \mathcal{Y}} \left(\prod_{i \in \mathcal{A}} p(y_{t+1}^i \mid y_t^i, a) \prod_{i \in \mathcal{A}} p(y_t^i) \right) \mathrm{d}y_t \\ &= \prod_{i \in \mathcal{A}} \int_{y_t^i \in \mathcal{Y}_i} p(y_{t+1}^i \mid y_t^i, a) p(y_t^i) \mathrm{d}y_t^i \\ &= \prod_{i \in \mathcal{A}} p(y_{t+1}^i \mid b, a), \end{aligned} \tag{4.10}$$

where $\mathcal{Y}_i$ denotes the domain of the random variables $F(i)$. The second equation follows by representing the state transition model according to Equation (4.7) and the prior as in the assumption. The rest follows by simple arithmetic. □

Stationarity of unoperated arms. Property 2 of a MAB (page 48) states that arms that are not operated remain in their current state. Additionally, Property 1 states that the state of each arm in the bandit evolves in an uncontrolled manner. Unless further conditions are placed on the state transition model, these properties are violated in the relaxed problems. We next state the conditions that guarantee that Property 2 and Property 1 are satisfied. We remark that for the special case of stationary environment variables, the fact that both properties hold is easy to verify.

The next definition indicates the environmental variables that must remain stationary when an action $a \in \mathcal{A}$ is executed.

Definition 4.6 (Inactive variable)**.** *Let $F_t : \mathcal{A} \to 2^{Y_t}$ define a partition of the set $Y_t = \{Y_t^1, Y_t^2, \ldots, Y_t^n\}$ of random variables. For $a \in \mathcal{A}$, the random variables not in $F_t(a)$ are called inactive variables.*

The next definition states when the inactive variables are stationary and evolve in an uncontrolled manner.

Definition 4.7 (Stationarity of inactive variables)**.** *Let $F_t : \mathcal{A} \to 2^{Y_t}$ define a partition of $Y_t = \{Y_t^1, Y_t^2, \ldots, Y_t^n\}$ for any decision epoch $t \in \mathcal{T}$, and let $\{G_{\to}^a\}_{a \in \mathcal{A}}$ with PDFs $p(y_{t+1} \mid y_t, a)$ denote the set of transition DAG templates in a DBN with actions where state transitions are independent in the partition defined by F_t. Given any action $i \in \mathcal{A}$, if for all $j \in \mathcal{A}$*

$$p(y_{t+1}^j \mid y_t^j, i) = \begin{cases} p(y_{t+1}^j \mid y_t^j) & \text{if } i = j \\ \mathbb{1}_{\mathcal{Y}_j}(y_{t+1}^j, y_t^j) & \text{if } i \neq j \end{cases} \tag{4.11}$$

then the inactive variables in the DBN with actions are stationary in the partition defined by F_t.

Consider an action $a \in \mathcal{A}$. Plugging Equation (4.11) into Equation (4.7), we notice that for a state transition model satisfying Definition 4.7 variables in $F(a)$ transition according to an *action-independent* transition PDF $p(y^a_{t+1} \mid y^a_t)$. The inactive variables in each of the subsets $F(a'), a' \neq a$, transition according to identity mappings, i.e. they are *stationary*. This type of situation might arise e.g. in reactive target tracking, where targets react to the agent's surveillance attempts but otherwise remain stationary.

Consider the DBN transition DAG template shown in Figure 4.2. If Definition 4.7 holds, for any action a_i the DAG would only contain edges from a_i to the members of the subset $F_{t+1}(a_i)$ of environmental variables.

Independence of observations. As seen from Lemma 4.5, for a prior PDF that is independent over the partition of the environmental variables the independence is preserved. If the observation model of the relaxed problems satisfies the next property, the independence property is also preserved in the posterior PDF after an observation is perceived.

Definition 4.8 (Independence of observations)**.** *Let $\mathbb{O} : \mathcal{Z} \times \mathcal{Y} \times \mathcal{A} \to [0, 1]$, and $F_t : \mathcal{A} \to 2^{Y_t}$ define a partition of $Y_t = \{Y^1_t, Y^2_t, \ldots, Y^n_t\}$ for any decision epoch $t \in \mathcal{T}$. If for all $t \in \mathcal{T}$ and all $a \in \mathcal{A}$,*

$$\mathbb{O}(z_{t+1}, y_{t+1}, a) = p(z_{t+1} \mid y_{t+1}, a) = p(z_{t+1} \mid y^a_{t+1}), \tag{4.12}$$

then the observations Z_{t+1} are independent in the partition defined by F_t.

The definition states that the observation the agent obtains after executing $a \in \mathcal{A}$ only depends on the environmental variables in $F(a)$.

If the prior over Y_t is independent across the sets determined by F_t and the observations are independent in the partition defined by F_t, the independence is maintained in the posterior PDF after a measurement update.

Lemma 4.9 (Independence preserved in posterior)**.** *Let $F_t : \mathcal{A} \to 2^{Y_t}$ define a partition of $Y_t = \{Y^1_t, Y^2_t, \ldots, Y^n_t\}$ for any decision epoch $t \in \mathcal{T}$. Furthermore, let $\{G^a_{\to}\}_{a \in \mathcal{A}}$ with PDFs $p(y_{t+1} \mid y_t, a)$ denote the set of transition DAG templates in a DBN with actions where state transitions are independent in the partition defined by F_t, and let $\mathbb{O} : \mathcal{Z} \times \mathcal{Y} \times \mathcal{A} \to [0, 1]$ be such that the observations Z_{t+1} are independent in the partition defined by F_t. If*

$$p(y_t) = \prod_{i \in \mathcal{A}} p(y^i_t) \tag{4.13}$$

with $Y^i_t = F(i)$, denoting $b \equiv p(y_t)$, then for all $a \in \mathcal{A}$ and $z_{t+1} \in \mathcal{Z}$, the independence is preserved in the posterior PDF given by the belief update equation $\tau(b, a, z_{t+1})$:

$$\tau(b, a, z_{t+1}) = p(y^a_{t+1} \mid b, a, z_{t+1}) \prod_{i \in \mathcal{A} \setminus \{a\}} p(y^i_{t+1} \mid b, a). \tag{4.14}$$

Proof. By Lemma 4.5, the predictive PDF is $p(y_{t+1} \mid b, a_t) = \prod_{i \in \mathcal{A}} p(y^i_{t+1} \mid b, a_t)$. By the definition of the belief update equation (Equation (2.4)),

$$\begin{aligned}
\tau(b, a, z_{t+1}) &= p(y_{t+1} \mid b, a, z_{t+1}) \\
&= \frac{\mathbb{O}(z_{t+1}, y_{t+1}, a_t) p(y_{t+1} \mid b, a)}{p(z_{t+1} \mid b, a)} \\
&= \frac{p(z_{t+1} \mid y^a_{t+1}) \prod_{i \in \mathcal{A}} p(y^i_{t+1} \mid b, a)}{p(z_{t+1} \mid b, a)} \\
&= \frac{p(z_{t+1} \mid y^a_{t+1}) p(y^a_{t+1} \mid b, a)}{p(z_{t+1} \mid b, a)} \left(\prod_{i \in \mathcal{A} \setminus \{a\}} p(y^i_{t+1} \mid b, a) \right). \\
&= p(y^a_{t+1} \mid b, a, z_{t+1}) \prod_{i \in \mathcal{A} \setminus \{a\}} p(y^i_{t+1} \mid b, a)
\end{aligned} \tag{4.15}$$

The result follows by replacing the observation model according to Definition 4.8 and algebraic manipulation, and by the Bayes' rule for the final equality. □

Reward contributed only by operated arms. Property 4 of a MAB states that arms that are not operated do not contribute any reward. In the relaxed environment monitoring problem, this requirement corresponds to a reward function that satisfies the following factorisation property.

Definition 4.10 (Independence of rewards)**.** *Let $F_t : \mathcal{A} \to 2^{Y_t}$ define a partition of $Y_t = \{Y^1_t, Y^2_t, \ldots, Y^n_t\}$ for any decision epoch $t \in \mathcal{T}$, and let $R : \mathcal{B} \times \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$ be a reward function with $\mathcal{B} = \bigtimes_{i \in \mathcal{A}} \mathcal{P}(\mathcal{Y}_i)$. If for all $a \in \mathcal{A}$*

$$R(b_t, y_t, a) = R_a(b^a_t, y^a_t), \tag{4.16}$$

where $R_a : \mathcal{P}(\mathcal{Y}_a) \times \mathcal{Y}_a \to \mathbb{R}$, then the reward according to R is independent in the partition defined by F_t.

The definition states that the reward for executing $a \in \mathcal{A}$ only depends on the environmental variables in $F_t(a)$ and the current belief over them. It is required that the belief space is factorized, i.e. that the belief state PDF is independent in the partition defined by F_t.

The next lemma shows that MI satisfies this property, given conditions on the prior belief and observations.

Lemma 4.11 (Independence of MI)**.** *Let $F_t : \mathcal{A} \to 2^{Y_t}$ define a partition of $Y_t = \{Y^1_t, Y^2_t, \ldots, Y^n_t\}$ for any decision epoch $t \in \mathcal{T}$, and let $\{G^a_{\to}\}_{a \in \mathcal{A}}$ with PDFs $p(y_{t+1} \mid y_t, a)$ denote the set of transition templates in a DBN with actions where the inactive variables are stationary in the partition defined by F_t, and let $\mathbb{O} : \mathcal{Z} \times \mathcal{Y} \times \mathcal{A} \to [0, 1]$ be such that the observations Z are independent in the partition defined by F_t. If $p(y_t) = \prod_{i \in \mathcal{A}} p(y^i_t)$, then for all $a \in \mathcal{A}$*

$$I(Y_{t+1}; Z_{t+1} \mid a) = I(F_{t+1}(a), Z_{t+1} \mid a). \tag{4.17}$$

Proof. Denote $b = p(y_t)$. By Equation (3.6),

$$I(Y_{t+1}; Z_{t+1} \mid a) = H(Y_{t+1} \mid a) - \mathbb{E}_{Z_{t+1}} \left[H(Y_{t+1} \mid Z_{t+1} = z_{t+1}, a) \right], \quad (4.18)$$

where $H(Y_{t+1} \mid a)$ is the entropy of the predictive PDF $p(y_{t+1} \mid b, a_t)$, and $H(Y_{t+1} \mid Z_{t+1} = z_{t+1}, a)$ is the entropy of $p(y_{t+1} \mid b, a, z_{t+1})$. According to Lemmas 4.5 and 4.9, the predictive and posterior PDF both maintain the independence property of $p(y_t)$. For independent random variables, their joint entropy is the sum of individual entropies:

$$H(Y_{t+1} \mid a) = \sum_{i \in \mathcal{A}} H(F_{t+1}(i) \mid a), \quad (4.19)$$

and similarly for $H(Y_{t+1} \mid Z_{t+1} = z_{t+1}, a)$. We now write based on Equation (4.18)

$$\begin{aligned} &I(Y_{t+1}; Z_{t+1} \mid a) = \\ &\sum_{i \in \mathcal{A}} H(F_{t+1}(i) \mid a) - \mathbb{E}_{Z_{t+1}} \left[H(F_{t+1}(i) \mid Z_{t+1} = z_{t+1}, a) \right]. \end{aligned} \quad (4.20)$$

We observe from the aforementioned lemmas that for $j \neq a$, $p(y^j_{t+1} \mid b, a) = p(y^j_{t+1} \mid b, a, z_{t+1})$, consequently for $j \neq a$, $H(F_{t+1}(j) \mid a) = \mathbb{E}_{Z_{t+1}} \left[H(F_{t+1}(j) \mid Z_{t+1} = z_{t+1}, a) \right]$. Applying this to Equation (4.20), we are finally left with

$$\begin{aligned} &I(Y_{t+1}; Z_{t+1} \mid a) \\ &\quad = H(F_{t+1}(a) \mid a) - \mathbb{E}_{Z_{t+1}} \left[H(F_{t+1}(a) \mid Z_{t+1} = z_{t+1}, a) \right] \\ &\quad = I(F_{t+1}(a), Z_{t+1} \mid a). \end{aligned} \quad (4.21)$$

□

MAB equivalence of relaxed environment monitoring problems. The following theorem states the conditions under which the relaxed environment monitoring problem is equivalent to a POMDP MAB (Definition 2.11).

Theorem 4.12. *Consider the universal sensor relaxation $P_u = \langle \mathcal{T}, \mathcal{Y}, \mathcal{A}, \mathcal{Z}, \mathbb{T}_y, \mathbb{O}, R \rangle$ (Problem 4.1). Let $Y_t = \{Y_t^1, Y_t^2, \ldots, Y_t^n\}$ denote the set of random environmental variables in the problem. Furthermore, let G_0 be a DAG containing the variables in Y_0 and let $p(y_0)$ be a prior PDF which together with a set $\{G^a_{\rightarrow}\}_{a \in \mathcal{A}}$ of transition templates with PDFs $p(y_{t+1} \mid y_t, a)$ form a DBN with actions defining the state transition model $\mathbb{T}_y$.*

If $\mathcal{T} = \{0, 1, \ldots, \}$ and for all $t \in \mathcal{T}$ there exists a function $F_t : \mathcal{A} \to 2^{Y_t}$ such that

1. *$p(y_0) = \prod_{i \in \mathcal{A}} p(y_0^i)$, and in the partition defined by F_t for all $a \in \mathcal{A}$*
2. *the inactive variables in the DBN with actions are stationary (Definition 4.7),*
3. *the observations Z_{t+1} are independent (Definition 4.8), and*
4. *the reward according to R is independent (Definition 4.10),*

then P_u is a POMDP MAB with $|\mathcal{A}|$ arms.

Proof. By checking that P_u with the conditions given above is equal to the POMDP MAB definition (Definition 2.11).

Equivalence of the decision epochs is obvious.

The state space of P_u is the cross product of states in the partition defined by F_t, i.e. $\mathcal{Y} = \bigtimes_{i \in \mathcal{A}} \mathcal{Y}_i$ where $\mathcal{Y}_i$ denotes the domain of $F(i)$. By Lemma 4.9, the factorisation property in the prior is maintained applying the belief update equation for any sequence of actions and observations. Thus, the belief space in P_u may be viewed as a cross product $\mathcal{B} = \bigtimes_{i \in \mathcal{A}} \mathcal{B}_i$, where $\mathcal{B}_i = \mathcal{P}(\mathcal{Y}_i)$. Equivalence of the state and belief spaces has thus been shown.

The set of applicable actions in P_u is the same in every belief state b, which we denote $\mathcal{A}_b \equiv \mathcal{A} = \{1, 2, \ldots, |\mathcal{A}|\}$, showing equivalence of the action space and the correct number of arms. The observation sets $\mathcal{Z}$ are likewise equal.

By Definition 4.4 and Equation (4.7), the state transition model factorises into independent transition models $\mathbb{T}_i : \mathcal{Y}_i \times \mathcal{A} \times \mathcal{Y}_i \to [0, 1]$. By Definition 4.7, for $a \in \mathcal{A}$, variables *not* in $F_t(a)$ are stationary. Hence the equivalence of the state transition model has been established.

As the observations Z_{t+1} are independent, the observation model $\mathbb{O}$ satisfies the required condition of Equation (2.30). Finally, as the reward according to R is independent, the reward function R satisfies the required condition of Equation (2.35). □

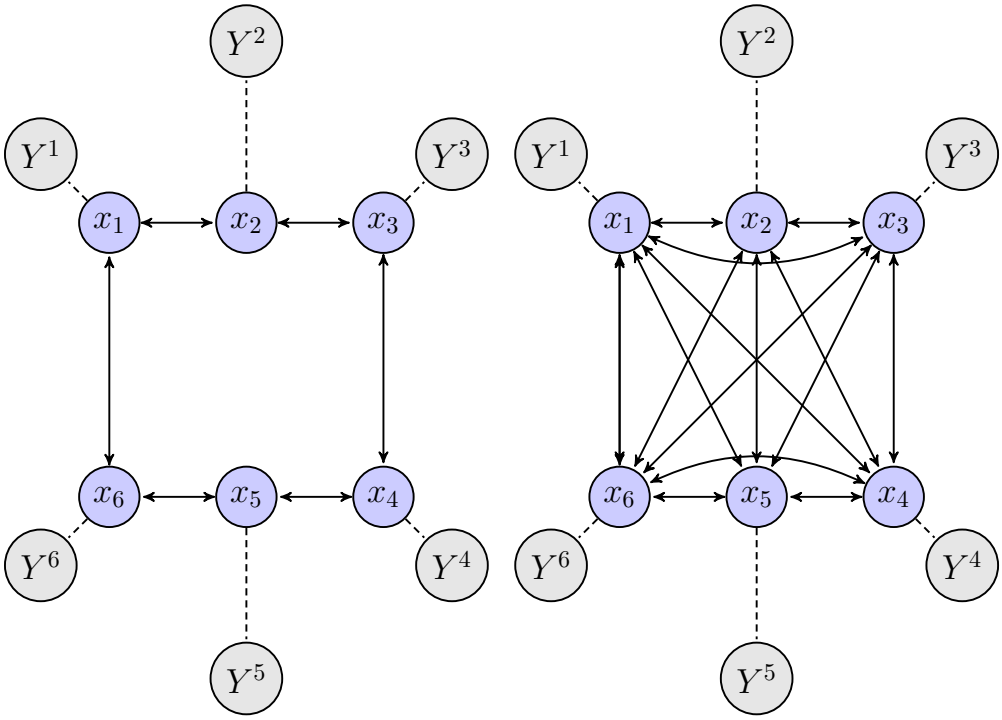
A similar equivalence can be derived for the d-step sensor relaxation with the modified notion of the partition discussed after Definition 4.3. We also note that *any* POMDP that fulfils the conditions of Theorem 4.12 is a POMDP MAB – the result is not restricted only to relaxations.

Of the four conditions required by Theorem 4.12, the one requiring stationarity of inactive variables is the most restrictive. As discussed above, there are instances of reactive target tracking problems that satisfy this condition. Furthermore, we note that since the problem under consideration is a POMDP MAB, the requirement exactly says that the *belief state* over the arm state must remain stationary. As argued by Krishnamurthy and Evans (2001), this is less restrictive than assuming that the *actual state* of the arm does not evolve. The POMDP MAB problem over belief states thus places restrictions primarily on the belief state estimation process that can be controlled. In addition, Krishnamurthy and Evans note that especially in the case of slow dynamics where the true underlying state of the environmental variables evolves slowly, a MAB approximation is justified even if the inactive variables are not exactly stationary[2].

The relation between the original constrained problem and the universal sensor relaxation is illustrated in Figure 4.3. In both subfigures, the nodes labelled by x_i denote the spatial locations the robot may occupy, and the

[2]Specifically, Krishnamurthy and Evans defined slow dynamics as a case where the state transition model is an identity mapping perturbed by a constant proportional to ϵ and showed the MAB approximation to be reasonable in the limit $\epsilon \to 0$.

shaded nodes labelled by Y^j denote sets of environmental variables. The dashed line between x_j and Y^j illustrates the partition of the environmental variables; each set is related to a unique spatial location. The solid arrows between the nodes denoting spatial locations indicate how the agent can move between them. In the constrained problem shown on the left hand side in Figure 4.3a, the agent can only travel to a subset of locations from any given location. In the universal sensor relaxation shown on the right hand side in Figure 4.3b these constraints have been relaxed and the agent can travel freely between any two locations.



(a) The constrained problem. **(b)** The universal sensor relaxation.

Figure 4.3: Relation of the constrained and relaxed problem.

In the universal sensor relaxation, the graph over $\mathcal{X}$ is replaced by a complete graph. In the case of the d-step sensor relaxation starting at $x \in \mathcal{X}$, the difference is that only the subset $M(x, d)$ of spatial locations is fully connected.

4.3.3 Computing the upper bound

Theorem 4.12 states the conditions when the relaxed environment monitoring problem is a POMDP MAB. In this subsection we show how the property may be exploited to compute an upper bound for the optimal value function of the original unrelaxed problem. We prove a monotonicity result for the Gittins index, which may be taken advantage of to practically compute the upper bound. A similar monotonicity result was shown for reward functions linear in the belief state by Krishnamurthy and Wahlberg (2009). We extend it to reward functions concave in the belief state.

Consider a POMDP MAB with n arms. We restrict our attention to the special case where $|\mathcal{Y}_i| = 2$, $1 \leqslant i \leqslant n$ and $|\mathcal{Z}| = 2$ with MI as the reward function. If the ith arm is activated, its state transition model is a two-state

Markov chain with parameters $p^i_{11} \in [0,1]$ and $p^i_{01} \in [0,1]$, where p^i_{jk} denotes the probability that arm i transitions from state j to state k. The observation model can be described by two parameters $p(z' = 1 \mid y_i = 0) = q_+$ and $p(z' = 0 \mid y_i = 1) = q_-$ denoting false positive and false negative probabilities, respectively.

The belief about the state of the ith arm can be represented by a single parameter $p_i \in [0,1]$ denoting the probability that the arm is in state 1, i.e. $b_i = [1 - p_i, p_i]^{\mathrm{T}}$. In this case it is more convenient to work with the representative parameter p_i instead of the belief vector. We make the following surrogate definition for the belief update equation.

Definition 4.13. *Let $\tau_i : \mathcal{B}_i \times \mathcal{A} \times \mathcal{Z} \to \mathcal{B}_i$ be the belief update equation for the ith arm. An equivalent surrogate belief update equation $\hat{\tau}_i : [0,1] \times \mathcal{Z} \to [0,1]$ that operates on the real parameter p_i is defined*

$$\hat{\tau}_i(p_i, z') = g(f(p_i), z'), \tag{4.22}$$

where

$$f(p_i) = (p^i_{11} - p^i_{01})p_i + p^i_{01} \tag{4.23}$$

is the Chapman-Kolmogorov equation (Equation (2.5) and

$$g(p_i, z') = \frac{p(z' \mid y'_i = 1)p(y_i = 1)}{p(z' \mid b_i)} \tag{4.24}$$

is the Bayes' rule for this specific case.

The following monotonicity results for the Chapman-Kolmogorov equation, the surrogate belief update equation and the prior probability of observations are obtained as a special case of (Lovejoy, 1987, Lemma 1.2.).

Lemma 4.14. *Let $p(y_i = 1) \equiv p_i$ denote the prior probability that $Y_i = 1$. If $p^i_{01} < p^i_{11}$, $q_- < 0.5$ and $q_+ < 0.5$, then*

1. *$f(p_i)$ is increasing in p_i and $f(p_i) \in [p^i_{01}, p^i_{11}]$,*
2. *$\hat{\tau}_i(p_i, z')$ is increasing in both p_i and z', and*
3. *$p(z' = 0 \mid b_i)$ is decreasing in p_i and $p(z' = 1 \mid b_i)$ is increasing in p_i.*

We make the following surrogate definition for the reward function.

Definition 4.15. *Let $R_i : \mathcal{B}_i \times \mathcal{Y}_i \to \mathbb{R}$ denote the reward function for the ith arm. Define an equivalent surrogate reward function $\hat{R}_i : [0,1] \times \{0,1\} \to \mathbb{R}$ as*

$$\hat{R}_i(p_i, y_i) = R_i([1 - p_i, p_i]^{\mathrm{T}}, y_i). \tag{4.25}$$

In the environment monitoring problem, R_i is the MI of the state and observation. Applying Equation (3.1) on page 54 the surrogate reward function in this case is independent of y_i and equal to

$$\hat{R}_i(p_i) = H(f(p_i)) - \mathbb{E}_Z \left[H(g(f(p_i), z')] \right, \tag{4.26}$$

where $H(\cdot)$ refers to the entropy of a binary random variable (see Appendix A, Equation A.2). According to (Cover and Thomas, 2006, Thm. 2.7.4), $\hat{R}_i(p_i)$ is concave in $f(p_i)$.

To prove the monotonicity result for the Gittins index, we will need the following lemma determining the conditions when MI is *increasing* in p_i. The lemma is given for MI reward, but can be generalised to other reward functions concave in p_i.

Lemma 4.16. *Let $p^i_{11} \in [0,1]$ and $p^i_{01} \in [0,1]$, and let $p(z' = 1 \mid y'_i = 0) = q_+$, and $p(z' = 0 \mid y'_i = 1) = q_-$ and define functions f and g as in Definition 4.13. Let $\hat{R}_i(p_i) = H(f(p_i)) - \mathbb{E}_Z\left[H(g(f(p_i), z')\right]$ which is concave, and let*

$$p^* = \operatorname*{argmax}_{p\in[0,1]} (H(p) - \mathbb{E}_Z\left[H(g(p, z'))\right]. \tag{4.27}$$

If $p^i_{01} < p^i_{11} \leqslant p^*$, then $\hat{R}_i(p_i)$ is increasing in $p_i \in [0,1]$.

Proof. Let $p_1, p_2 \in [0,1]$ such that $p_1 > p_2$. By Lemma 4.14, $f(p_1) > f(p_2)$, and both are in the range $[p^i_{01}, p^i_{11}]$. Since $\hat{R}_i$ is concave in $f(p_i)$, it is increasing for $f(p_i) \in [0, p^*]$. Since $f(p_2) < f(p_1) \leqslant p^i_{11} \leqslant p^*$, $\hat{\rho}_i(p_1) > \hat{\rho}_i(p_2)$. □

We next prove the monotonicity of the Gittins index of a POMDP MAB arm for this special case with a concave belief-dependent reward function. The proof technique is similar to (Krishnamurthy and Wahlberg, 2009, Theorem 4.1.).

Theorem 4.17 (Monotonicity of the Gittins index)**.** *Consider the ith arm of a POMDP MAB with $\mathcal{Y}_i = \{0,1\}$ and $\mathcal{Z} = \{0,1\}$, with $p^i_{11} \in [0,1]$, $p^i_{01} \in [0,1]$, $p(z' = 1 \mid y'_i = 0) = q_+$ and $p(z' = 0 \mid y'_i = 1) = q_-$, and a reward function $\hat{R}_i(p_i) = H(f(p_i)) - \mathbb{E}_Z\left[H(g(f(p_i), z')\right]$, and define $p^* \in [0,1]$ as in Equation (4.27). If*

1. *$p^i_{01} < p^i_{11} \leqslant p^*$, and*
2. *$q_- < 0.5$ and $q_+ < 0.5$,*

then the Gittins index $v_i(p_i)$ for the arm is increasing in p_i.

Proof. Consider the characterisation of the Gittins index presented in Equation (2.37) through the Bellman recursion.

We first show inductively that the value function $V^i(p_i, M)$ is increasing in p_i. Consider the following value iteration scheme stated in terms of the surrogate belief update function (Definition 4.13) and surrogate reward function (Definition 4.15):

$$V^i_{k+1}(p_i, M) = \max\left\{ \hat{R}_i(p_i) + \gamma \sum_{z'\in\mathcal{Z}} p(z' \mid b_i) V^i_k(\hat{\tau}_i(p_i, z'), M), M \right\}. \tag{4.28}$$

Now choose $V_0^i(p_i, M) = \hat{R}_i(p_i)$ which by Lemma 4.16 is increasing in p_i. Then assume the induction hypothesis: for $p_1 > p_2$, $V_k^i(p_1, M) \geqslant V_k^i(p_2, M)$, i.e. $V_k^i(p_1, M)$ is increasing. Consider V_{k+1}^i as defined above in Equation (4.28). Lemma 4.16 indicates that $\hat{R}_i(p_1) \geqslant \hat{R}_i(p_2)$, so we can concentrate on the latter sum part of the equation. Furthermore by Lemma 4.14, $p(z' = 0 \mid b_1) \leqslant p(z' = 0 \mid b_2)$ and $p(z' = 1 \mid b_1) \geqslant p(z' = 1 \mid b_2)$, and consequently for any increasing function $\phi : \mathcal{Z} \to \mathbb{R}$, $\sum_{z' \in \mathcal{Z}} p(z' \mid b_1)\phi(z') \geqslant \sum_{z' \in \mathcal{Z}} p(z' \mid b_2)\phi(z')$. We may thus write

$$\sum_{z' \in \mathcal{Z}} p(z' \mid b_1) V_k^i(\hat{\tau}_i(p_1, z'), M) \geqslant \sum_{z' \in \mathcal{Z}} p(z' \mid b_2) V_k^i(\hat{\tau}_i(p_1, z'), M), \tag{4.29}$$

and by Lemma 4.14 $\hat{\tau}_i(p_1, z') \geqslant \hat{\tau}_i(p_2, z')$ by which we further bound the above equation from below:

$$\geqslant \sum_{z' \in \mathcal{Z}} p(z' \mid b_2) V_k^i(\hat{\tau}_i(p_2, z'), M). \tag{4.30}$$

Combining the above we have shown

$$\begin{aligned} V_{k+1}^i(p_1, M) & \\ &= \hat{R}_i(p_1) + \gamma \sum_{z' \in \mathcal{Z}} p(z' \mid b_1) V_k^i(\hat{\tau}_i(p_1, z'), M) \\ &\geqslant \hat{R}_i(p_2) + \gamma \sum_{z' \in \mathcal{Z}} p(z' \mid b_2) V_k^i(\hat{\tau}_i(p_2, z'), M) \\ &= V_{k+1}^i(p_2, M) \end{aligned}. \tag{4.31}$$

As value iteration converges to a fixed point, i.e. $V_{k+1}^i(p_i, M) \to V^i(p_i, M)$ as $k \to \infty$, we conclude that $V^i(p_i, M)$ is increasing in p_i. The rest of the proof follows the same steps as (Krishnamurthy and Wahlberg, 2009, Theorem 4.1.). According to Equation (2.37) the Gittins index is

$$v_i(p_i) = \min \left\{ M \in \mathbb{R} : V^i(p_i, M) - M = 0 \right\}. \tag{4.32}$$

Suppose again that $p_1 > p_2$, implying $V^i(p_1, M) \geqslant V^i(p_2, M)$ for all M. Further, $V^i(p_1, v_i(p_2)) - v_i(p_2) \geqslant V^i(p_2, v_i(p_2)) - v_i(p_2) = 0$. According to (Ross, 1983, Lemma 2.1), $V^i(p_i, M) - M$ is decreasing in M. It follows that

$$\min\{M : V^i(p_1, M) - M = 0\} > \min\{M : V^i(p_2, M) - M = 0\}. \tag{4.33}$$

Thus, $v_i(p_1) > v_i(p_2)$. □

Theorem 4.17 says that the Gittins index for an arm satisfying the required properties is increasing in the probability p_i that the arm state is 1. Suppose that p_{01}^i, p_{11}^i and q_- and q_+ are the same for all arms. As the optimal policy in a MAB is to play the arm with the greatest Gittins index, the optimal policy in this case assumes a simple form: select the arm $\operatorname{argmax}_i p_i$.

4.4 Value iteration in continuous-state task support

We consider the case where the continuous-state POMDP for the task support problem (Problem 3.6) is represented via Gaussian mixture models (GMMs).

State transition and observation models and belief states can be represented as GMMs in a continuous-state POMDP. The reward function can likewise be represented by a weighted sum of Gaussians, removing the constraints on the weights.

The α-function representation of the value function in a continuous-state POMDP was introduced by Porta et al. (2006). The representation is analogous to the α-vector representation of Subsection 2.2.1. When the belief states and the POMDP model are represented by GMMs, the α-functions are GMMs as well. Value iteration (Equations (2.12)-(2.14)) with this representation requires computing products of GMMs. In the product of two GMMs $p(x)$ and $q(x)$ the unity-sum property is lost and the result is a weighted sum of Gaussians. The number of components in the product is equal to the product of the number of components n_p and n_q in $p(x)$ and $q(x)$, respectively. To keep the task computationally feasible, the number of components must be reduced by an approximation that maps a GMM to another GMM with a lower number of components. Three of these methods, based on the minimisation of the Kullback-Leibler (KL) divergence $d(p, q)$, or the infinity norm or L^2 norm of $p - q$ are reviewed in Appendix B.

If a point-based value iteration algorithm is applied to the problem, the error between the optimal finite horizon value functional $V_t^{\pi*}$ and the approximate point-based value function V_t^{PB} is bounded by (Brunskill et al., 2010, Theorem 1)

$$||V_t^{\pi*} - V_t^{PB}||_\infty \leqslant \frac{\epsilon_{\mathrm{PB}} + \epsilon_{\mathrm{proj}}}{1 - \gamma}, \tag{4.34}$$

where ϵ_{PB} is the error due to point-based value iteration and ϵ_{proj} is the error due to projecting the GMMs to ones with fewer components. The errors are magnified proportionally to the discount factor γ.

In Subsection 4.4.1, a bound for the error ϵ_{PB} of performing a single point-based value iteration step in a POMDP with a continuous state and finite action and observation spaces is derived. The error bound is especially well suited for the GMM representation, since it contains the L^2-norm between belief states that can be calculated in closed form for GMMs. An algorithm for constructing a set of belief states for use in a point-based value iteration algorithm is constructed in Subsection 4.4.2. The results of this section were published in Lauri and Ritala (2013b).

4.4.1 Error bound for point-based value iteration

We derive a new bound for ϵ_{PB} (Equation (4.34)) based on the L^2-norm of the α-functions and belief states. The proof is similar to Pineau et al. (2006) applying the L^1-norm for a finite-state POMDP and Brunskill et al. (2010) applying the infinity norm in a continuous-state POMDP. The reason for using the L^2-norm is that it can be computed in closed form for the GMM representation of beliefs and α-functions.

Theorem 4.18 (Error bound for point-based value iteration)**.** *Consider the task support problem $\langle \mathcal{T}, \mathcal{S}, \{\mathcal{A}_b\}, \mathcal{Z}, \mathbb{T}, \mathbb{O}, R\rangle$ (Problem 3.6). The error ϵ_{PB} induced by performing a single point-based value iteration step using a subset*

$B_R \subset \mathcal{B}$ of belief states is at most

$$\epsilon_{\mathrm{PB}} \leqslant \frac{\max_{i,j\in\mathcal{A}} ||R(\cdot,i) - R(\cdot,j)||_2 \cdot \max_{b'\in\mathcal{B}} \min_{b\in B_R} ||b' - b||_2}{1-\gamma}, \tag{4.35}$$

where $||\cdot||_2$ is the norm in L^2 function space.

Proof. Let $b' \in \mathcal{B}$ be the belief state at which the point-based value functional has the largest error compared to the full value functional over $\mathcal{B}$, and let $b \in B_R$ be the closest sampled belief state to b' in terms of the L_2 distance. Let α and α' be the α-functions which maximise the value functional at b and b', respectively. If point-based value iteration does not include α' in the value functional, it makes an error of at most

$$\epsilon_{\mathrm{PB}} \leqslant \int_{s\in\mathcal{S}} (\alpha(s) - \alpha'(s))b'(s)\mathrm{d}s. \tag{4.36}$$

We bound the error in the same way as Brunskill et al. (2010):

$$\begin{aligned}
\epsilon_{\mathrm{PB}} &\leqslant \int_{s\in\mathcal{S}} \alpha(s)b'(s) - \alpha'(s)b'(s)\mathrm{d}s \\
&= \int_{s\in\mathcal{S}} \alpha'(s)b'(s) - \alpha(s)b'(s) + (\alpha'(s)b(s) - \alpha'(s)b(s))\mathrm{d}s \\
&\leqslant \int_{s\in\mathcal{S}} \alpha'(s)b'(s) - \alpha(s)b'(s) + \alpha(s)b(s) - \alpha'(s)b(s)\mathrm{d}s,
\end{aligned} \tag{4.37}$$

the last inequality is due to the value of α' at b being lower than that of α, since we earlier defined α to maximise the value functional at b. Continuing,

$$\begin{aligned}
\epsilon_{\mathrm{PB}} &\leqslant \int_{s\in\mathcal{S}} (\alpha'(s) - \alpha(s))(b'(s) - b(s))\mathrm{d}s \\
&\leqslant \int_{s\in\mathcal{S}} |(\alpha'(s) - \alpha(s))(b'(s) - b(s))|\mathrm{d}s \\
&= ||(\alpha' - \alpha)(b' - b)||_1,
\end{aligned} \tag{4.38}$$

where $||\cdot||_1$ denotes the norm in L^1 function space and α, α', b and b' refer to functions. The product $(\alpha' - \alpha)(b' - b)$ is also a function, and we bound the expression by Hölder's inequality as

$$\epsilon_{\mathrm{PB}} \leqslant ||\alpha' - \alpha||_2 ||b' - b||_2, \tag{4.39}$$

where $||\cdot||_2$ is the norm in L^2 space. The distance between the belief states is bounded by (c.f. Pineau et al. 2006)

$$||b' - b||_2 \leqslant \max_{b'\in\mathcal{B}} \min_{b\in B_R} ||b' - b||_2. \tag{4.40}$$

It remains to bound the term $||\alpha' - \alpha||_2$. Since the value functional is constructed as a set of α-functions, each value $\alpha(s)$ of an α-function corresponds to the expected value of executing a sequence of actions conditional on future

observations starting from state $s \in \mathcal{S}$. We begin by considering the worst case loss when a suboptimal action is chosen during a single decision epoch. For a single decision, let $R_{\max} = \max_{i,j \in \mathcal{A}} ||R(\cdot, i) - R(\cdot, j)||_2$ denote the worst case reward loss when an action j is chosen instead of an optimal action i. In the worst case, this loss is encountered at every successive time step as well, leading to a geometric series for the total loss due to the discount factor γ:

$$R_{\max} + \gamma R_{\max} + \gamma^2 R_{\max} + \dots. \tag{4.41}$$

By sum of geometric series, the worst case error between two α-functions corresponds to the worst case total reward loss

$$||\alpha' - \alpha||_2 \leqslant \frac{\max_{i,j \in \mathcal{A}} ||R(\cdot, i) - R(\cdot, j)||_2}{1 - \gamma}. \tag{4.42}$$

Combining (4.42) and (4.40) to (4.39) gives the overall bound

$$\epsilon_{\text{PB}} \leqslant \frac{\max_{i,j \in \mathcal{A}} ||R(\cdot, i) - R(\cdot, j)||_2 \cdot \max_{b' \in \mathcal{B}} \min_{b \in B_R} ||b' - b||_2}{1 - \gamma} \tag{4.43}$$

□

4.4.2 Constructing a set of belief points for point-based value iteration

A point-based value iteration requires a set B_R of belief points on which to execute the value iteration. One application of the bound derived in Theorem 4.18 is to design an algorithm for iteratively constructing B_R in a way that minimises the error bound. At each stage of the algorithm, from a set of candidate beliefs the one which minimises (4.40) is incorporated into B_R.

An algorithm for generating a set of belief states B_R for a point-based value iteration algorithm based on minimising (4.35) is shown in Algorithm 4.1. The algorithm takes as input a POMDP, an initial belief state and the size of the set B_R to construct. The set B_R is initialised to contain b_0. Then the following iterative process is repeated: for each belief currently in B_R, a set B_c of candidate belief states is constructed (Lines 6-11) applying the generative model of the POMDP. Out of these candidate beliefs, the one which most helps minimise the error bound is inserted into B_R (Lines 12-13). The process is repeated until the desired number of belief states have been collected into B_R.

4.5 Mutual information in robotic exploration

We return to the definition of the reward function in the robotic exploration problem (Problem 3.7). Taking advantage of the conditional independence of the next map and next robot state given the current map and robot state as defined in Equation (3.7), we derive a Monte Carlo approximation for MI. The approximation allows separate computation of the MI of the next robot

Algorithm 4.1 An algorithm for constructing a subset of belief states, adapted from (Lauri and Ritala, 2013b).

Input: A POMDP $\langle \mathcal{T}, \mathcal{S}, \mathcal{A}_b, \mathcal{Z}, \mathbb{T}, \mathbb{O}, R \rangle$, an initial belief state b_0, the number of belief states $N_b > 1$ desired.
Output: A set B_R of N_b belief states sampled from the POMDP.

function BUILDBELIEFSET($\langle \mathcal{T}, \mathcal{S}, \mathcal{A}_b, \mathcal{Z}, \mathbb{T}, \mathbb{O}, R \rangle$, b_0, N_b)
  $B_R \leftarrow \{b_0\}$
  $k \leftarrow 1$
  while $k < N_b$ **do**
    for all $b \in B_R$ **do**
      $B_c \leftarrow \varnothing$
      for all $a \in \mathcal{A}_b$ **do**
        $s \sim b(s)$
        $(s', z', r) \sim \mathcal{G}(s, b, a)$
        $B_c \leftarrow B_c \cup \{\tau(b, a, z')\}$
      end for
      $b_{\text{new}} = \underset{b_c \in B_c}{\operatorname{argmax}} \min_{b \in B_R} ||b_c - b||_2$
      $B_R \leftarrow B_R \cup \{b_{\text{new}}\}$
      $k \leftarrow k + 1$
      if $k = N_b$ **then**
        break
      end if
    end for
  end while
  return B_R
end function

state and observation and the MI of the next map and observation. As the approximation uses samples from the related PDFs, it is straightforward to integrate it with the Monte Carlo algorithms discussed in earlier chapters. The approximation additionally avoids maintaining a SLAM filter state while planning which requires considerable computational effort (Stachniss et al., 2005; Blanco et al., 2008; Carlone et al., 2010). The approximation was first reported in Lauri and Ritala (2015b).

We shall make use of the following lemma on classical Monte Carlo integration, see e.g. (Robert and Casella, 1999, Ch. 3.2) for further discussion.

Lemma 4.19. *Let $X \sim p(x)$ and $h : \mathcal{X} \to \mathbb{R}$. The expectation*

$$\mathbb{E}[h(X)] = \int_{\mathcal{X}} h(x)p(x)\mathrm{d}x \tag{4.44}$$

is approximated by

$$h_N = \frac{1}{N} \sum_{i=1}^{N} h(x_i), \tag{4.45}$$

where $x_i \sim p(x)$ for $i = 1, \ldots, N$. By the strong law of large numbers, as $N \to \infty$, h_N converges to $\mathbb{E}[h(X)]$ almost surely[3].

[3] An event is said to occur "almost surely" if it happens with probability one (Grimmett and Stirzaker, 2001).

In the following, we denote variables at decision epoch t by a plain letter and variables at the following decision epoch $(t+1)$ by primed symbols, e.g. x_t becomes x and x_{t+1} becomes x'. However, the random variables S, X, M, and Z, for state, robot state, and map, respectively, refer to them at decision epoch $(t+1)$.

The MI between the state and observation given the current belief b and action a is $I(S; Z \mid b, a)$. Since the PDF of S, the belief state $b = p(x, m)$, is a joint PDF of the robot state X and map M, we equivalently write $I(X, M; Z \mid b, a)$. Accordingly, the MI is expressed via the measurement prior $p(z' \mid b, a)$, the state posterior $p(x', m' \mid b, a, z')$ and the state prediction $p(x', m' \mid b, a)$. From the definition of MI (Definition A.5) we derive

$$\begin{aligned} I(X, M; Z \mid b, a) = &\int_Z p(z' \mid b, a) \cdot \\ &\int_X \int_M p(x', m' \mid b, a, z') \log \frac{p(x', m' \mid b, a, z')}{p(x', m' \mid b, a)} \mathrm{d}m' \mathrm{d}x' \mathrm{d}z'. \end{aligned} \tag{4.46}$$

The following theorem gives an approximation to (4.46). The approximation is derived exploiting the assumption that the robot cannot affect the environment state through its actions and hence $p(x', m' \mid x, m, a) = p(x' \mid x, a)p(m' \mid m)$ (Equation (3.7)). We remark that this does not imply that the internal and environment state are independent, in general $p(x, m) \neq p(x)p(m)$.

Theorem 4.20. *Consider the robotic exploration problem (Problem 3.7). Let $b = p(x, m)$ denote the current belief state, and fix the action $a \in \mathcal{A}$. Assume*

$$p(x', m' \mid x, m, a) = p(x' \mid x, a)p(m' \mid m). \tag{4.47}$$

If $\{x'^{(i)}, z'^{(i)}\}_{i=1}^N \sim p(x', z' \mid b, a)$ is a sequence of N samples from the joint distribution of the robot state and measurement, then the N-sample approximation of the mutual information $I(X, M; Z \mid b, a)$ given b and a is

$$\begin{aligned} I_N = \frac{1}{N} \sum_{i=1}^N \Bigg[&\log \frac{p(x'^{(i)} \mid b, a, z')}{p(x'^{(i)} \mid b, a)} + \\ &\int_{\mathcal{M}} p(m' \mid b, a, z'^{(i)}, x'^{(i)}) \log \frac{p(m' \mid b, a, z'^{(i)}, x'^{(i)})}{p(m' \mid b, a)} \mathrm{d}m' \Bigg] \end{aligned} \tag{4.48}$$

with almost sure convergence $I_N \to I(X, M; Z \mid b, a)$ as $N \to \infty$.

Proof. By the chain rule for mutual information (Definition A.8),

$$I(X, M; Z \mid b, a) = I(X; Z \mid b, a) + I(M; Z \mid X, b, a). \tag{4.49}$$

The first term of (4.49) is the MI of X and Z,

$$I(X; Z \mid b, a) = \int_{\mathcal{Z}} \int_{\mathcal{X}} p(x', z' \mid b, a) \log \frac{p(x' \mid b, a, z')}{p(x' \mid b, a)} \mathrm{d}x' \mathrm{d}z'. \tag{4.50}$$

The second term of (4.49) is the conditional MI of M and Z given X. By Definition A.7,

$$I(M; Z \mid X, b, a) = \mathbb{E}_{Z,X,M}\left[\log \frac{p(m', z' \mid b, a, x')}{p(m' \mid b, a, x')p(z' \mid b, a, x')}\right], \tag{4.51}$$

where the expectation is w.r.t $p(z', x', m' \mid b, a)$. By the assumption (4.47), $p(m' \mid b, a, x') = p(m' \mid b, a)$, and by the rule of conditional probability we obtain

$$\begin{aligned} I(M; Z \mid X, b, a) = &\int_{\mathcal{Z}}\int_{\mathcal{X}} p(x', z' \mid b, a)\cdot \\ &\left[\int_{\mathcal{M}} p(m' \mid b, a, z', x') \log \frac{p(m' \mid b, a, z', x')}{p(m' \mid b, a)} \mathrm{d}m'\right] \mathrm{d}x'\mathrm{d}z'. \end{aligned} \tag{4.52}$$

Now both terms of the right hand side of (4.49); (4.50) and (4.52), respectively, are expectations under the joint PDF $p(x', z' \mid b, a)$. By sampling from this PDF and applying Lemma 4.19, we derive a particle approximation for the left hand side of (4.49) equal to (4.48). □

To apply the result of Theorem 4.20, we need to draw samples from $p(x', z' \mid b, a)$. This is practically achieved by following for each $i = 1, \ldots, N$ the following three-step procedure.

1. Sample $(x^{(i)}, m^{(i)}) \sim p(x, m)$ from the current belief state.
2. Propagate the sample $(x^{(i)}, m^{(i)})$ through the factored state transition model (4.47), yielding $x'^{(i)} \sim p(x' \mid x^{(i)}, a)$ and $m'^{(i)} \sim p(m' \mid m^{(i)})$.
3. The sample from the previous step is distributed according to the PDF $p(x', m' \mid b, a)$. By basic probability,

$$p(x', z' \mid b, a) = \int_{\mathcal{M}} p(z' \mid x', m', a)p(x', m' \mid b, a)\mathrm{d}m', \tag{4.53}$$

and we sample $z'^{(i)} \sim p(z' \mid x'^{(i)}, m'^{(i)}, a)$ from the observation model.

The end result is a particle set $\{x'^{(i)}, z'^{(i)}\}_{i=1}^{N} \sim p(x', z' \mid b, a)$ sampled from the desired PDF.

In Theorem 4.20, the first sum term in I_N is the approximation of the expected information gain on the robot state. The second sum term is the expected information gain on the environment state. The integral term is equal to the KL divergence between the posterior and predictive map information. We have chosen not to apply a particle approximation to this integral. This is due to the fact that the KL divergence is possible to compute exactly in closed form for several map types, in particular for occupancy grid maps. The integral is replaced by a summation over all map cells, and each sum term is equal to the KL divergence between PDFs over a binary-valued random variable. Additionally, the PDF $p(m' \mid b, a, z'^{(i)}, x'^{(i)})$ can be computed in closed form for known state and observation, in the

case of occupancy grid maps this corresponds to mapping with known poses (Moravec, 1988).

Our approximation places some light restrictions on the underlying POMDP. First, we must be able to draw samples from the state transition and observation models either directly or e.g. applying importance sampling. Secondly, evaluating the first sum term in (4.48) requires us to be able to evaluate the predictive probability and the posterior probability of a robot pose $x'^{(i)}$ given an action a and a measurement $z'^{(i)}$. The predictive PDF can be evaluated applying the state transition model. It is not generally easy to find the posterior $p(x'^{(i)} \mid b, a, z'^{(i)})$, but in some cases it can be approximated by a unimodal (typically Gaussian) distribution. As argued by Grisetti et al. (2007), such a case arises e.g. for robots equipped with accurate range sensors such as LRFs. LRF data can be leveraged via scan matching to obtain localisation estimates that are significantly more precise than estimates based only on robot's state transition models, justifying a Gaussian approximation with a small variance.

5

Case studies

In this chapter, a set of case studies in both simulated and real-world robotic sensor management problems is presented. The objectives of the chapter are threefold. First, the objective is to validate the analysis of Section 4.1 on selection of suitable types of solvers for the problems. Secondly, through the case studies general purpose POMDP solvers are compared against solvers that take advantage of domain-specific features according to the ways introduced in Chapter 4. Finally, the goal is to demonstrate that with suitable approximations the POMDP methods presented in this thesis can be applied to large real-world problems in challenging domains such as autonomous robotic exploration.

The chapter presents the case studies according to the order of the canonical problem types of Chapter 3. In Section 5.1, the path planning problem is considered. Section 5.2 is concerned with the environment monitoring problem. Section 5.3 studies the task support problem, and the chapter is concluded in Section 5.4 by considering the robotic exploration problem.

5.1 Path planning

In this section, we study the path planning problem (Problem 3.4) defined on page 64. In this problem, a robot is traversing an environment to reach a given goal state. The basic problem consists of planning a path from a given start vertex to a goal vertex on an undirected graph $G = (\mathcal{X}, E)$.

The edge costs in G are determined by partially observable environment variables, for which we consider two cases. In the first case, the environment variables are independent, modelling local conditions at each vertex. In the second case, the environment variables model a set of moving obstacles that the robot must avoid colliding with. The robot obtains a negative reward if it occupies the same graph vertex with an obstacle. As the obstacles are moving, this means that the environment variables are correlated.

To solve the problem, we build a belief tree over a finite look-ahead horizon while evaluating the values of leaf nodes in the tree by applying the upper bounds and heuristics derived in Subsection 4.2. The receding horizon control principle was applied, where the look-ahead stage was repeated after

executing the first action in the plan. Both cases were studied by means of simulation experiments, presented in the following subsections.

5.1.1 Independent environment variables

Let $G = (\mathcal{X}, E)$ be a two-dimensional, four-connected grid graph with 8 vertices in both dimensions, corresponding to $|\mathcal{X}| = 64$. The start vertex $x_0 \in \mathcal{X}$ and goal vertex $g \in \mathcal{X}$ are in the bottom left and the top right corners of the grid graph, respectively. At $x \in \mathcal{X}$, the robot may choose to move to any of the four neighbouring vertices in $N(x)$, or stay where it is. The robot's state transition model $\mathbb{T}_x$ is deterministic and the current vertex the robot occupies is fully observable.

Each vertex $x \in \mathcal{X}$ has a related binary environmental variable Y^x with domain $\mathcal{Y}_x = \{0, 1\}$. The environment state transition model $\mathbb{T}_y$ is described by an identical two-state Markov chain for each environmental variable. The Markov chain parameters were $p_{11} = 0.9$ and $p_{01} = 0.05$, where p_{jk} denotes the probability that the environmental variable transitions from state j to state k.

At any decision epoch, the robot chooses a vertex to move to (or to stay in place). Let $x' \in \mathcal{X}$ be the vertex the robot chooses to move to. Additionally, the robot chooses to sense the environmental state at another vertex $x_m \in N(x')$. A noisy observation $z' \in \mathcal{Z} = \{0, 1\}$ is perceived with false positive and false negative probabilities $q_+ < 0.5$ and $q_- < 0.5$, respectively.

At each vertex, the environmental state 0 is considered favourable and 1 unfavourable for the robot, indicating e.g. the presence of obstacles. The reward function $R(s, a) = R_m(s, a) + R_c(s, a)$ consists of two parts: the cost of movement R_m and the cost R_c of entering a vertex with an unfavourable environmental state. Noting that the state is $s = (x, y)$, we define

$$R_m(s, a) = \begin{cases} -c_{\text{wait}} & \text{if } x = a \text{ and } a \neq g \\ -c_{\text{move}} & \text{if } x \neq a \text{ and } a \neq g \\ r_{\text{goal}} & \text{if } a = g \end{cases} \tag{5.1}$$

indicating the cost of waiting in place or moving, and a reward for reaching the goal vertex g. For the other term,

$$R_c(s, a) = \begin{cases} 0 & \text{if } y_a = 0 \\ -c_{\text{u}} & \text{if } y_a = 1 \end{cases} \tag{5.2}$$

indicating a negative reward for entering a vertex with an unfavourable environmental state. In the experiment, we set $c_{\text{wait}} = 0.5$, $c_{\text{move}} = 1$, $c_{\text{u}} = 3$, and $r_{\text{goal}} = 10$.

We solved the problem as a POMDP with a discount factor of 0.95. The initial belief $p(y_i)$ over each environmental variable was set to the stationary distribution of the Markov chain.

We applied the RTBSS algorithm with a upper bound for the optimal value function via the optimistic bound of Section 4.2, i.e. assuming all environmental variables to be in the favourable state and taking the shortest

path. A heuristic lower bound by solving a shortest path problem in a time-expanded network was applied, as discussed in Section 4.2. The traversal costs in the time expanded network were the expected rewards under the PDF over the environmental states, propagated the appropriate number of decision epochs. The look-ahead horizon for RTBSS was varied from $d = 1$ to $d = 3$ decision epochs.

We compared the RTBSS algorithm with the solution found by applying POMCP. The number of simulations N_s for POMCP was varied between 2^1 and 2^{11}. Simulations were run up to a depth of 50 decision epochs. As the rollout policy (see Algorithm 2.6), we applied the typical random policy, and an informative policy for path planning that selected actions such that the expected cost to reach the goal in the time expanded network was minimised.

Each experiment was repeated for 20 times for both algorithms. The experiments were continued up to 50 decision epochs.

Table 5.1: Mean and 95% confidence interval of the sum of rewards for RTBSS and the (average, maximum) planning time in seconds as function of the look-ahead depth d. Table adapted from Lauri and Ritala (2013a).

	$d = 1$	$d = 2$	$d = 3$
Sum of rewards	-11.20 ± 2.64	-11.65 ± 1.93	-10.65 ± 1.61
Planning time [s]	(0.89, 2.56)	(24.43, 93.37)	(557.48, 1998.4)

Table 5.1 shows the mean sum of rewards and its 95% confidence interval for RTBSS with each of the look-ahead depths d, along with the average and maximum planning times per decision. A greater look-ahead depth slightly improves the sum of rewards, although the improvement is not significant. However, performance is more consistent as indicated by decreasing confidence intervals as function of d. Computation times increase exponentially as function of d.

Table 5.2: Average and maximum computation times for POMCP with random and informed rollouts as function of the number of simulations N_s.

	Planning time (average, max) [s]	
N_s	Random rollout	Informed rollout
2^1	(0.033, 0.044)	(0.027, 0.190)
2^2	(0.057, 0.091)	(0.048, 0.550)
2^3	(0.103, 0.239)	(0.082, 0.370)
2^4	(0.198, 0.360)	(0.155, 0.630)
2^5	(0.387, 0.532)	(0.311, 0.860)
2^6	(0.764, 1.191)	(0.602, 1.480)
2^7	(1.459, 1.805)	(1.167, 2.210)
2^8	(2.952, 3.740)	(2.341, 3.370)
2^9	(6.291, 9.141)	(4.831, 7.610)
2^{10}	(13.408, 23.100)	(10.162, 19.800)

The corresponding results for POMCP as function of the number of simulations N_s are shown in Figure 5.1 and Table 5.2 for both rollout policies.

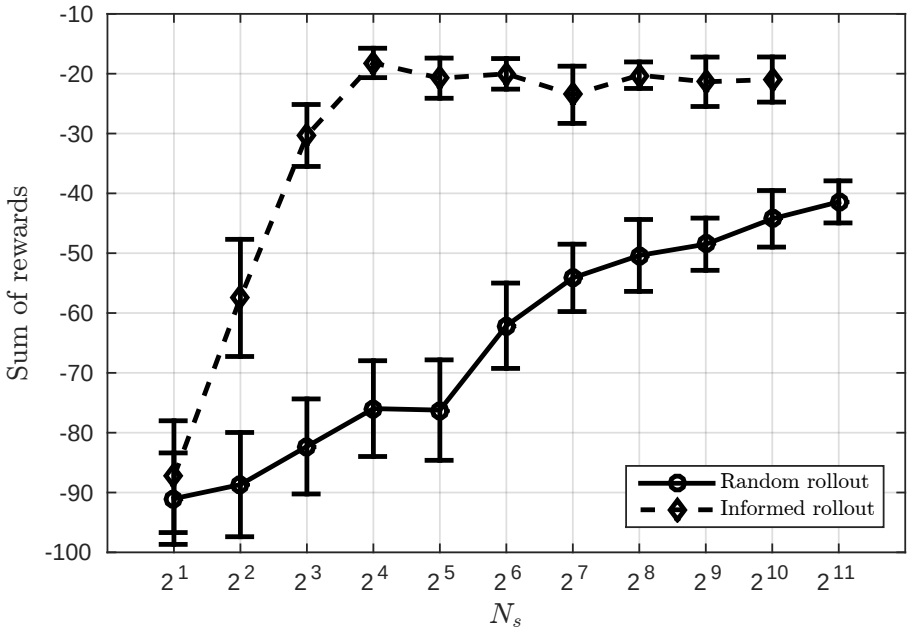


Figure 5.1: Sum of rewards (lines) with 95% confidence intervals (vertical bars) for POMCP with a random or informed rollout policy as function of the number of simulations N_s. Figure adapted from Lauri and Ritala (2013a).

Comparing with Table 5.1, we note that to achieve a sum of reward comparable to RTBSS, POMCP with a random rollout requires more than 2^{11} simulations. However, the informed rollout policy taking advantage of the known graph topology performs much better, reaching a greater sum of rewards than random rollout for $N_s \geqslant 2^2$ simulations. Performance with $N_s = 2^{10}$ simulations is still not as good as RTBSS, however. In terms of planning times, POMCP with informed rollout performs faster than with a random rollout. This is due to the informed rollout being equivalent to a simple look-up operation giving the estimated value of reaching the goal vertex from the end state reached during the tree policy. Direct comparison of computation times between POMCP and RTBSS is not meaningful, as implementation details differ.

The primary difference in the methods is how they estimate the values of the fringe nodes in the belief tree. For RTBSS and POMCP with an informed rollout policy, the time expanded network is applied to estimate the accumulated cost until reaching the goal vertex. This approach taking advantage of the known graph structure outperforms POMCP with a random rollout policy.

5.1.2 Avoiding moving obstacles

In this section, we consider a case similar to the previous subsection but with spatial dependencies in the environmental variables. A maze-like environment as shown in Figure 5.2 was set up. A robot is traversing an undirected graph $G = (\mathcal{X}, E)$. The robot starts at vertex $x_0 \in \mathcal{X}$ in the bottom left hand corner of the environment had the task of navigating to the goal vertex $g \in \mathcal{X}$ in the upper right hand side of the environment, as indicated by the triangle markers in the figure. Each square in the figure corresponds to a

graph vertex, with the dark squares corresponding to vertices that cannot be entered. At $x \in \mathcal{X}$, the robot may choose to move to any of the at most eight neighbouring vertices in $N(x)$. Staying in place was not permitted in this experiment. The state transition model $\mathbb{T}_x$ for the robot is deterministic, and its current location $x \in \mathcal{X}$ is always fully observable.

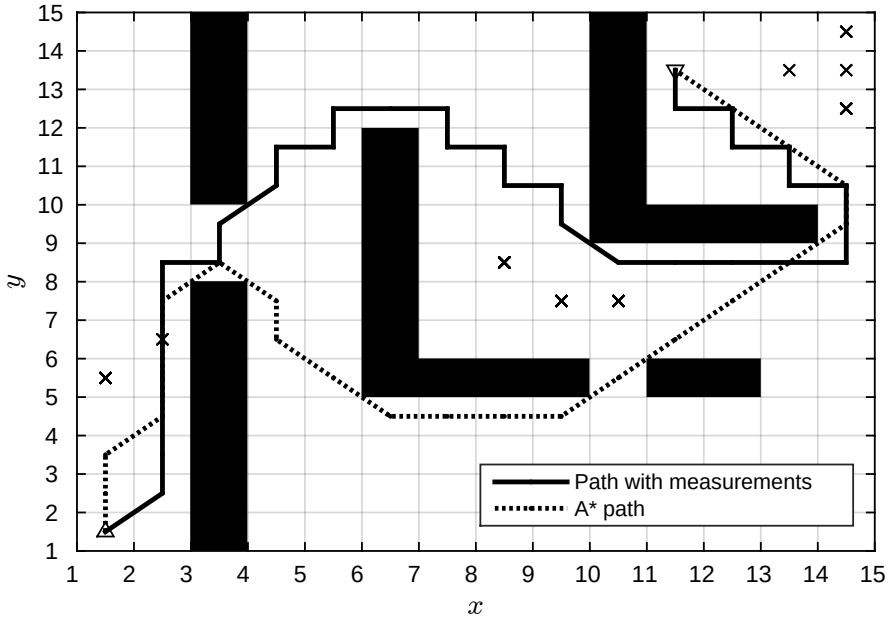


Figure 5.2: The maze navigated by the robot, where the black areas are not traversable. The up- and downward facing triangle markers denote the start and goal locations, respectively. The solid line denotes the path taken by the robot optimising the operation of its measurement resources. A path computed by an A* search using the expected traversal costs based on information in the initial belief state is shown by the dashed line. The black crosses indicate locations where an obstacle was present at any time during the experiment. Figure adapted from Lauri and Ritala (2012).

Four moving obstacles y_1, y_2, y_3 and y_4 with random walk dynamics are also traversing the graph. The state transition model $\mathbb{T}_i$ for any obstacle i is such that with probability $p_s = 0.9$ it remains at its current location, and with probability $1 - p_s = 0.1$ it moves to a neighbouring vertex uniformly at random. The presence of obstacles at any vertex is partially observable.

A separate belief over the location of each of the obstacles was maintained. Furthermore, the beliefs over each obstacle were approximated to be independent.

While traversing the environment, the robot may observe its surroundings by a noisy sensor with eight operational degrees of freedom. The degrees of freedom correspond to choosing two squares adjacent to the robot's current vertex to observe. The robot first selects a single cardinal direction: either north, east, south, or west. This is the primary direction of observation where the robot focuses its attention. In addition, for any choice of cardinal direction the robot can choose another secondary focus of attention such that it is adjacent to both the robot and the primary focus of attention. The eight observation modalities are indicated in Figure 5.3. The robot observes one bit of information per vertex observed, i.e. $\mathcal{Z} = \{0, 1\}^2$, with

0 corresponding to no obstacle observed and 1 corresponding to obstacle observed. The observation model $\mathbb{O}$ is such that there is a symmetric false positive and false negative probability $p_1 = 0.1$ for the primary focus of attention and $p_2 = 0.25$ for the secondary focus of attention.

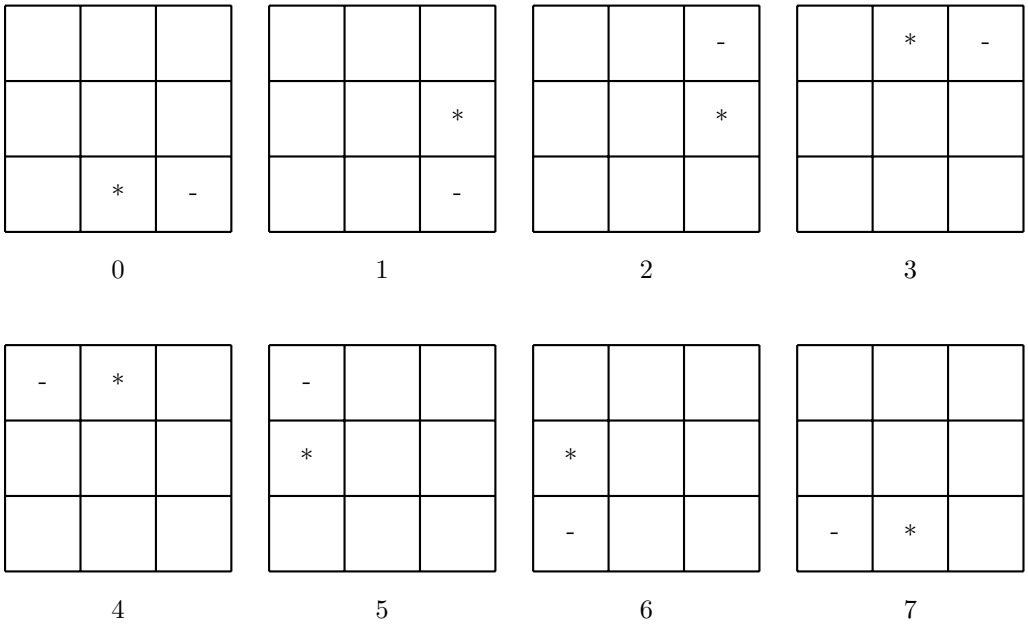


Figure 5.3: The eight observation options available to the robot. The robot is located at the centre of each grid labelled consecutively from 0 to 7. The asterisk "*" indicates the cardinal direction corresponding to the primary focus of attention, and the hyphen "-" indicates the secondary focus of attention.

As in the previous subsection, the reward function $R(s,a) = R_m(s,a) + R_c(s,a)$ consists of the cost of movement R_m and the possible cost R_c of entering a vertex with an obstacle. Noting that the state is $s = (x,y)$, we define

$$R_m(s,a) = -c_{\text{terr}}(a) + \begin{cases} -c_{\text{move}}(x,a) & \text{if } a \neq g \\ r_{\text{goal}} & \text{if } a = g \end{cases}. \tag{5.3}$$

Here $c_{\text{terr}}(a)$ is a terrain cost of entering vertex $x' = a$, sampled uniformly at random for each vertex from the range $[0,1]$. The movement cost $c_{\text{move}}(x,a)$ from x to $a = x' \in n(x)$ is 0.1 times the Euclidean distance between the vertices[1]. A one-time reward $r_{\text{goal}} = 100$ was accumulated then the goal vertex g is entered. For the other term,

$$R_c(s,a) = \begin{cases} 0 & \text{if } \nexists i \in \{1,2,3,4\} : y_i = a \\ -c_{\text{coll}} & \text{if } \exists i \in \{1,2,3,4\} : y_i = a \end{cases} \tag{5.4}$$

which indicates that a collision cost of $c_{\text{coll}} = 1500$ is accumulated if the agent enters a vertex $x' = a$ where there is at least one moving obstacle.

We solved the problem applying RTBSS with the optimistic upper bound and a lower bound obtained by the time expanded network approach. The solid line in Figure 5.2 illustrates the path taken by the robot while optimising the use of its observation resources to detect and avoid the moving obstacles. A look-ahead depth of $d = 2$ was applied in this case. Compared to the A*

[1]Distance measured in squares as seen in Figure 5.2.

solution, a preference for non-diagonal movements is seen in several parts of the path. Moving non-diagonally allows the robot to observe more vertices possibly traversed in the near future, helping it avoid obstacles.

In this problem, the depth of the look-ahead is limited by the more complicated system dynamics compared to the previous subsection. For $d = 1$, required planning times were less than 1 second, and for $d = 2$, on average 70 seconds with maximum times around 200 seconds. Experimenting with $d = 3$, we could not obtain solutions in a reasonable time.

The computational burden is the most severe barrier to the application of RTBSS to the problems presented in this section. As a non-valid upper bound via the time expanded network was applied for RTBSS, the solution it finds cannot be guaranteed to be optimal. Based on the experimental data, we conjecture that POMCP with an informative rollout policy is preferable to RTBSS in this problem type.

5.2 Environment monitoring

In the environment monitoring problem (Problem 3.5), a robot is traversing an undirected graph and observing sets of environmental variables related to the vertices in the graph. We determined in Section 4.3 the conditions when the environmental monitoring problem can be relaxed into a POMDP MAB. In this section, we empirically examine the effects of using the upper bounds obtained via the relaxation to solve the original, constrained monitoring problem. We consider two cases: one where the MAB relaxation conditions are fulfilled and one where some of the conditions are violated. In the first case, the upper bounds obtained via the MAB relaxation are valid, while in the second case the MAB value is applied as a heuristic value. We compare the RTBSS algorithm to the sampling-based POMCP algorithm.

5.2.1 Problem definition

A robot is monitoring environmental variables on an undirected graph $G = (\mathcal{X}, E)$. The graph vertices are arranged in a two-dimensional grid configuration and each of them is connected to their four neighbouring vertices in the cardinal directions. The grid graph has 6 vertices in each dimension, leading to a total of $|\mathcal{X}| = 6 \cdot 6 = 36$ vertices. At vertex $x \in \mathcal{X}$ the robot can choose to move to any of the neighbouring vertices $N(x)$ or stay in place at x, leading to an action set $\mathcal{A}_x = N(x) \cup \{x\}$. The robot's state transition model $\mathbb{T}_x : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is defined such that choosing $a \in \mathcal{A}_x$ deterministically transitions the robot to $x' = a$.

The set of environmental variables was $Y = \{Y^1, Y^2, \ldots, Y^{|\mathcal{X}|}\}$. A function $F : \mathcal{A} \to 2^Y$, defined $F(a) = Y^a$ for each a, determines a partition of the environmental variables according to the actions, i.e. each vertex $x \in \mathcal{X}$ is related to a corresponding Y^x. Each $y_i \in \{0, 1\}$ indicates whether a target is present at i (1) or not (0).

The target dynamics are independent, $\mathbb{T}_y(y', a, y) = \prod_{i \in \mathcal{A}} \mathbb{T}_i(y^{i'}, a, y^i)$. More specifically, the targets act differently depending on the robot's presence.

The state transition models are given by

$$\mathbb{T}_i(y^{i'}, a, y^i) = \begin{cases} w_i(y^{i'}, y^i) & \text{if } a = i \\ r_i(y^{i'}, y^i) & \text{if } a \neq i \end{cases}, \tag{5.5}$$

where w_i, r_i are Markov chains $\mathcal{Y}_i \times \mathcal{Y}_i \to [0, 1]$ describing target i's behaviour when the robot is present at i or not, respectively.

At each decision epoch, the robot obtains a noisy binary measurement $z' \in \mathcal{Z} = \{0, 1\}$ of the presence of a target at its current location. A PMF

$$q(z' = 0 \mid y', a) = \begin{cases} 1 - q_- & \text{if } y'_a = 0 \\ q_+ & \text{if } y'_a = 1 \end{cases} \tag{5.6}$$

with parameters $q_- < 0.5, q_+ < 0.5$ indicating false negative and false positive probabilities, respectively, gives the conditional probability of perceiving either measurement conditional on the action and the environmental variables. The case $q(z' = 1 \mid y', a)$ is obtained by $1 - q(z' = 0 \mid y', a)$. The two cases fully specify the complete observation model $\mathbb{O} : \mathcal{Z} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$.

The reward is equal to the mutual information of the state and observation as discussed in Section 3.4.4. The initial belief over the environmental variables is independent such that $p_0(y) = \prod_{i \in \mathcal{A}} p_0(y^i)$.

5.2.2 MAB conditions satisfied

We first examine the case where the relaxed versions of the environment monitoring problem are such that they satisfy the MAB conditions in Theorem 4.12. If the conditions hold, either of the relaxations introduced in Subsection 4.3.1 can be applied to find the bounds that satisfy Equation (4.5).

To fulfil the conditions we set r_a as an identity mapping for every $a \in \mathcal{A}$. Thus the targets are stationary if the agent is not observing them. The state transition models w_a are two-state Markov chains on $\mathcal{Y}_i = \{0, 1\}$, with parameters p^i_{01}, p^i_{11}, where p^i_{jk} denotes the probability that Y^i transitions from state j to state k.

We set $q_- = q_+ = 0.05$, leading to MI reaching its maximum value at $p^* = 0.5$. We sampled uniformly at random $p^i_{01} \in [0.0, 0.2]$ and $p^i_{11} \in (0.2, p^*]$. For each $1 \leqslant i \leqslant |\mathcal{X}|$, the parameters p^i_{01} and p^i_{11} were the same, enabling straightforward application of the monotonicity result for the Gittins index from Theorem 4.17 at page 90.

A naive approximation to the problem is to apply a greedy policy that selects at each decision epoch an action that maximises the immediate expected reward. We compared the greedy policy to an optimal policy found via an exhaustive search. A set of 200 belief states was sampled such that $x \in \mathcal{X}$ and each $p_i \in [0, 1]$, $1 \leqslant i \leqslant |\mathcal{X}|$ were chosen uniformly at random. Table 5.3 shows as function of the search depth d the percentage of cases where the greedy policy coincided with the optimal policy. We see that as the length of the task increases, the greedy policy is increasingly rarely optimal.

To quantify the difference in expected value from executing either policy, we compared the value of an optimal action to the value of the greedy action

Table 5.3: Percentage of cases (out of 200 total) where the optimal policy is the greedy policy as function of the search depth d.

	Optimal policy is greedy, %
$d = 2$	86.0%
$d = 3$	72.0%
$d = 4$	67.0%
$d = 5$	62.5%
$d = 6$	61.5%

Table 5.4: The average and worst-case performance loss of the greedy policy compared to the optimal solution as function of the search depth d.

	Mean loss	Maximum loss
$d = 2$	0.0342	0.1213
$d = 3$	0.0365	0.2553
$d = 4$	0.0422	0.3337
$d = 5$	0.0518	0.3424
$d = 6$	0.0644	0.3593

when the two were not in agreement. The difference between the two values is the performance loss when the greedy policy is followed instead of selecting an optimal action. Table 5.4 shows the mean loss along with the maximum loss observed among the 200 belief states as function of the search depth d. We note that both the average and maximum losses increase as function of d.

We then sampled another set of 1000 random initial belief states $b_0 = (x, p(y))$ for which we computed the optimal action over a search depth d of 3 to 6 decision epochs applying the RTBSS algorithm. An upper and lower bound on the optimal value function are required for the algorithm. The upper bound was found either using the universal or d-step sensor relaxations P_u or $P_d(x)$, respectively. Theorem 4.17 with forward simulation was applied to find the upper bound value. The lower bound for the optimal value function was obtained by computing the value of a finite horizon greedy policy.

As the MAB conditions of Theorem 4.12 were satisfied, we found the optimal policy applying RTBSS in all cases. To study the efficiency of RTBSS and the MAB upper bounds, we compare the number of search tree nodes expanded by RTBSS to the number of nodes expanded by an exhaustive search. The number of visited nodes in the search tree for each of the 1000 belief states is shown in Figure 5.4 for both upper bounds and all search depths. Since the bound from the d-step sensor relaxation is tighter, applying it results in a lower or equal number of visited nodes than the universal sensor relaxation. For comparison, the average number of visited nodes for the exhaustive search is shown in Table 5.5. We note that applying either bound greatly reduces the number of visited nodes, in some cases by up to an order of magnitude.

We studied the performance of POMCP on the same problem with the same

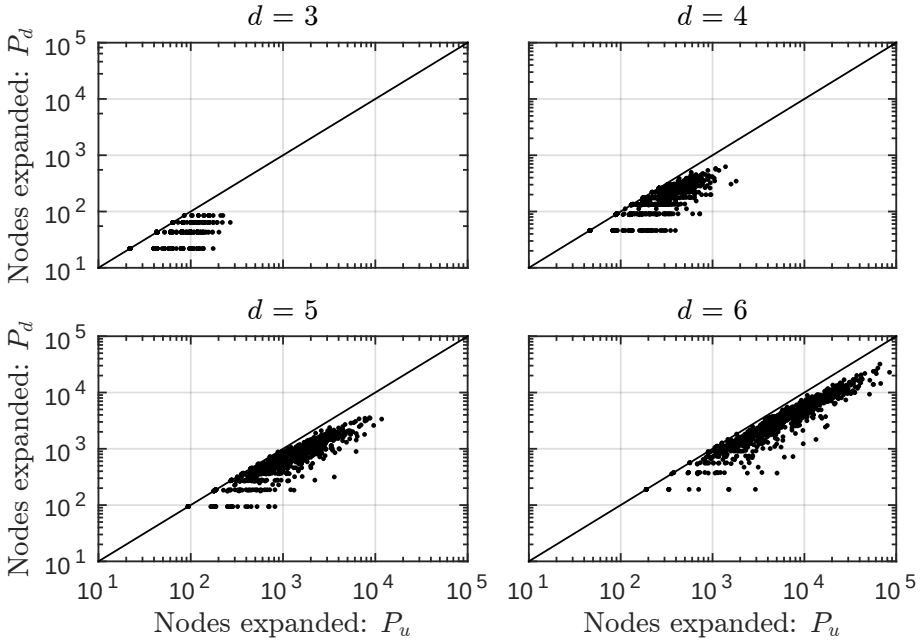


Figure 5.4: The number of nodes expanded in the RTBSS search tree applying either the universal sensor relaxation P_u or the d-step sensor relaxation $P_d(x)$ to find an upper bound. Results in each subfigure correspond to a different search depth d. Each point corresponds to the size of a search tree for one of the 1000 randomly sampled belief states $b = (x, p(y))$. The solid line shows where the two values are equal. Figure adapted from Lauri and Ritala (2015a).

Table 5.5: Average number of nodes expanded by exhaustive search as function of the search depth d. Table adapted from Lauri and Ritala (2015a).

	$d = 3$	$d = 4$	$d = 5$	$d = 6$
Nodes	$4.0 \cdot 10^2$	$3.8 \cdot 10^3$	$3.6 \cdot 10^4$	$3.5 \cdot 10^5$

Table 5.6: Percentage of POMCP recommendations agreeing with optimal as function of the simulation depth d and number of simulations N_s. Table adapted from Lauri and Ritala (2015a).

N_s	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
10^1	40.7%	36.8%	31.3 %	28.1%	29.2%
10^2	57.4%	50.1%	45.3%	43.2%	40.1%
10^3	69.5%	62.2%	54.0%	47.6%	46.8%
10^4	75.0%	74.9%	69.1%	63.5%	59.1%

belief states. For reference, we computed the optimal solution for each belief state by an exhaustive search. Table 5.6 shows the percentage of belief states for which the POMCP action recommendation coincided with the optimal action. As expected, the percentage increases as a function of the number of simulations N_s and tends to be lower for greater simulation depths d. It is also illustrative to compare these percentages with the corresponding percentages for the greedy policy, shown in Table 5.3. We note that to outperform the greedy policy, POMCP needs at least 10^4 simulations.

Table 5.7: The average and worst-case performance loss of POMCP compared to the optimal solution as function of the simulation depth d and number of simulations N_s. Table adapted from Lauri and Ritala (2015a).

	$d = 2$		$d = 3$		$d = 4$		$d = 5$		$d = 6$	
N_s	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max
10^1	0.0749	0.3930	0.0948	0.5278	0.1061	0.5283	0.1084	0.5587	0.1175	0.6278
10^2	0.0298	0.2296	0.0518	0.3299	0.0584	0.2953	0.0642	0.3251	0.0670	0.3968
10^3	0.0122	0.1025	0.0294	0.1751	0.0402	0.2035	0.0487	0.3639	0.0518	0.2839
10^4	0.0064	0.0503	0.0168	0.0893	0.0261	0.1399	0.0317	0.1816	0.0380	0.2020

We likewise compared the value of an optimal action to that recommended by POMCP when the two were not in agreement. The difference between the two values is the expected performance loss when the POMCP recommendation is followed instead of selecting an optimal action. We computed the average values and worst-case maximum values for the performance loss, shown in Table 5.7. The loss tends to be greater for fewer simulations N_s and a greater simulation depth d. When the number of simulations N_s is increased, the average performance loss for POMCP is low, indicating good average performance compared to the optimal solution. However, a typical feature of MC algorithms is that they do not provide worst case guarantees on performance. This is seen from the table as well: even if the average performance loss is low, the worst case loss may be significantly greater. Comparing to data from Table 5.4 for the greedy policy show that at least in the order of 10^4 simulations are needed for POMCP to achieve lower average and worst-case loss.

We conclude that in problems where suboptimal actions may lead to unacceptable performance loss, methods such as RTBSS with valid upper bounds may be preferable to POMCP.

5.2.3 MAB conditions violated

We next considered the case where the requirements of Theorem 4.12 are violated. Nevertheless, it is possible to approximate the problem as a POMDP MAB and derive the upper bounds as earlier. However, the upper bounds are not guaranteed to be valid – they are merely heuristics.

We set $r_i = w_i$ for each i. We considered three cases distinguished by the rate of the state transitions in w_i: slow, medium, or fast. For slow dynamics, the parameters were sampled for each i uniformly at random such that $p_{01}^{i,slow} \in [0.0, 0.2], p_{11}^{i,slow} \in [0.8, 1.0]$, for medium dynamics $p_{01}^{i,med} \in [0.2, 0.4], p_{11}^{i,med} \in [0.6, 0.8]$, and for fast dynamics $p_{01}^{i,fast} \in [0.4, 0.6], p_{11}^{i,fast} \in [0.4, 0.6]$. Each experiment was again repeated for 1000 randomly sampled initial belief states and dynamics models.

In this problem, POMCP performance was also observed to be good on average, with a low performance loss. Since the MAB conditions are not satisfied, the upper bounds found via the index policy solution of the relaxed problems are not in general valid. Hence, optimality for RTBSS cannot be guaranteed.

Table 5.8 shows the percentage of solutions equal to the optimal solution in case of slow, medium, or fast dynamics. Both the case where the upper bound was computed via the universal sensor relaxation P_u or the d-step sensor relaxation P_d is shown, varying the maximum search depth from $d = 2$ to $d = 6$. Optimal solutions are found in the majority of cases, with the percentage decreasing as the search depth is greater and the rate of dynamics faster. As the value of the bound from the d-step sensor relaxation is lower than the value of the bound from the universal sensor relaxation, it is more likely that the universal sensor relaxation bound does not overestimate the optimal value. Consequently, better agreement with the optimal solution is observed for P_u. The results suggest that in some problem domains it may still be reasonable to approximate upper bounds for the optimal value

Table 5.8: Percentage of RTBSS solutions agreeing with optimal solution as function of the search depth d when the MAB conditions were not satisfied. Results are shown for the slow, medium and fast dynamics cases. Table adapted from Lauri and Ritala (2015a).

Dynamics	Bound	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
Slow	P_u	100%	100%	100%	100%	100%
	P_d	100%	99.9%	99.9%	99.6%	99.6%
Medium	P_u	100%	100%	99.9%	100%	100%
	P_d	100%	99.8%	99.7%	99.1%	98.8%
Fast	P_u	100%	100%	100%	100%	100%
	P_d	100%	99.9%	99.7%	99.0%	98.9%

function by the value of index policies in the relaxed problems, even if optimality cannot be guaranteed.

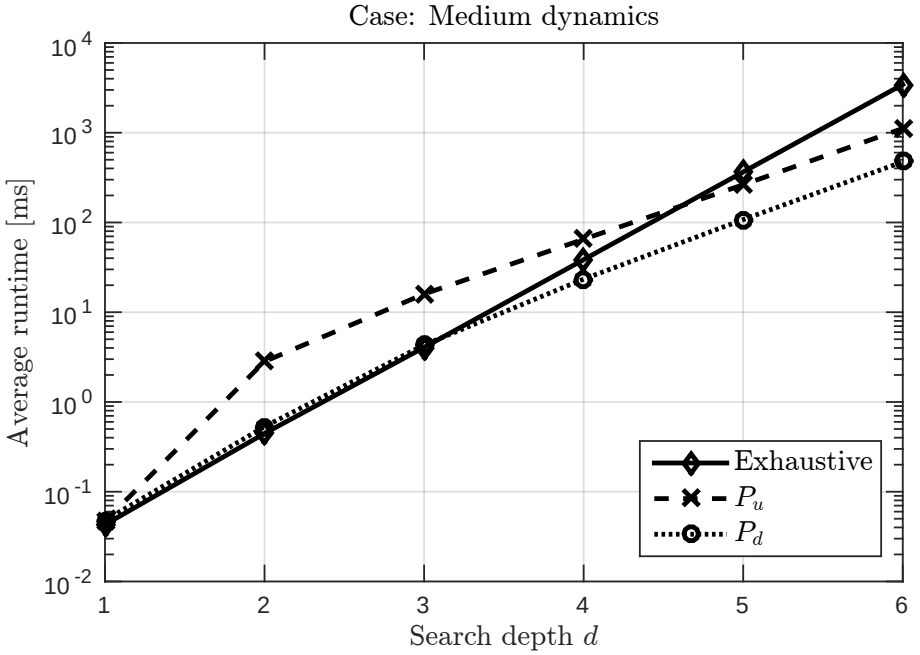


Figure 5.5: The mean runtime per decision in milliseconds as a function of the search depth d for the exhaustive search and branch-and-bound search applying upper bounds either from the universal sensor relaxation P_u or the d-step sensor relaxation P_d. Figure adapted from Lauri and Ritala (2015a).

The pruning effect on the size of the search tree was not affected significantly compared to the previous subsection. Applying either bound greatly reduced the number of expanded nodes in the search tree. Although the reduction in the number of visited nodes is substantial, evaluating the bounds has a computational cost that must be balanced with the savings from visiting fewer nodes. A representative comparison is shown in Figure 5.5 for the case of medium dynamics, showing the mean computation times per decision as function of the search depth d for the exhaustive search and applying either of the two relaxation bounds. At search depths $d < 3$, exhaustive search to find the optimal solution is still the fastest as the computational burden of

computing the bounds outweighs the potential savings from visiting fewer nodes. The advantages of pruning are seen for $d \geqslant 4$. At best, with pruning the computation time is an order of magnitude faster than exhaustive search. For $d \geqslant 3$, the upper bound from the m-step sensor relaxation is fastest. Using the upper bound from the universal sensor relaxation is faster than exhaustive search for $d \geqslant 5$.

5.3 Task support

In this section, we consider the task support problem (Problem 3.6) defined on page 66. The POMDP is represented by GMMs, and the value iteration approach for continuous-state POMDPs presented by Porta et al. (2006) is applied to solve the task. The value iteration approach is compared against POMCP. The simulation experiments are carried out in a robotic domain likewise defined by Porta et al., modified by us to include a choice of measurement channel.

In Subsection 5.3.1, the definition of the problem is given. Subsection 5.3.2 studies the effect of the GMM component reduction methods reviewed in Appendix B on the solution of the problem. Subsection 5.3.3 considers the resulting policies and the value of information in the sensor management problem, and Finally, Subsection 5.3.4 compares point-based value iteration to the sampling-based POMCP algorithm for solving this problem.

5.3.1 Problem definition

We study the robotic domain originally presented in Porta et al. (2006). We have modified the domain to include a selection of measurement channel at each decision epoch. The domain features a robot in a corridor attempting to enter the correct one among the multiple available doors, while only being able to partially observe its own location.

The corridor traversed by the robot is modelled by the continuous interval $\mathcal{S} = [-22, 22] \subset \mathbb{R}$, and at each decision epoch the robot is located at $s \in \mathcal{S}$. The robot has at its disposal three actions on the process: move left, move right, or attempt to enter door at the current location. Additionally, the robot has a choice of two measurement actions: a low noise channel with high cost, or a high noise channel with a lower cost. Thus $\mathcal{A}_b = \mathcal{A}_p \times \mathcal{A}_m$ $\forall b \in \mathcal{B}$, where $\mathcal{A}_p = \{a_{\text{left}}, a_{\text{right}}, a_{\text{enter}}\}$ are the three movement actions and $\mathcal{A}_m = \{a_{\text{low-noise}}, a_{\text{high-noise}}\}$ are the two measurement actions, leading to a total of six actions.

The action effects are modelled by a simple linear-Gaussian model

$$\mathbb{T}(s', a_p, s) = N(s'; s + \Delta(a_p), \Sigma^{a_p}), \tag{5.7}$$

where $a_p \in \mathcal{A}_p$ and $\Delta(a_p)$ is -2, 2, or 0 for moving left, right, or entering a door, respectively. We set $\Sigma^{a_p} = 0.05\, \forall a_p \in \mathcal{A}_p$.

The robot perceives one of the four observations in $\mathcal{Z} = \{$*left-end*, *right-end*, *door*, *corridor*$\}$ depending on its state. As noted by Porta et al. (2006), specifying the observation model directly as multinomial distributions over

$\mathcal{Z}$ for all $s' \in \mathcal{S}, a \in \mathcal{A}$ is not possible, so instead an indirect definition via

$$\mathbb{O}(z', s', a_m) \equiv p(z' \mid s', a_m) = \frac{p(s' \mid z', a_m)p(z')}{p(s', a_m)} \tag{5.8}$$

is applied. Here, $p(s' \mid z', a_m)$ is a GMM with 22 components. The means are placed every two state space units from -21 to 21. The five left- and right-most means correspond to the observations *left-end* and *right-end*, respectively. Means at -11, -5, 3 and 9 correspond to the observation *door*, while the rest of the Gaussians correspond to the observation *corridor*. If $a_m = a_{\text{low-noise}}$, all the components have a variance of 1.6. If $a_m = a_{\text{high-noise}}$, all the components have a variance of 8.

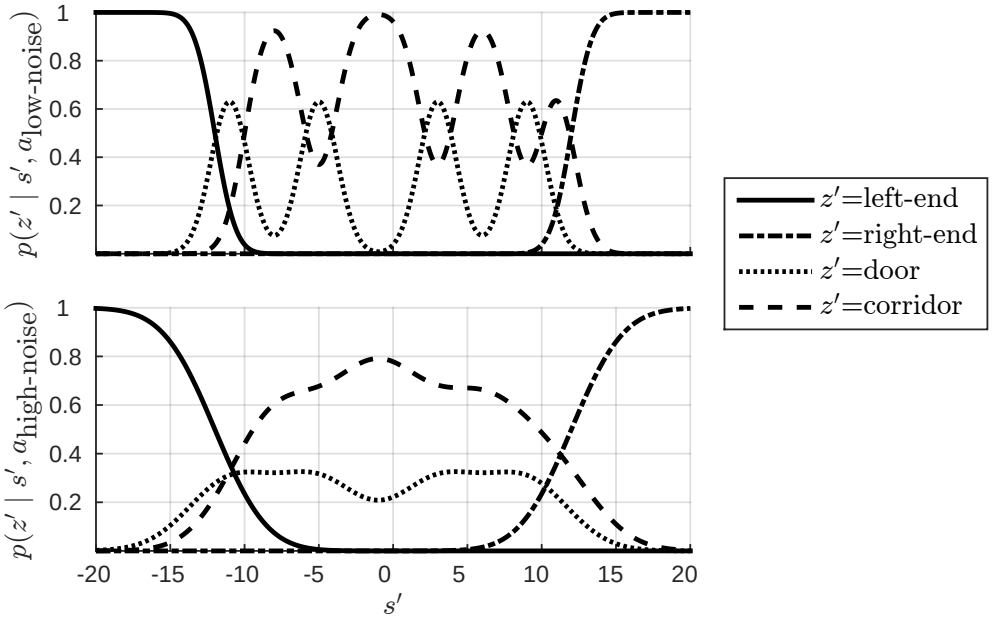


Figure 5.6: A comparison of the low noise (top) and high noise (bottom) observation models. Figure adapted from Lauri and Ritala (2013b).

As the means of the Gaussians are placed uniformly, it is assumed that $p(s', a_m)$ is uniform. It follows that $\mathbb{O}(z', s', a_m)$ is a GMM with the same means and covariances as $p(s' \mid z', a_m)$ but scaled by a factor determined by $p(z')$ and $p(s', a_m)$. Furthermore, it is required that for any $s \in \mathcal{S}, a_m \in \mathcal{A}_m$ $\sum_{i=1}^{|\mathcal{Z}|} p(Z = z_i | s', a_m) = 1$ to ensure that the probability for perceiving an observation is 1. A comparison between the low-noise and high-noise observation models is shown in Figure 5.6. The total probability of observations in the figure for any s' sums to one. The corridor and doors are harder to distinguish in case of the high noise sensing action.

The robot collects a positive reward when it enters the correct door, which is the second one from the right. The reward function is defined as $R(s, a) = R(s, a_p) + R(a_m)$, where $R(s, a_p)$ is a reward for the movement actions and $R(a_m)$ is a reward or cost related to the measurement actions. Both terms are represented as weighted sums of Gaussians.

For moving left or right, $R(s, a_p)$ has three Gaussians with variance 1 and weight -2. The means are at $\pm 21, \pm 19, \pm 17$, with positive signs for moving

right and negative signs for moving left. These Gaussians represent the negative reward for moving outside of the corridor. For entering the door, $R(s, a_p)$ has three Gaussians with means ± 25 and 3. The Gaussians at ± 25 have a variance of 250 and a weight of -10, representing the penalty for attempting to enter the door at the wrong location. The Gaussian at 3 has a variance of 3 and a weight of 2, representing the positive reward for entering the correct door.

The reward $R(a_{\text{high-noise}})$ for high-noise measurements is defined to be zero. To reflect the cost of more accurate measurements, $R(a_{\text{low-noise}})$ is set to a single Gaussian with variance 1000 and a weight of -0.1 to model uniform cost regardless of s.

5.3.2 Comparison of GMM component reduction methods

To solve the problem, we applied the point-based value iteration scheme of Porta et al. (2006). A set B_R of $N_b = 500$ belief states for the value iteration was generated applying Algorithm 4.1 (page 95). The initial belief state b_0 was approximately uniform, modelled by a GMM with four Gaussian components, with means at ± 5, ± 15, variances of 30, and weights of 0.25.

The value iteration algorithm was run for 10 repetitions, performing 50 value iteration updates each time. The maximum number of components in the α-functions was set to $N_a = 18$, and the three GMM component reduction methods presented in Appendix B were applied to compress any GMMs encountered having more than 18 components. The methods are denoted HE, ZK, and GR for the heuristic minimisation of $d_\infty(p, q)$, the upper bound $d_{\text{ZK}}(p, q)$ of the L^2 distance, and the KL divergence related $d_{\text{GR}}(p, q)$ between the original GMM p and its reduced-component approximation q, respectively.

We examined three quantities; the number of policy changes, the number N_{bs} of stationary belief states, and the number of α-functions; as function of the value iteration step. The number of policy changes at a given value iteration step is the number of belief points in B_R at which the optimal action changed. We call a belief state stationary at a given value iteration step if its optimal action does not change again until the final value iteration step. The number of α-functions characterises the complexity of the policy found, and is at most equal to $N_b = 500$. These quantities are shown on the three rows from top to bottom in Figure 5.7, while in each column a different component reduction method was applied.

We note from the top row that the number of policy changes remains at a level of approximately 100 changes per step, or one fifth of the belief states, until the end of the value iteration. This may be indicative of problems converging to an optimal action for some belief states. We noted that the switching is primarily due to switching between the two measurement actions while applying the same process actions a_p. The switching is due to the small difference in the reward functions for the measurement actions, making it difficult to reliably distinct between the related α-functions. When we purged the measurement switching phenomenon from the results of the top row of Figure 5.7, we obtained results as shown in Figure 5.8 which are

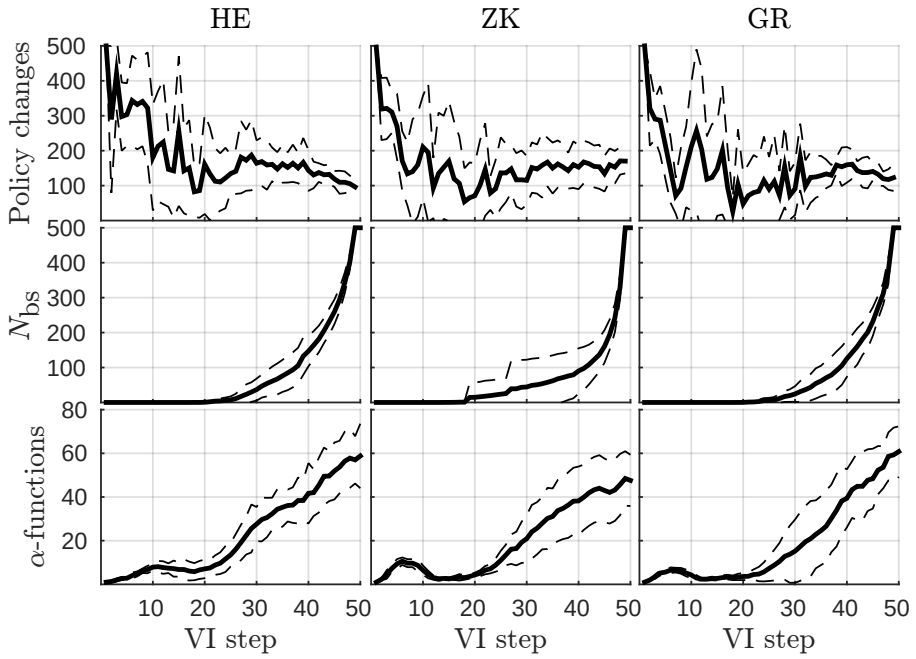


Figure 5.7: Policy changes (top row), number of stationary belief states (middle row), and α-functions (bottom row) as function of value iteration step. The columns from left to right show results when the HE, ZK, or GR GMM component reduction method was applied. Means are shown by solid lines and standard deviations by dashed lines. Figure adapted from Lauri and Ritala (2013b).

in agreement with similar data from the variant of the problem without measurement options (Lauri and Ritala, 2013b; Porta et al., 2006).

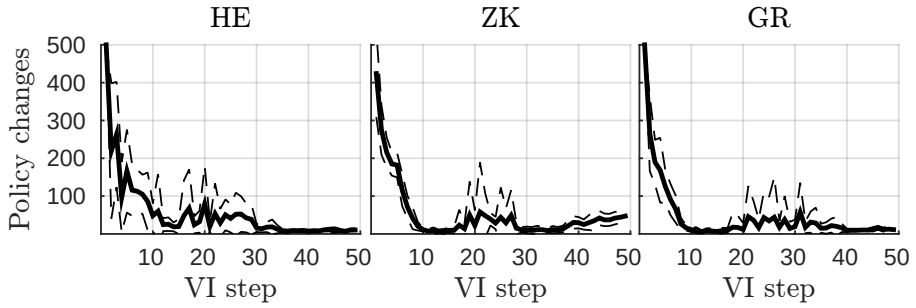


Figure 5.8: Policy changes when the effect of measurement switching was purged. The columns indicate which GMM component reduction method was applied; from left to right: HE, ZK, or GR. Means are shown by solid lines and standard deviations by dashed lines. Figure adapted from Lauri and Ritala (2013b).

The HE simplification method achieves the lowest average number of policy changes, followed closely by the GR method. Both have on average less than 20 changes in the policy on value iteration steps from 30 up to 50. The ZK method has on average 40 policy changes during this stage. The somewhat poorer performance of the ZK method may be explained by the fact that it requires that the α-function be separated into two mixtures, one with positive and the other with negative weights which are approximated separately and then combined again. During initial value iteration steps, the α-functions have only components with negative weights. As positive weights start to appear after 25 value iteration steps, the performance of the

ZK method diverges from HE and GR. Similar conclusions may be drawn from the middle row of Figure 5.7, showing slower growth in the number of stationary belief states for which the optimal action remains the same for the ZK method.

The number of α-functions shown on the bottom row of Figure 5.7 are similar for the HE and ZK methods but greater for the GR method at the end of the value iteration steps. The difference may be explained by the type of divergence quantification used by each method: whereas the HE and ZK methods are designed specifically for simplification of mixture models with arbitrary weights, the KL-divergence related quantification applied by the GR method is only natural for PDFs, i.e. GMMs with strictly positive weights summing to one.

Empirically, the policies obtained applying any of the GMM component reduction methods complete the required task, obtaining a positive sum of discounted rewards. This was verified by numerical simulation, repeating 10 times for each of the 10 policies at each value iteration step a simulation run consisting of 30 decision epochs. The results of the simulations are shown in Figure 5.9.

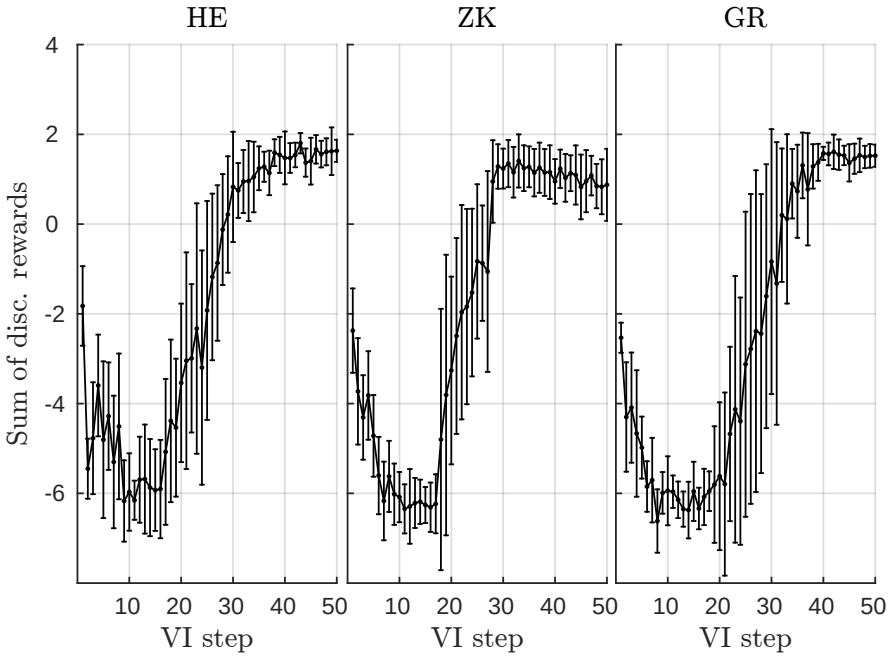


Figure 5.9: The sum of discounted rewards averaged over 10 simulations applying the policy obtained at each value iteration step. Results are shown for the HE (left), ZK (middle), and GR (right) GMM simplification methods. Means are shown by a solid line and standard deviations by the vertical bars. Figure adapted from Lauri and Ritala (2013b).

The mean cumulative discounted reward accumulated in the 10 repetitions and its 95% confidence interval are summarised in Table 5.9. We note that the HE and GR GMM simplification methods perform equally well, while the ZK method has a statistically significantly lower mean cumulative discounted reward. The likely reason is the splitting of α-functions into positive and negative sub-mixtures and approximating them separately, which causes loss of accuracy in the approximation and leads to inaccurate estimates of the

Table 5.9: Mean of the cumulative discounted reward and its 95% confidence interval applying the policy obtained from point-based value iteration after 50 value iteration steps, with each of the three GMM simplification methods.

HE	ZK	GR
1.63 ± 0.15	0.88 ± 0.50	1.52 ± 0.15

α-functions. Likewise, the variability in the mean cumulative discounted reward is significantly greater for the ZK method as indicated by the greater 95% confidence interval values.

5.3.3 Policies and the value of information

Visualisation of the policies obtained is difficult for belief states which are GMMs with multiple components. Instead, to illustrate the robot's behaviour in the task, we extracted the optimal action for a set of Gaussian belief states with means $\mu \in [-15, 15]$ and variances $\sigma^2 \in [0.1, 10]$. This provides a useful visualisation of the policy, with the weakness that policies for multi-modal belief states cannot be explored.

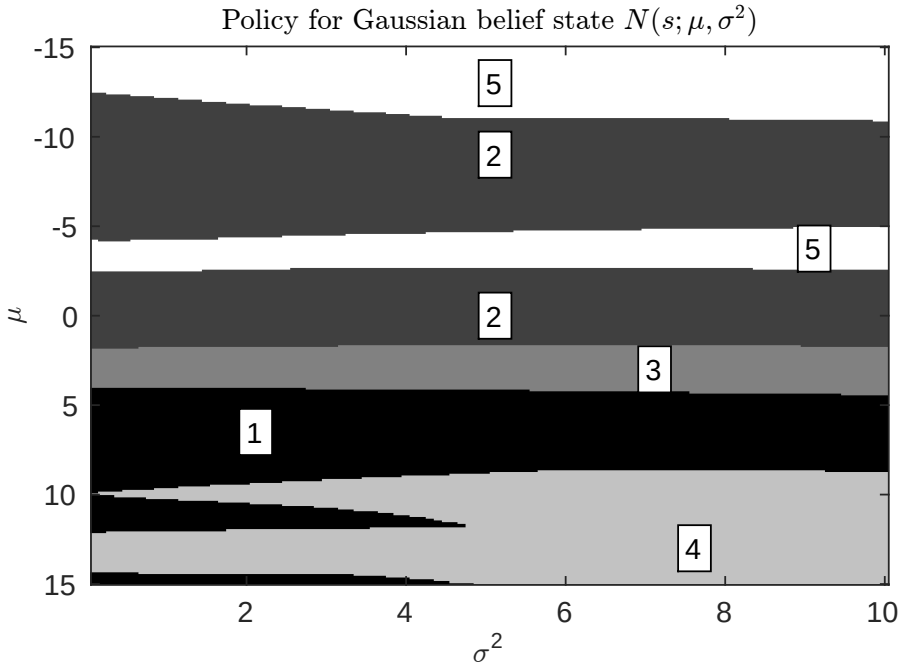


Figure 5.10: An example policy for a belief state consisting of a single Gaussian with mean μ and variance σ^2. The numerical value labels on the image indicate actions. For 1, 2, and 3, a_p is "move left", "move right", or "open door", respectively, with $a_m = a_{\text{high-noise}}$, while for 4, 5, and 6, a_p is ordered as before but $a_m = a_{\text{low-noise}}$. Figure adapted from Lauri and Ritala (2013b).

An example policy is shown in Figure 5.10. Near the edges of the state space at the upper and lower edges of the μ-axis the agent prefers to move towards the centre of the state space while applying a low noise measurement. This indicates that after the agent has successfully identified at which end of the corridor it is, it seeks to improve information on its location to find and enter the door at location 3. Door opening actions with a high-noise

measurement, labelled by the number 3, are preferred in the vicinity of the correct door at location 3.

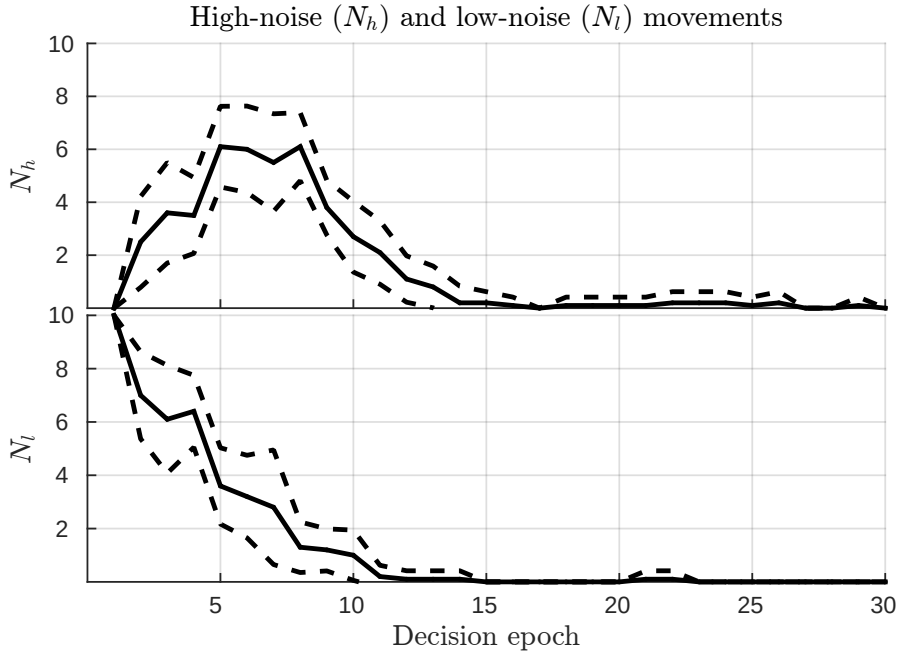


Figure 5.11: The number of movement actions with a high-noise (top) or low noise (bottom) measurement. Means are shown by solid lines and standard deviations by dashed lines. Figure adapted from Lauri and Ritala (2013b).

The overall strategy adopted by the robot is illustrated in Figure 5.11. The figure shows the numbers N_h and N_l of movement actions with a high-noise or low-noise measurement, respectively, as function of the decision epoch. The robot first moves to either direction while observing the low-noise measurement channel. Once the robot's information regarding its position is sufficiently accurate, the proportion of movement actions with a low noise measurements decreases as the robot traverses towards the correct door. Finally, the robot reaches the correct door typically after 10-15 decision epochs and enters it, as indicated by the sum of N_h and N_l falling close to zero indicating no movement actions.

The value functional for a set of Gaussian belief states with means $\mu \in [-15, 15]$ and variances $\sigma^2 \in [0.1, 10]$ is shown in Figure 5.12. As expected, the high value beliefs are those where the state is near the correct door to enter at state space unit 3 with high probability (small variance). The sinusoid-like behaviour with respect to the mean μ is due to the discrete expected step size $\Delta(a_p)$ per action. For example, for $\Delta(a_{\text{left}}) = -2$, it is more valuable to be near state space unit 5 than 4, since the correct door is more likely reached from the former.

The utility of selecting either measurement channel consists of the explicitly determined difference in the costs of the measurements, as defined in the reward function, together with the implicitly defined value of information. The value of information is the difference in expected future rewards between the posterior belief states resulting from either measurement choice. When the benefits gained from a more accurate measurement outweigh the increased cost, a low-noise measurement will be chosen.

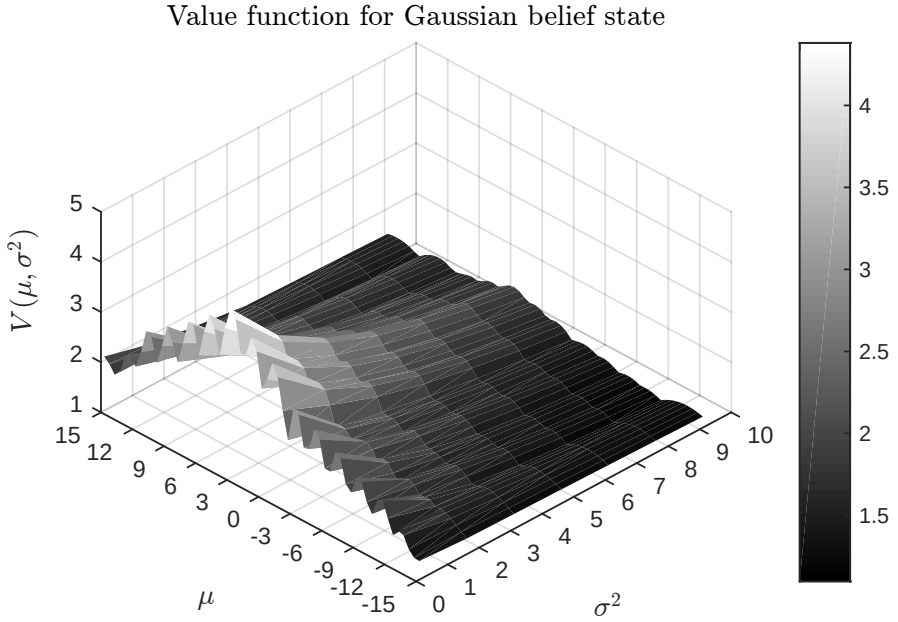


Figure 5.12: An example value function for a belief state consisting of a single Gaussian with mean μ and variance σ^2. Figure adapted from Lauri and Ritala (2013b).

5.3.4 Comparison with sampling-based planning

According to the comparison of solution algorithms presented in Table 4.3 on page 74, online and sampling based planners may be an good alternative to point-based value iteration in the task support problem. However, there is no apparent way of computing bounds on the optimal value function or an effective heuristic in this problem. Given that the problem features a continuous state space but a finite action and observation space, MCTS type algorithms are a suitable choice. We implemented the POMCP algorithm[2] for the task support problem to compare it against PBVI.

We applied either $N_s = 100$, 1000, 10000, or 50000 simulations in POMCP, with a maximum depth of $d = 3$, 4, 5, 6, or 7 decision epochs. Similarly to the value iteration algorithms, the experiment was repeated 10 times with each set of parameters with at most 30 decision epochs in each repetition. The cumulative rewards were recorded and are summarised in Table 5.10. For $d = 6$ and $d = 7$, results are not listed as the computation time required per decision exceeded the threshold of 1500 s (see Table 5.11).

We compared the data of Table 5.10 with Table 5.9 showing the same results for PBVI. We note that POMCP can reach comparable performance in the mean cumulative discounted reward when $4 \leqslant d \leqslant 5$ and $N_s \geqslant 50000$ or when $6 \leqslant d \leqslant 7$ and $N_s \geqslant 10000$. However, POMCP does not reach the same level of uniformity in the mean cumulative discounted reward as PBVI, as indicated by the significantly greater confidence intervals for POMCP. Although POMCP does not provide any worst case performance guarantees with respect to the reward, a greater number of simulations N_s or a greater maximum simulation depth d will result in POMCP being more likely to find a better approximation to the optimal solution (Silver and Veness, 2010).

[2]See algorithms 2.4, 2.5, and 2.6 starting at page 38.

Table 5.10: Mean of cumulative discounted reward and its 95% confidence interval for POMCP with N_s simulations and maximum simulation depth of d decision epochs.

N_s	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$
10^2	0.58 ± 0.86	0.30 ± 0.57	0.42 ± 0.71	1.56 ± 0.57	1.08 ± 0.68
10^3	0.08 ± 0.91	0.76 ± 1.00	0.71 ± 0.77	1.10 ± 0.68	0.79 ± 0.96
10^4	1.57 ± 0.76	0.64 ± 1.13	0.97 ± 1.09	1.45 ± 0.78	1.52 ± 0.59
$5 \cdot 10^4$	1.49 ± 1.06	1.92 ± 0.60	1.85 ± 1.03	-	-

Table 5.11: Mean planning time per decision in seconds with the POMCP algorithm with N_s simulations and maximum simulation depth of d decision epochs.

N_s	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$
10^2	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
10^3	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
10^4	71 s	95 s	103 s	101 s	108 s
$5 \cdot 10^4$	292 s	455 s	690 s	>1500 s	>1500 s

However, as indicated in part by the results of our comparison, sufficiently great values for these parameters are in general hard to determine and numerical experimentation is often required.

5.4 Robotic exploration

In this section, the problem of autonomous robotic exploration (Problem 3.7) is considered. A robot is traversing an a priori unknown or only partially known environment with the objective of collecting the greatest possible amount of information regarding the map and its own pose.

In the first case studied in Subsection 5.4.1, a robot is traversing along a fixed trajectory while the operation of a camera system with one degree of freedom is optimised (Lauri and Ritala, 2014). A multivariate Gaussian belief state is assumed. The simulated problem is solved applying the POMCP algorithm.

The second case in Subsection 5.4.2 concerns an exploration task in an a priori unknown environment (Lauri and Ritala, 2015b). The robot's objective is to choose its trajectory in a manner that provides the most information regarding the environment. The belief state in this case is non-Gaussian. The problem is studied in simulations applying the POMCP and SMC-OLFC algorithms. Finally, a real-world implementation is presented and demonstrated.

5.4.1 Operating a camera system

Consider a robot traversing a planar environment, equipped with a camera system that can be rotated independently of the robot's motion to focus attention in a different direction. The robot's motion controls are decided

elsewhere, while the camera system is to be controlled. This problem may be thus viewed as an instance of the robotic exploration problem.

The robot's state x_t at decision epoch t consists of its location defined by x_t^r and y_t^r, its heading angle θ_t^r, and the orientation angle ϕ_t^r of the vision system. The superscript r emphasises that the variables are related to the robot. Thus a tuple $x_t = (x_t^r, y_t^r, \theta_t^r, \phi_t^r)$ fully describes the robot's configuration at decision epoch t.

The robot's motion is controlled by applying a translational velocity v_t and rotational velocity ω_t. Let $\mu_t = (v_t, \omega_t)$ denote the control vector. The PDF is Gaussian $N(\mu_t; \hat{\mu}_t, Q_t)$, where the mean $\hat{\mu}_t$ is the nominal desired control input, and $Q_t = \mathrm{diag}(\sigma_v^2, \sigma_\omega^2)$ is a diagonal noise covariance matrix. The vision system's orientation angle ϕ_t^r with respect to the robot's body can be controlled independently by choosing its desired rotational velocity $a_t \in \mathcal{A} = [a_{\min}, a_{\min}]$, constrained to a range between a minimum and maximum rotational velocity. We assume $\hat{\mu}_t$ given, while the robot has to choose a_t.

The robot's motion model $\mathbb{T}_x(x', a, x)$ is stochastic and consists of two parts. For $(x_t^r, y_t^r, \theta_t^r)$ we apply a non-linear motion model from (Thrun et al., 2006, Chap. 5) while a linear model is applied for ϕ_t^r:

$$x_{t+1} = f_v(x_t, \mu_t, a_t) = x_t + \begin{bmatrix} -\frac{v_t}{\omega_t}\sin\theta_t^r + \frac{v_t}{\omega_t}\sin(\theta_t^r + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta_t^r - \frac{v_t}{\omega_t}\cos(\theta_t^r + \omega_t\Delta t) \\ \omega_t\Delta t \\ (\omega_t + a_t)\Delta t \end{bmatrix}, \tag{5.9}$$

where Δt denotes the time discretisation parameter such that each decision epoch covers a real-valued time interval $[t, t + \Delta t]$. We assume the noise for rotating the vision system to be negligible with respect to the other sources of uncertainty, and thus assume the vision system motion model noiseless.

The environment the robot is traversing consists of a known number n of stationary features that the robot can perceive with its camera. Let $m = [x_f^1, y_f^1, \ldots, x_f^n, y_f^n]^{\mathrm{T}}$ denote the vector of feature locations. The vector m can be thought of as a map of the environment. Since m is stationary, the map state transition model $\mathbb{T}_m(m', a, m)$ is an identity mapping, independent of $a \in \mathcal{A}$.

The complete state in the problem is a pair $s = (x, m)$ consisting of the robot state and environment state. The nominal dynamics of the whole system are described by a function

$$s_{t+1} = f(s_t, \mu_t, a_t) = \begin{bmatrix} f_v(x_t, \mu_t, a_t) \\ m \end{bmatrix}. \tag{5.10}$$

The belief over the robot state and map state is assumed multivariate Gaussian, $b = N(s; \hat{s}, S)$, with mean $\hat{s}$ and covariance S.

The robot can perceive features that are within the cone of observation $D(x_t, \alpha, r_{\max})$, determined by the vision system's view angle α and the maximum observable range $r_{\max}$. The cone of observation with α =60° and $r_{\max}$ =6 m is illustrated in Figure 5.13. We denote by $W = \{1, 2, \ldots, n\}$ the set of all features, and by $\tilde{W}(s_t) = \{i \in W \mid (x_f^i, y_f^i) \in D(x_t, \alpha, r_{\max})\}$ the subset of features observed in state s_t.

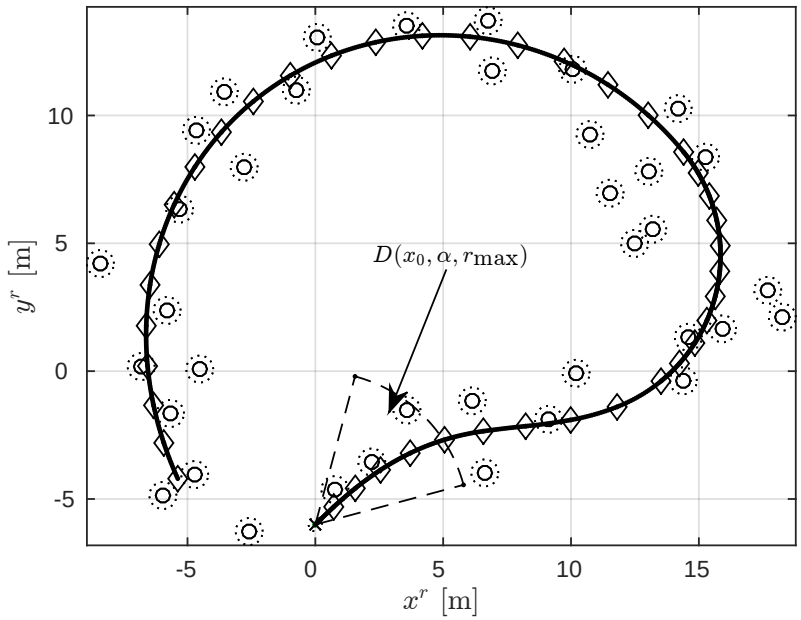


Figure 5.13: An example of a robot trajectory, map and the observation cone. The solid line denotes the expected trajectory of the robot. Diamond markers denote the locations at which control actions are decided. The circle markers denote the features in the map, and the associated dashed circles represent the 50% confidence intervals in the initial belief state. Features within the cone of observation $D(x_0, \alpha, r_{\max})$ limited by dashed lines may be observed. Figure adapted from Lauri and Ritala (2014).

An observation $z_{t,i}$ of the ith feature consists of the measured distance d_i and angle β_i of the feature relative to the robot's state. In practice, such observations could be obtained e.g. from a stereo camera system or a monocular camera system observing targets with known dimensions. We assume known data association, i.e. it is known from which feature a particular observation originates from. The observations are affected by additive zero-mean Gaussian noise with independent variances σ_d^2 and σ_β^2, respectively. The non-linear observation model for feature i may be written as $z_{t,i} = h_i(s_t) + r_t^i$, where

$$h_i(s_t) = \begin{bmatrix} d_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \sqrt{(x_t^r - x_f^i)^2 + (y_t^r - y_f^i)^2} \\ \arctan \frac{y_t^r - y_f^i}{x_t^r - x_f^i} - \phi_t^r - \theta_t^r \end{bmatrix}, \tag{5.11}$$

and r_t^i is independent zero-mean Gaussian noise with covariance $R_t^i = \mathrm{diag}(\sigma_d^2, \sigma_\beta^2)$. The complete observation z_t at time t is now the vector of all individual feature observations. The observation model h for the whole system is then given by

$$z_t = h(s_t) + r_t = \begin{bmatrix} h_1(s_t) + r_t^1 \\ \vdots \\ h_n(s_t) + r_t^n \end{bmatrix}, \tag{5.12}$$

where r_t^i are i.i.d. and correspondingly r_t is zero-mean Gaussian noise with covariance matrix R_t equal to the block diagonal matrix of all R_t^i.

State estimation. The belief update equation in the problem is implemented via an extended Kalman filter (EKF) (see e.g. Särkkä, 2013, Ch. 5). The EKF handles non-linear state transition and observation models by linearisation at means of the belief states. Maintaining the belief state via the EKF solves the SLAM problem in this instance (see e.g. Durrant-Whyte and Bailey, 2006).

Our EKF implementation works as follows. Given the current belief state $b_t = N(s_t; \hat{s}_t, S_t)$ and the actions $\hat{\mu}_t$, a_t, a predictive PDF $N(s_{t+1}; \hat{s}^+_{t+1}, S^+_{t+1})$ is computed with

$$\hat{s}^+_{t+1} = f(\hat{s}_t, \hat{\mu}_t, a_t) \tag{5.13a}$$

$$S^+_{t+1} = F_s S_t F_s^{\mathrm{T}} + F_\mu Q_t F_\mu^{\mathrm{T}}, \tag{5.13b}$$

where $F_{\{s,\mu\}} \equiv F_{\{s,\mu\}}(\hat{s}_t, \hat{\mu}_t, a_t)$ denote the Jacobians of the dynamic model f with respect to the state s_t and control μ_t, respectively, evaluated at $(\hat{s}_t, \hat{\mu}_t, a_t)$.

Once the measurement z_{t+1} is obtained, the posterior PDF is calculated in the update step as $b_{t+1} = N(s_{t+1}; \hat{s}_{t+1}, S_{t+1})$ with

$$v_{t+1} = z_{t+1} - h(\hat{s}^+_{t+1}) \tag{5.14a}$$

$$L_{t+1} = H_s S^+_{t+1} H_s^{\mathrm{T}} + R_{t+1} \tag{5.14b}$$

$$K_{t+1} = S^+_{t+1} H_s^{\mathrm{T}} L^{-1}_{t+1} \tag{5.14c}$$

$$\hat{s}_{t+1} = \hat{s}^+_{t+1} + K_{t+1} v_{t+1} \tag{5.14d}$$

$$S_{t+1} = S^+_{t+1} - K_{t+1} L_{t+1} K^{\mathrm{T}}_{t+1}, \tag{5.14e}$$

where $(\cdot)^{-1}$ denotes matrix inverse and $H_s \equiv H_s(\hat{s}_{t+1})$ is the Jacobian of h with respect to the state s_t evaluated at the predictive mean $\hat{s}^+_{t+1}$. Measurements $z_{t+1,i}$ for which $i \notin \tilde{W}(s_{t+1})$ should have no contribution to the posterior. This is achieved by setting for each $i \notin \tilde{W}(s_{t+1})$ the corresponding rows of H_s to zero.

Approximating mutual information. The MI between state and observation is equivalent to the expected KL divergence between the posterior and predictive PDFs p and q, taking the expectation w.r.t. Z_{t+1}. This is seen e.g. from Definition A.5 and applying rules of probability. The posterior PDF depends on the subset $\tilde{W}$ of features observed, which is a random variable. Taking the expectation over $\tilde{W}$ as well leads to

$$I(S; Z \mid a_t) = \mathbb{E}_{\tilde{W}} \left[\mathbb{E}_{Z_{t+1}} [D(p, q) \mid a_t] \right]. \tag{5.15}$$

We approximate (5.15) in two stages, first by approximating the inner expectation and the KL divergence, then by approximating the outer expectation over the subsets of features observed.

The expected KL divergence between the two Gaussians $p(x) = N(x; m_0, P_0)$ and $q(x) = N(x; m_1, P_1)$ is

$$D(p, q) = \frac{1}{2} \left[\mathrm{tr}(P_1^{-1} P_0) + (m_1 - m_0)^{\mathrm{T}} P_1^{-1} (m_1 - m_0) - l + \log \frac{|P_1|}{|P_0|} \right], \tag{5.16}$$

where $\mathrm{tr}(\cdot)$ is the trace operator, l is a constant equal to the dimensionality of x, and $|\cdot|$ denotes matrix determinant.

Let $q(s_{t+1}) = N(s_{t+1}; \hat{s}^+_{t+1}(a_t), S^+_{t+1})$ be the predictive PDF over the state. The predictive mean $\hat{s}^+_{t+1}(a_t)$ is a function of the camera control a_t applied. For a fixed subset $\tilde{W}$ of observed features, the posterior distribution may be found via equations (5.14a)-(5.14e). We note that the posterior mean depends on the actual observation z_{t+1}, while the posterior covariance S_{t+1} only depends on the subset $\tilde{W}$ of observed features. We thus denote the posterior distribution as $p(s_{t+1}) = N(s_{t+1}; \hat{s}_{t+1}(z_{t+1}), S^{\tilde{W}}_{t+1})$.

To evaluate the inner expectation of Equation (5.15), we take the expectation of Equation (5.16) with the posterior and predictive distributions p and q as above, respectively, assuming for now a fixed set $\tilde{W}$ of features observed. This yields

$$\begin{aligned}\mathbb{E}_{Z_{t+1}}[D(p,q) \mid a_t] = \frac{1}{2}\Big(&\operatorname{tr}\left(\left[S^+_{t+1}\right]^{-1} S^{\tilde{W}}_{t+1}\right) \\ &+ \mathbb{E}_{Z_{t+1}}[(\hat{s}^+_{t+1} - \hat{s}_{t+1}(z_{t+1}))^{\mathrm{T}} \left[S^+_{t+1}\right]^{-1} (\hat{s}^+_{t+1} - \hat{s}_{t+1}(z_{t+1}))] \\ &-l + \log\frac{|S^+_{t+1}|}{|S^{\tilde{W}}_{t+1}|}\Big)\end{aligned} \tag{5.17}$$

where only the quadratic term is dependent on Z_{t+1}. To evaluate the expectation of the quadratic term, we first determine by applying Equations (5.14) that

$$\begin{aligned}\mu_v &= \mathbb{E}_{Z_{t+1}}[\hat{s}^+_{t+1} - \hat{s}_{t+1}(z_{t+1})] \\ &= \mathbb{E}_{Z_{t+1}}[\hat{s}^+_{t+1} - (\hat{s}^+_{t+1} + K_{t+1}(z_{t+1} - h(\hat{s}^+_{t+1})] \\ &= -K_{t+1}(\underbrace{\mathbb{E}_{Z_{t+1}}[z_{t+1}]}_{=h(\hat{s}^+_{t+1})} - h(\hat{s}^+_{t+1})) = 0,\end{aligned} \tag{5.18}$$

and by definition of variance and since $\hat{s}^+_{t+1} - \hat{s}_{t+1}(z_{t+1}) = -K_{t+1}v_{t+1}$,

$$\begin{aligned}&\operatorname{Var}[\hat{s}^+_{t+1} - \hat{s}_{t+1}(z_{t+1})] = K_{t+1}\operatorname{Var}[v_{t+1}]K^{\mathrm{T}}_{t+1} \\ &= K_{t+1}L_{t+1}K^{\mathrm{T}}_{t+1}.\end{aligned} \tag{5.19}$$

The expectation of the quadratic term in Equation (5.17) can be calculated as

$$\begin{aligned}&\mathbb{E}_{Z_{t+1}}\left[(\hat{s}^+_{t+1} - \hat{s}_{t+1}(z_{t+1}))^{\mathrm{T}}[S^+_{t+1}]^{-1}(\hat{s}^+_{t+1} - \hat{s}_{t+1}(z_{t+1}))\right] \\ &\quad= \operatorname{tr}([S^+_{t+1}]^{-1}K_{t+1}L_{t+1}K^{\mathrm{T}}_{t+1}) + \mu_v^{\mathrm{T}}[S^+_{t+1}]^{-1}\mu_v \\ &\quad= \operatorname{tr}([S^+_{t+1}]^{-1}K_{t+1}L_{t+1}K^{\mathrm{T}}_{t+1}),\end{aligned} \tag{5.20}$$

the last equality following since $\mu_v = 0$. Plugging Equation (5.20) into Equation (5.17) yields the result

$$\begin{aligned}D^{\tilde{W}}_{\mathrm{KL}}(a_t) \equiv \mathbb{E}_{Z_{t+1}}[D(p,q) \mid a_t] = \frac{1}{2}\Big(&\operatorname{tr}\left(\left[S^+_{t+1}\right]^{-1} S^{\tilde{W}}_{t+1}\right) \\ &+\operatorname{tr}([S^+_{t+1}]^{-1}K_{t+1}L_{t+1}K^{\mathrm{T}}_{t+1}) - l + \log\frac{|S^+_{t+1}|}{|S^{\tilde{W}}_{t+1}|}\Big).\end{aligned} \tag{5.21}$$

Expanding the outer expectation of (5.15), we have

$$I(S; Z \mid a_t) = \sum_{\tilde{W} \in 2^W} D^{\tilde{W}}_{\mathrm{KL}}(a_t) P(\tilde{W}) \tag{5.22}$$

where the summation is over the power set 2^W, i.e. the set of all subsets of W. The term $P(\tilde{W})$ is the probability that the subset $\tilde{W}$ of features is observed. The size of the power set for n features is 2^n, an infeasible amount to enumerate to calculate (5.22) exactly. It is also difficult to estimate values for the probabilities $P(\tilde{W})$. We instead apply a MC approximation based on sampling possible subsets of observed features. A set of samples $\{s_{t+1}^{(i)}\}_{i=1}^{N} \sim N(s_{t+1}; \hat{s}_{t+1}^{+}(a_t), S_{t+1}^{+})$ is drawn from the state predictive distribution. For each sample we find the deterministic subset $\tilde{W}^{(i)} = \tilde{W}(s_{t+1}^{(i)})$ of features inside the cone of observation $D(s_{t+1}^{(i)}, \alpha, r_{\max})$. By Lemma 4.19 we obtain

$$I_N = \frac{1}{N} \sum_{i=1}^{N} D_{\mathrm{KL}}^{\tilde{W}^{(i)}}(a_t) \tag{5.23}$$

which converges to $I(S; Z \mid a_t)$ as $N \to \infty$.

Simulation of camera control policies. Suppose that a robot is following a trajectory defined as a sequence of T expected control inputs $\{\mu_t\}_{t=0}^{T-1}$, resulting in the trajectory shown by the solid line in Figure 5.13. The objective is to find a policy for selecting controls for the camera rotational velocity, a_t, such that the expected sum of mutual information between the state and observation is maximised.

While traversing the trajectory, we solved the controls a_t for the camera rotational velocity by the POMCP algorithm. As the observation space $\mathcal{Z}$ in the problem is continuous, we apply an open loop approximation together with the receding horizon control principle, i.e. an OLFC variant of POMCP. The difference to the POMCP algorithm presented in Subsection 2.2.4.5 is that the effect of observations is ignored, i.e. the search tree does not branch due to them. The algorithm finds the optimal open loop control trajectory $a_{t:t+d-1}^{*}$ over a search depth of d decision epochs. To evaluate the mutual information reward function, we apply Eq. (5.23). The continuous space $\mathcal{A}$ of possible camera rotational velocities was discretised uniformly to a finite set of 12 control values, such that $\mathcal{A} = \{a_1, a_2, \ldots, a_{12}\}$.

The simulation environment had $n = 37$ features to observe, while the robot traversed a trajectory of length $T = 40$. The initial belief state was a multivariate Gaussian with mean corresponding to the true values of both robot pose and feature locations, and with a diagonal covariance matrix with variance $\sigma_r^2 = 0.05\,\mathrm{m}^2$ for the robot position, $\sigma_\theta^2 = 0.05\,\mathrm{rad}^2$ for the robot's heading angle, and $\sigma_f^2 = 0.2\,\mathrm{m}^2$ for each feature coordinate. The camera angle was assumed known, and thus its variance in the initial belief state was set close to zero.

Decisions on camera control inputs are made once per second, and the camera outputs an observation at the same rate. The range of rotational values was $a_{\min} = -15\,^{\circ}\,\mathrm{s}^{-1}$ and $a_{\max} = 15\,^{\circ}\,\mathrm{s}^{-1}$, respectively. The noise parameters were $\sigma_v^2 = 1 \times 10^{-3}\,\mathrm{m}^2\,\mathrm{s}^{-2}$, $\sigma_\omega^2 = 1 \times 10^{-3}\,\mathrm{rad}^2\,\mathrm{s}^{-2}$, $\sigma_d^2 = 1 \times 10^{-3}\,\mathrm{m}^2$, $\sigma_\beta^2 = 2 \times 10^{-4}\,\mathrm{rad}^2$.

The camera system's parameters were $\alpha = 60^{\circ}$ and $r_{\max}$ was set to either 3 m, 6 m, or 9 m. We compared the OLFC-POMCP policy against a random policy picking a_t at each decision epoch uniformly at random from $\mathcal{A}$, and a

greedy policy selecting actions a_t such that (5.23) with $N = 50$ samples was maximised. For POMCP, the search depth d of the simulations was varied from 2 to 5 decision epochs. The experiment was repeated 8 times for each policy and set of parameters.

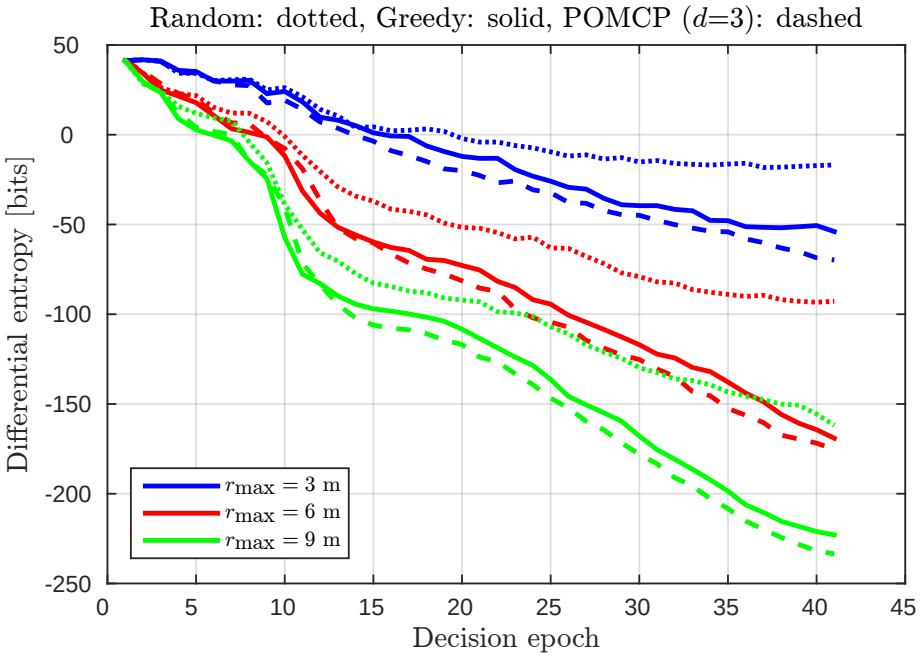


Figure 5.14: The mean differential entropy of the belief state as function of the number of camera observations for different maximum observable ranges $r_{\max}$ for the random, greedy and POMCP policy with $d = 3$.

The mean differential entropy of the belief state as function of the number of camera observations was examined. The lower the differential entropy is, the lower the uncertainty related to the belief state. Figure 5.14 shows the results for the random, greedy, and POMCP policy with $d = 3$. The colours indicate values for the maximum observable range $r_{\max}$ such that 3 m is shown in blue, 6 m in red, and 9 m in green. The dotted lines indicate results for the random policy, the solid line for the greedy policy, and the dashed line for the POMCP policy.

We first observe that the uncertainty of the state estimate decreases to a lower level and at a faster pace the greater the maximum observable range $r_{\max}$ is. The greater the maximum observable range is, the larger the area of the cone of observation $D(x_t, \alpha, r_{\max})$ is. In turn, this leads to a greater number of features detected per observation, leading to more informative measurements and hence a lower uncertainty.

We also observe that in some cases the POMCP seems to outperform the greedy policy. However, this is likely to be a random variation in the simulation, as when we examined the final differential entropy after 40 observations (Table 5.12) we noted that the difference between the greedy and POMCP policies is not statistically significant. In the table the lowest average entropies are written in boldface font for each value of $r_{\max}$.

POMCP seems to reach the lowest differential entropy. We compared the best values obtained with $d = 3$ to those of the greedy policy by using a two-sided t-test to test equivalence of the means in the samples. For

Table 5.12: Mean differential entropy after 40 observations and its ±95% confidence intervals for the random, greedy and POMCP policies.

Policy	$r_{\max} = 3\,\text{m}$	$r_{\max} = 6\,\text{m}$	$r_{\max} = 9\,\text{m}$
Random	-16.9 ± 16.5	-92.8 ± 28.6	-161.9 ± 37.1
Greedy	-53.9 ± 21.1	-169.1 ± 11.8	-222.9 ± 5.8
POMCP, $d = 2$	-58.6 ± 17.9	-173.6 ± 10.9	-229.4 ± 5.8
POMCP, $d = 3$	**-69.8 ± 12.5**	**-175.3 ± 17.2**	**-233.6 ± 6.8**
POMCP, $d = 4$	-32.7 ± 27.1	-167.4 ± 19.5	-229.8 ± 3.8
POMCP, $d = 5$	-52.3 ± 14.8	-148.2 ± 17.8	-212.4 ± 12.2

$r_{\max}$=3 m and $r_{\max}$=6 m, the null hypothesis that the means are equal could not be rejected at a 5% significance level. For $r_{\max}$=9 m, however, the null hypothesis could be rejected (p-value 0.035), showing that POMCP performs better than the greedy policy with a 5% significance level. We finally note that in all cases both the greedy and POMCP policies clearly outperform random action selection, leading to a lower differential entropy.

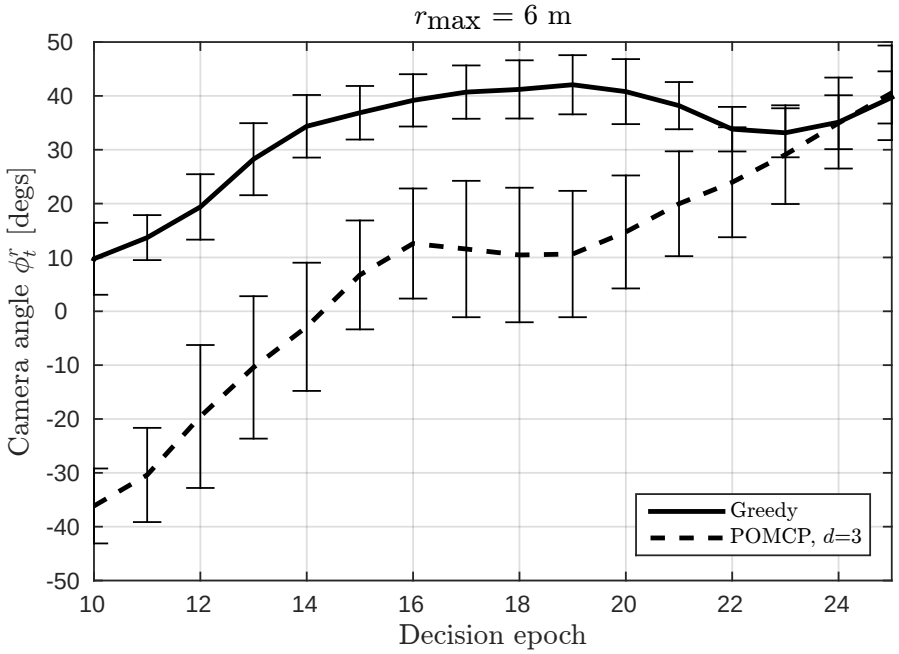


Figure 5.15: Mean camera angle ϕ_t^r and its 95% confidence intervals (vertical bars) between 10 and 25 decision epochs for the greedy policy and POMCP policy with $d = 3$.

We look more closely at the differential entropy between the range of 10 and 25 decision epochs from the data of Lauri and Ritala (2014) for $r_{\max}$=6 m. These decision epochs correspond to the part of the trajectory shown on the lower right hand side in Figure 5.13.

Figure 5.15 shows the camera angle ϕ_t^r during the same decision epochs. From the figure we see that the greedy policy is observing features on the right hand side of the trajectory around x-coordinate greater than 15, and orients the camera towards them. In contrast, the POMCP policy due to the non-myopic nature can notice the possibility to observe these same features

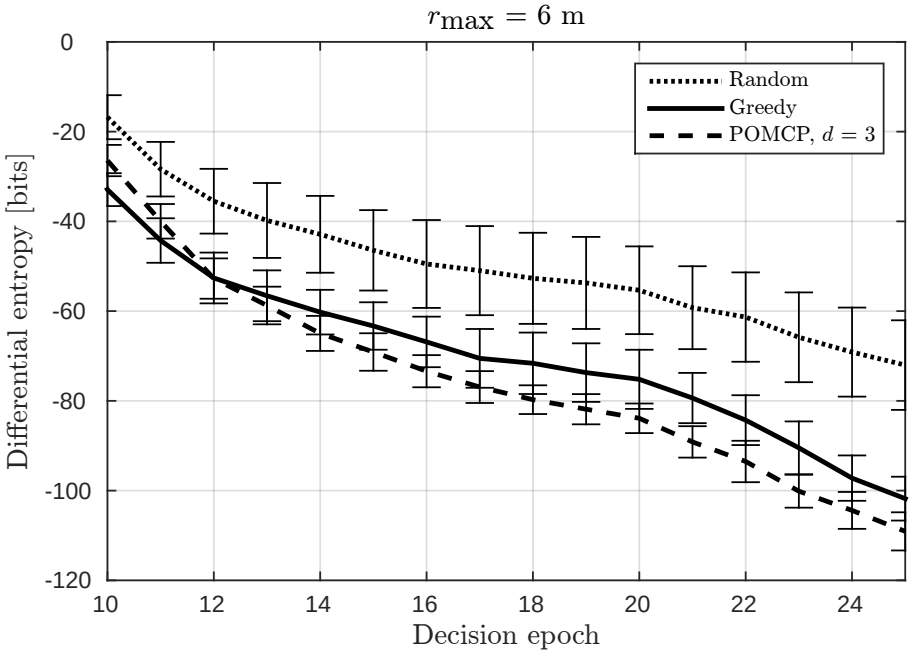


Figure 5.16: The mean differential entropy and its 95% confidence intervals (vertical bars) between 10 and 25 decision epochs for the random, greedy and POMCP policy with $d = 3$.

on later decision epochs and instead prefers to first obtain more information on features around x-coordinate 10 and y-coordinate -1.

The effect these different policies have on the differential entropy is seen in Figure 5.16. The greedy policy first achieves a lower differential entropy at decision epochs up to 11, while afterwards the benefits of the longer-term POMCP policy are seen. The POMCP policy reaches a significantly lower mean differential entropy by decision epoch 21.

Acting greedily considers only immediate information gain and neglects focus change over time. If the allowed rate of change in the camera angle is small, it may take multiple decision epochs to rotate the camera to a desired angle. The greater the range of possible rotational velocities, the less the current decision on the rotational velocity constrains the possible camera orientations reachable over the next decision epochs. We conjecture that in a case with dynamic or moving features, non-greedy planning is also beneficial. Consider for instance a case where a case where some features are observable only over a short period of time. A non-greedy policy may be able to detect situations when new opportunities to sense a particular feature will appear again and they may be observed later, and when it is useful to try and observe them immediately.

5.4.2 Exploration of unknown environments

We next consider a second case where the map representation is geometric instead of feature based, leading to a non-Gaussian belief state.

The robot's state $x \in \mathcal{X}$ is defined as its two-dimensional pose, consisting of its location within the map M and its heading angle. The action space $\mathcal{A}$ in

the problem is continuous, corresponding to the possible control signals (e.g. the linear and rotational velocities) to the mobile robot.

As a representation for the map M, we apply so-called dynamic occupancy grid maps (Meyer-Delius et al., 2012), an extension of classic occupancy grids (Moravec, 1988). This map representation divides $M \subset \mathbb{R}^2$ into equally sized cells $c \in M$. Each cell is in one of two hidden states, free (0) or occupied (1). Cell dynamics are statistically mutually independent two-state Markov chains with parameters p_{11}^c and p_{01}^c , depicting the conditional probabilities that the cell c in the next decision epoch remains in the occupied state and that it transitions from the free state to the occupied state, respectively. Due to the independence of cells, the environment state transition model $\mathbb{T}_m(m', a, m)$ is the product of the individual cells' state transition models, and independent of the robot's actions $a \in \mathcal{A}$.

The robot's belief state is the joint PDF $b = p(x, m)$ over the pose and map. Due to the independence assumption, the PDF over the map is the product of the occupancy probabilities $p_c(t) \in [0, 1]$ of individual cells. It is often practically desirable to constrain the action space in the problem according to the belief state. We adopt a collision-risk constrained approach to constraining the action space by only allowing control actions that do not traverse across any cell c with an occupancy probability $p(c) > 0.15$.

5.4.3 Simulation

We set up a simulated domain where the Markov chain parameters p_{11}^c and p_{01}^c for each cell were known to the robot. The environment is depicted in Figure 5.17, with white showing non-traversable and black showing traversable areas. Throughout the simulations, the robot pose was assumed to be fully observable, and the motion model noiseless. We examined the performance of the POMCP algorithm with the open loop approximation and the SMC algorithm (Algorithm 2.2), applying the RHC principle in both cases.

In all cases, we applied the approximation of MI derived in Theorem 4.20, which can use the same samples as the sampling-based planning algorithms applied. Computational resources can be saved by sampling lazily, i.e. only drawing map samples in parts of the map that are actually observed and thus affect the next observation. The implementation details of such a sampling scheme are given in Appendix C.

In a two-state Markov chain, the PDF over the states monotonically tends towards the stationary distribution. If the occupancy probability of a cell c at decision epoch t is $p_c(t)$ and no observations are perceived, the Markov chain distribution after n steps is

$$p_c(t + n) = p_c^s + \exp(-n/n_0)(p_c(t) - p_c^s), \tag{5.24}$$

where $p_c^s = p_{01}^c/(1 - p_{11}^c + p_{01}^c)$ is the stationary distribution of the Markov chain and $n_0 = -1/\log(p_{11}^c - p_{01}^c)$ quantifies the rate at which the stationary distribution is approached. We considered three cases for the environment dynamics. In the first case, we set $p_{01}^c = 0.01$ and $p_{11}^c = 0.99\,\forall c \in W$, resulting in slow dynamics ($n_0 \approx 49.5$). In the second case, $p_c^{o|f} = 0.15$ and

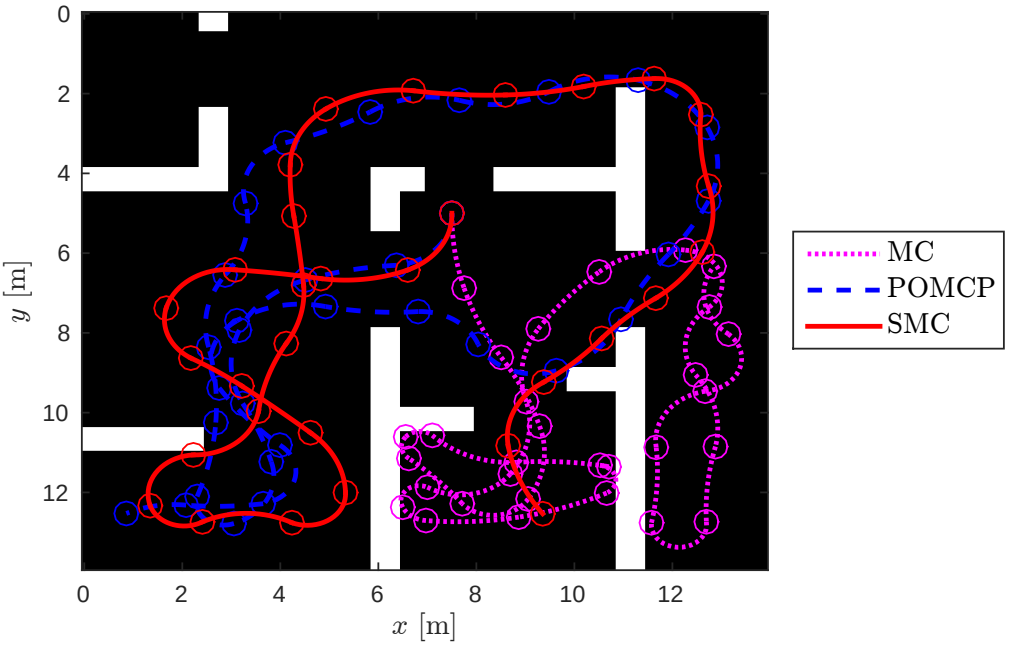


Figure 5.17: The simulation environment and example trajectories obtained by from the MC, POMCP ($N_s = 4000$, $d = 5$) and SMC ($M = 100$, $d = 5$) algorithms. Obstacles are marked by white colour and traversable area by black colour. The poses of the agent at decision times are shown by circle markers. The agent starts at the pose below the opening in the middle-top part of the figure.

$p_c^{o|o} = 0.85\,\forall c \in W$, resulting in fast dynamics ($n_0 \approx 2.8$). The final case was a combination of the two, with $p_c^{o|f} \in [0.01, 0.15], p_c^{o|o} \in [0.85, 0.99]$ sampled uniformly at random independently for each cell. We refer to this case as having medium dynamics. In all cases, the initial information $p_c(0)$ on the map was equal to the stationary distribution p_c^s for each cell c.

Besides the POMCP and SMC algorithms, we evaluated the informativeness of each action was evaluated with a simple myopic Monte Carlo approximation of the MI according to Theorem 4.20 with 50 samples. This corresponds to maximising the one-step look-ahead value, and is similar to the myopic objective function applied e.g. in Stachniss et al. (2005); Sim and Roy (2005); Kollar and Roy (2008). For POMCP and the myopic Monte Carlo method the action space was discretised uniformly to 72 control values.

For POMCP, the number of simulation episodes N_s was set equal to 1000, 2000, or 4000. The exploration parameter e was set to 200 to scale the exploration bonus to a range comparable with typical per-decision MI values. An upper bound for the exploration parameter may be estimated by $e < \frac{N_c I_{\max}}{d} \sum_{t=0}^{d-1} \gamma^t$, where N_c is an upper bound for the number of cells observed per time step, d is the search depth, and $I_{\max}$ is an upper bound for MI per cell, obtained from the observation model.

For SMC, the number of particles M was set to 25, 50, or 100, and $l_{\max} = 12$ iterations were run. The schedule for number of state-observation replicas was linear with $\eta_l = 5 + 3(l-1)$, where l is the iteration number. Resampling was triggered by the effective sample size N_{eff} becoming less than $M/4$. We empirically determined these parameters to be suitable for the experiment,

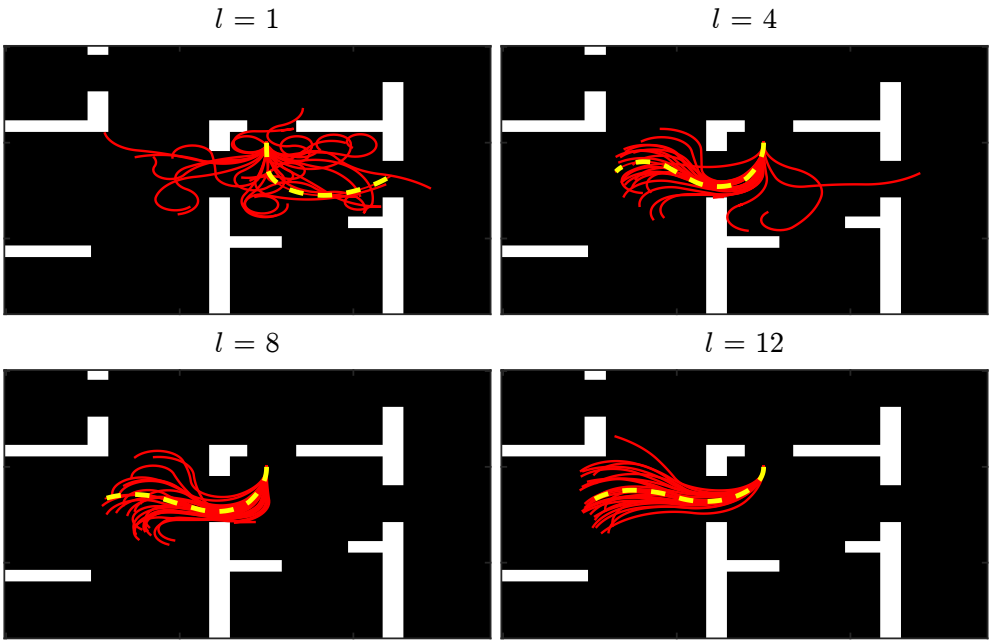


Figure 5.18: Example of SMC algorithm convergence for $M = 100$ particles as function of algorithm iteration step l. The dark areas denote traversable and light areas non-traversable parts of the environment. The lines denote possible trajectories, and the trajectory for the particle with the greatest weight is shown as a thicker dashed line.

observing e.g. that the trajectory judged optimal remained almost constant after less than 10 iterations, see Fig. 5.18. It is more likely that the agent obtains more information while covering a large fraction of the environment with its sensors. This fact was taken into account when sampling the initial control sequences by giving higher probability to velocities closer to the agent's maximum velocity. Steering angles were sampled uniformly at random. Independent Gaussian kernels were applied when modifying solution candidates (Line 4 in Algorithm 2.2) with variance decreasing proportional to the inverse square of the iteration l as a fraction of the full range of possible velocity and steering angle values.

The discount factor in all cases was $\gamma = 0.95$. The search depth d was varied between 1 and 7. In each simulation run, 30 decisions were implemented and the results recorded. For each algorithm and value of d, the simulation was repeated 12 times. All simulations started from the same underlying true state.

Table 5.13: The mean of total mutual information ($\pm$ its 95% confidence interval) for the reference myopic Monte Carlo method, for each rate of dynamics.

Dynamics rate	**Slow**	**Medium**	**Fast**
Total MI [bits]	7360 ± 601	10124 ± 66	12508 ± 78

Table 5.13 summarises the results for the reference MC method. As the rate of dynamics increases, the total MI tends to increase. The stationary occupancy probability p_c^s is 0.5 in each cell for slow and fast dynamics, and on average for medium dynamics. Thus, the maximum amount of information

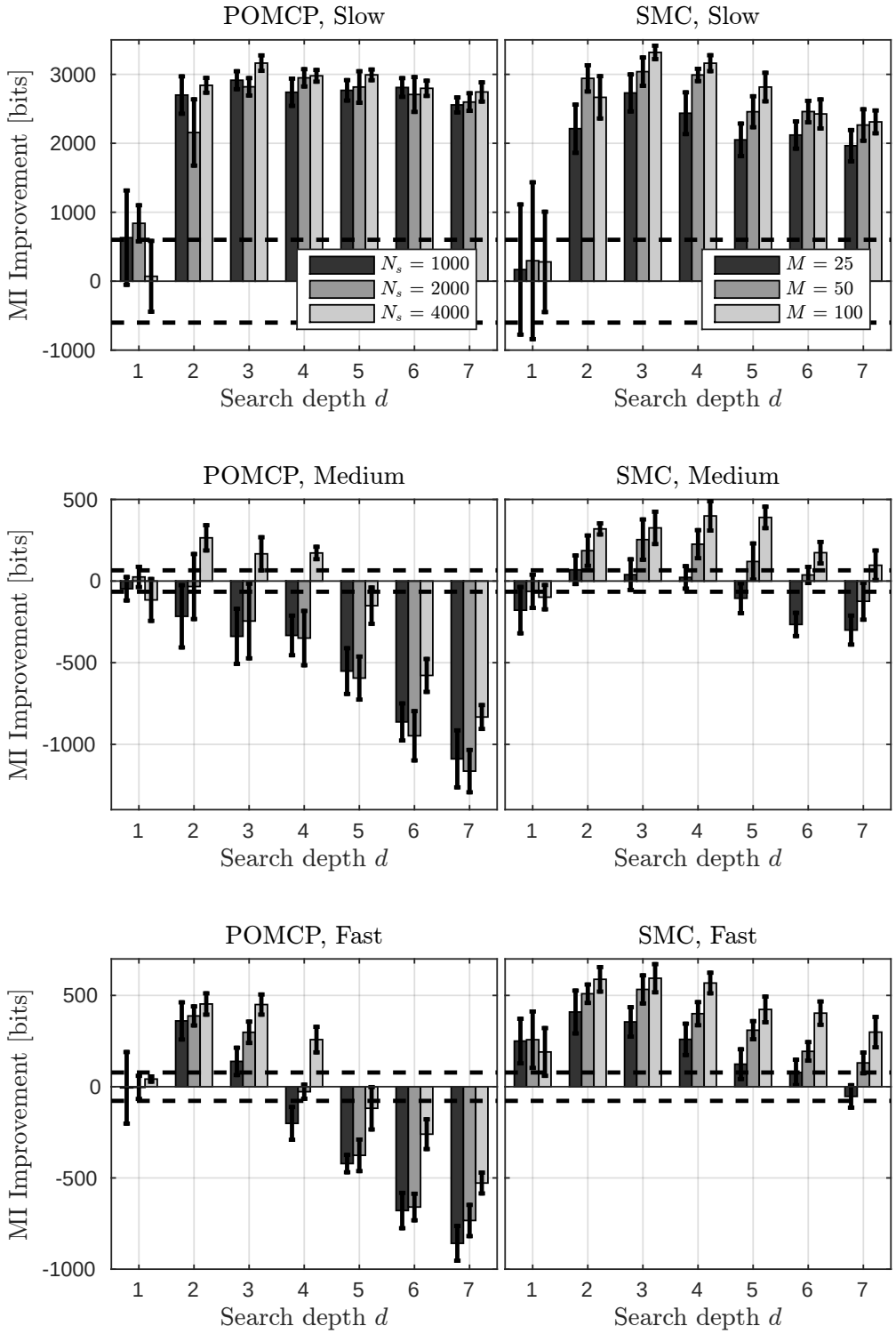


Figure 5.19: Performance of POMCP (left) and OLFC-SMC (right) planners compared to the greedy reference method as function of the search depth d, in the case of slow (top), medium (middle), or fast dynamics (bottom). The bars indicate mean improvement over reference level (Table 1). The 95% confidence intervals are shown by the horizontal error bars. The dashed line indicates the 95% confidence interval of the reference level. The bars in each subfigure correspond to the number of simulations N_s (POMCP) or number of particles M (SMC) shown in the legend of the top subfigures.

about cell occupancy is available when the cell is observed when its Markov chain is at its stationary distribution. As the dynamics rate increases, the Markov chain tends faster towards the stationary distribution, and more MI is accumulated. The greater variation for slow dynamics is also due to the greater variance in the occupancy probabilities, which do not quickly return to the stationary distribution.

Figure 5.19 shows the improvements over the reference results in Table 5.13 obtained with the POMCP and SMC. Results are shown for each search depth and for each dynamics rate. The mean improvement and its 95% confidence intervals are shown. Looking at $d = 1$ in all cases, we note that POMCP has performance statistically equal to the reference, as expected. SMC performance is similar with the exception of the case of fast dynamics, although the difference even there is slight. Since the SMC method which considers the full continuous action space achieves similar performance as POMCP applying a discretised action space, we can conclude that our discretisation of the action space is dense enough for the myopic problem.

In all cases, performance tends to improve as the number of simulations N_s in the POMCP algorithm or the number of particles M in the SMC algorithm increase. Increasing these parameters improves the coverage over the search space in the underlying open loop optimisation problem and the accuracy of the MI approximation, helping to find solutions with greater expected MI more reliably.

Considering the cases with $d > 1$ in Figure 5.19, we see the greatest improvement over the reference for slow dynamics. For slow dynamics, moving away from the currently sensed area will provide a greater amount of MI than staying in the same area. However, a myopic planner with $d = 1$ was observed to select actions that lead to dead-ends. We observed the relative frequency each cell was observed, and noted that for $d = 1$ most frequently observed cells were at the lower middle or lower left corner of the environment, whereas for $d > 1$ cells around the map were observed equally frequently. Example trajectories illustrating this are shown in Figure 5.17 for MC, POMCP ($N_s = 4000, d = 5$) and SMC ($M = 100, d = 5$). The MC trajectory initially gets stuck at the lower part of the map, while both POMCP and SMC avoid this dead-end. In general, the presence of actions that are informative but ultimately lead to a situation with few options to gain more information can be avoided by multi-step planning.

For $d > 1$ in the case of medium and fast dynamics, Figure 5.19 shows mixed results. The rapid decrease of POMCP performance in these cases for $d \geqslant 4$ is due to insufficient sampling of the search space. The number of possible action sequences increases exponentially as a function of d. As the search depth d increases, a constant number of simulation samples covers an increasingly smaller fraction of possible action sequences and related possible realisations of the environment state. For example, with 4000 simulations, the typical depth of the search tree generated by the SMC algorithm was 4 in our experiments. This indicates that more simulations would be needed to find action sequences that cover the desired search depth. The effect is amplified for faster dynamics, since for slow dynamics even a few samples of the environment state provide an accurate prediction. A similar effect is seen also for SMC, although less severe. There, likewise, it is increasingly

difficult to cover the exponentially increasing search space by a constant number of particles.

If a large search depth results in trajectories that lead the robot beyond its current sensor range, the evaluated usefulness of such trajectories depends greatly on the prior map information in these areas. If no prior information exists, the usefulness of such trajectories is doubtful. However, if knowledge of the domain is applied, e.g. that the environment is an indoor office environment, to obtain map samples that reflect such prior information one could e.g. draw them from a corresponding database of map realisations (Strom et al., 2015).

As for the environment dynamics, we may conclude that increasing the search depth is useful up to the order of $n_0/2$ time steps. Consider n_0 in (5.24) as an inverse exponential decay parameter. After n_0 steps, the occupancy distribution is again close to the stationary distribution p_c^s and observing the cell again is beneficial. Specifically in the case of $p_c^s = 0.5$, reobserving the cell is maximally useful the closer its occupancy distribution is to p_c^s.

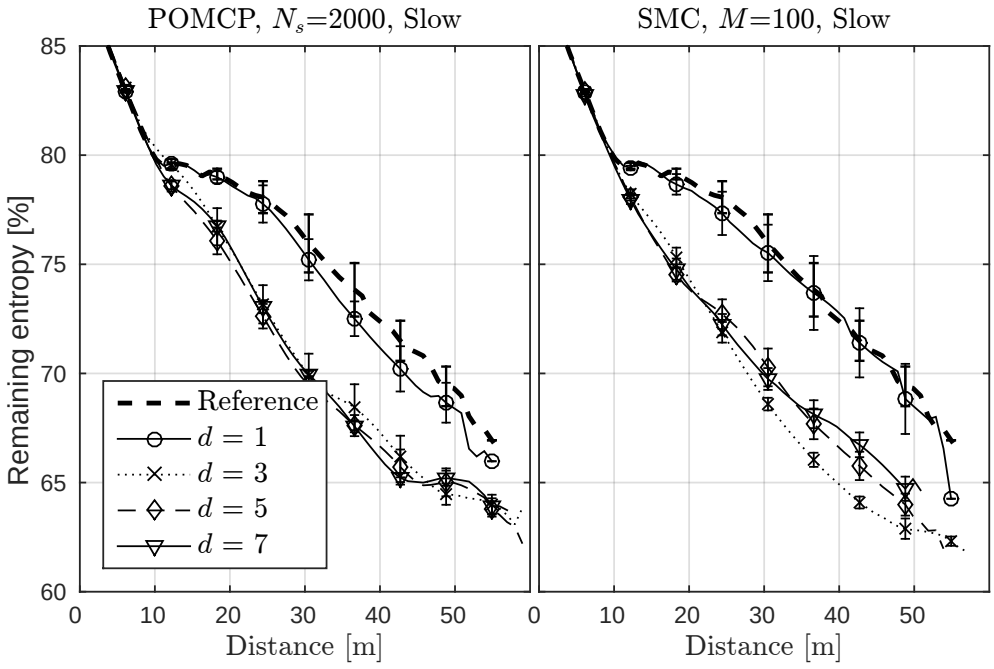


Figure 5.20: Average entropy (lines) and its 95% confidence intervals (bars) for different search depths d as a function of the distance travelled applying the POMCP algorithm with $N_s = 2000$ simulations (left) and the SMC algorithms with $M = 100$ particles (right), in the case of slow dynamics. The greedy Monte Carlo reference method's average entropy is shown by the dashed line.

We finally examined the evolution of the fraction of remaining map entropy as a function of distance travelled by the agent in the simulations with slow dynamics rate. Representative results for both algorithms are shown in Figure 5.20. For clarity of presentation, the results are shown for $d = 1, 3, 5, 7$ and the error bars are not drawn for every data point. We note that like for the total MI, $d = 1$ has equal performance compared to the reference MC algorithm. For search depths $2 \leqslant d \leqslant 4$, both POMCP and SMC typically achieve a significantly lower entropy than the reference method and thus a more informative belief state about the map. Results are mixed for $d > 4$ due to the reasons outlined earlier.

Applying multi-step planning in the simulation environment studied results in significantly better performance than myopic planning, both measured by the total mutual information collected and by the evolution of entropy of the map information over distance travelled. Multi-step planning avoids actions that seem useful at first but lead to situations where there are few possibilities for further information gain. We however also observed that increasing the search depth does not monotonically improve the results, and noted that this is due to decreasing coverage of the space of possible open loop solutions, and the combined effect of proposed trajectories leading the agent outside current sensor range and environment dynamics with no prior information.

5.4.4 Implementation

In this subsection, we discuss some details of the real-world implementation of a robotic exploration technique based on solving the robotic exploration problem (Problem 3.7) applying the SMC algorithm (Algorithm 2.2). The goal of the implementation was to enable use of the exploration technique together with an existing SLAM system with minimal redesigning of existing software.

We can distinguish three cases for the environment dynamics in an robotic exploration problem, in increasing order of generality:

1. a stationary map with $p_{11}^c = 1$ and $p_{01}^c = 0$ for all cells c,
2. a dynamic map with known p_{11}^c and p_{01}^c, or
3. a dynamic map with unknown p_{11}^c and p_{01}^c.

Figure 5.21 shows the proposed software implementation of our robotic exploration technique. The top right part of the figure shows a software module for estimating the cell state transition probabilities p_{11}^c and p_{01}^c at each cell, a task that is typically not handled by pre-existing SLAM implementations. The module takes as input the current map and pose information $p(x_t, m_t)$ from SLAM along with the sensor data z_t, e.g. odometry and LRF data.

A stationary map is handled by permanently setting the output of the estimation module as $\hat{p}_{11}^c = 1$ and $\hat{p}_{01}^c = 0$ for all cells. The case of a dynamic map with known dynamics is handled similarly, instead permanently setting $\hat{p}_{11}^c$ and $\hat{p}_{01}^c$ for all cells to their known values. A minimal implementation for estimation of the unknown parameters p_{11}^c and p_{01}^c in the case of a dynamic occupancy grid map is as follows. The dynamics estimation module is connected to an existing SLAM implementation, shown at the top left of the figure. First, the joint PDF of the pose and map obtained as SLAM output is marginalised to obtain a marginal PDF of the pose. Then, the estimates $\hat{p}_{01}^c$ and $\hat{p}_{11}^c$ of the state transition parameters are updated based on a Bayesian update rule for Binomial parameter estimation, where the correct cell c is determined assuming that the data z_t was collected at the maximum likelihood pose $\hat{x}_t$. In a more sophisticated implementation, one might instead consider the current PDF over the state without resorting to the maximum likelihood approximation.

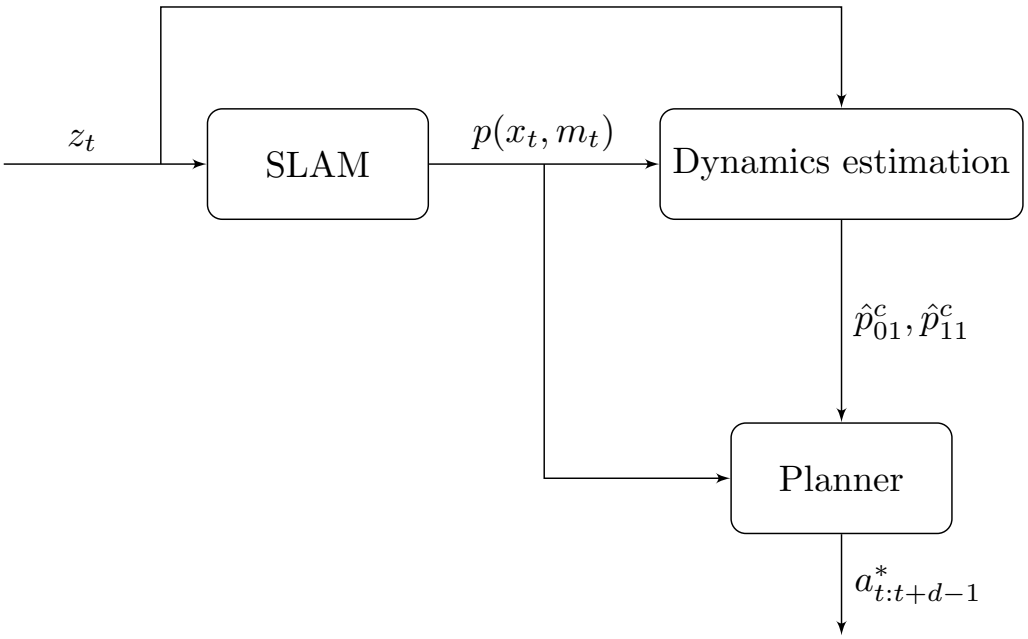


Figure 5.21: A software implementation of the exploration technique. The blocks indicate software modules, and arrows indicate propagation of signals between modules, labelled with the mathematical definition of the signal.

LRFs are typical sensing equipment for mobile robots. LRF output is a set of pairs of laser beam incidence angle and the distance from which the beam reflected back to the sensor. This can be handled directly, setting the observation space equal to the space of possible incidence angles and the continuous interval of possible distances. If representing the map by an occupancy grid, one could abstract the observation into an observation about a subset of the cells in the map; describing whether the laser beam hit an obstacle in an specific cell (and thus reflected back) or whether it passed through an cell without reflecting. A more detailed discussion on LRF sensing models is provided e.g. by Thrun et al., 2006, Ch. 6.

The planner module shown on the bottom left of Figure 5.21 implements the SMC algorithm, taking as inputs the current SLAM estimate of the robot and environment state, state, and the parameter estimates $\hat{p}^c_{01}$ and $\hat{p}^c_{11}$. The planner produces as output the open loop action sequence $a^*_{t:t+d-1}$ that is then output to an module executing the control signals.

5.4.5 Experimental validation

An exploration experiment was carried out with a car-like mobile robot platform equipped with a LRF. The robot was controlled by a linear velocity v and a steering angle ω. As these quantities reside in continuous spaces and we wished to not limit the space of possible solutions explored, the experiments applied exclusively the SMC algorithm (Algorithm 2.2). In addition, compared to the POMCP algorithm with a discrete set of actions, the SMC algorithm dynamically varies the action sampling space depending on the situation. The SLAM module in Figure 5.21 was the Rao-Blackwellized particle filter (RBPF) SLAM algorithm of Grisetti et al. (2007). The SMC algorithm within the planner module was implemented in C++ programming language as a component for the Robot Operating System (ROS) framework

(Quigley et al., 2009). All computations were performed using the robot's on-board computer, equipped with an Intel i7 multi-core processor, 16GB of RAM and running a Linux operating system.

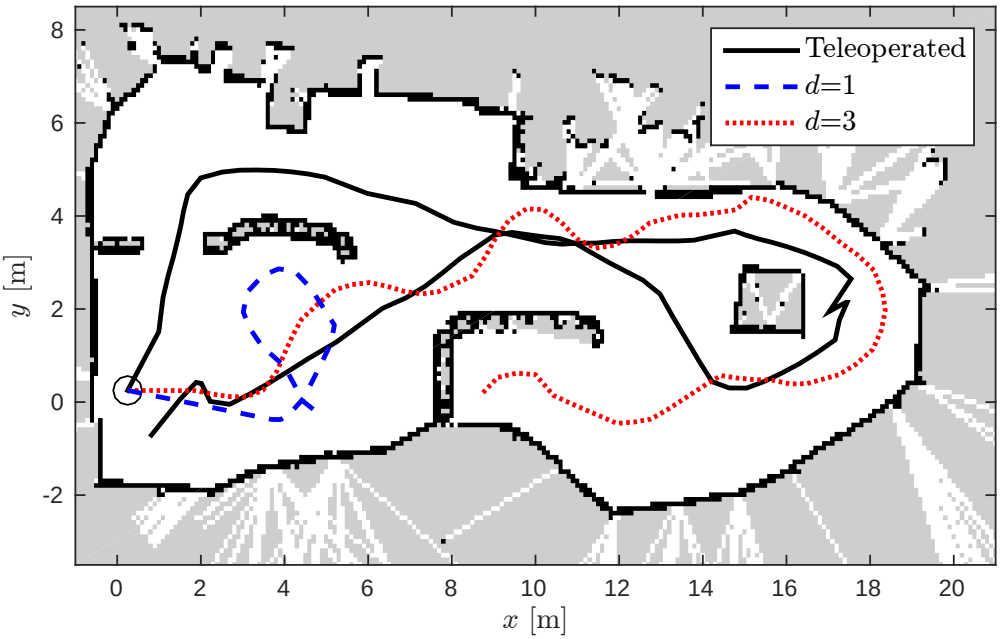


Figure 5.22: A map of the experimental area. White, black and gray cells denote free, occupied and unknown areas, respectively. The approximate starting location in the experiments is indicated by a circle marker. The trajectory travelled under human teleoperation is drawn with a solid line. Example trajectories for search depths $d = 1$ and $d = 3$ have been drawn with a dashed and dotted lines, respectively.

A map of the experiment area is shown in Figure 5.22. The size of the map is approximately 10 by 20 meters, with a cell resolution of 0.1 meters. The map is obtained by SLAM with a human teleoperating the robot and visiting each area in the environment. The trajectory travelled during teleoperation is drawn as a solid line. The environment has several open areas connected together by narrow openings. The walls between the open areas prevent the robot from observing occluded parts of the environment, necessitating exploration to fully reveal the structure of the environment. In addition, there are always at least two optional routes for further advance. Once the robot commits to one of the routes, switching to the other requires manoeuvring that takes time. In the simulation experiment, such features favoured multi-step planning while greedy algorithms performed poorly. Due to difficulty of implementing dynamic features within the environment consistent enough to preserve the repeatability between experiments, we did not attempt to include such elements. The dynamics estimation module was thus set to correspond to the static map assumption with output $\hat{p}^c_{11} = 1$ and $\hat{p}^c_{01} = 0$ for all map cells c.

The experiments applied a search depth d of 1, 3, 5 or 7 decisions. The other parameters for the planner were selected based on the results of the simulation experiment and to obtain reasonably short computation times. The number of iterations in the SMC algorithm was $l_{\max} = 6$, with a linear schedule for the number of simulations, $\eta_l = 5 \cdot l$. The number of particles was $M = 25$. A

Gaussian kernel with variance decreasing proportional to the inverse square of the iteration l was applied in modifying the solution candidates. While sampling of control sequences, we checked their corresponding trajectories against the current map information and rejected trajectories that intersected cells where occupancy probability was greater than 0.15.

A car-like dynamic model was applied for the vehicle. As the terrain in the experimental area was flat, the robot could execute single movement actions accurately. Thus the applied model was assumed noise-free. This coincides with the assumption of reaching the maximum likelihood pose $\hat{x}_{t+1}$ as a result of movement actions. Thus MI of robot pose and observation data, the first term in the sum of Equation (4.48) in Theorem 4.20, is assumed equal to zero.

For computing the MI of the map and observation, laser scans were sampled in the frontal sector of the robot. Incidence angles between -45 and 45 degrees were considered. The angular resolution was 0.5 degrees, and the maximum distance considered was 10 meters. The planner thus attempts to maximise mutual information of observations in this sector in front of the robot. The laser observation model was a simple ray-tracing model, with false positive and negative probabilities of 0.05 each. For sampling observations to evaluate Equation (4.48), we obtained observations by ray tracing beams from location $x_{t+1}^{(i)}$ in the known map sample $m_{t+1}^{(i)}$. The lazy sampling scheme from Appendix C was applied.

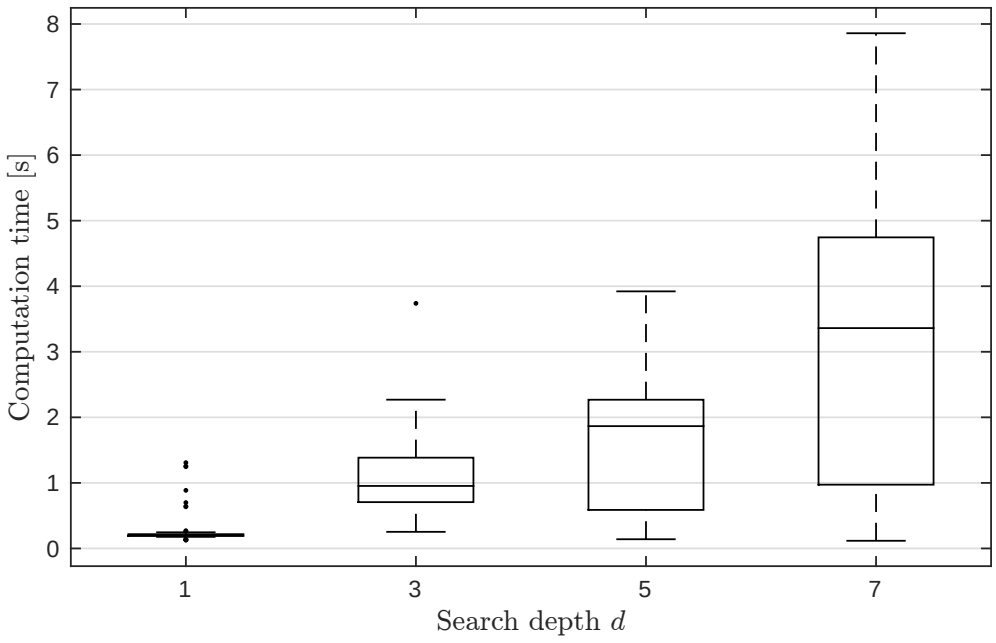


Figure 5.23: Computation time as a function of the search depth d. The lower and upper edges of the boxes indicate 25th and 75th percentiles, respectively. The line across the boxes indicates the median. The whiskers cover approximately 99% of the variation in the data. Outliers are drawn with dot markers.

Due to the simplicity of the models applied and by using parallel processing for the independent simulations in the planner, we were able to obtain planning times of less than 1 second for search depths 1 and 3, and typical times of less than 4 seconds even for longer search depths, as indicated by

Figure 5.23. Although computation time is strongly dependent on the choice of parameters and computer hardware, we believe this is an indication that the planner may be fast enough for some real-time applications.

Table 5.14: A qualitative summary of the experimental results. For each search depth the 5 experiments are classified into successes and failures. Failures are further classified by their reason, either due to planning algorithm finding no feasible solutions or no progress being made in exploring the environment (as judged by the experimenter).

Search depth d	**1**	**3**	**5**	**7**	**Total**
Success	1	4	4	3	12
Failure	4	1	1	2	8
Planner stuck	1	0	1	0	2
No progress	3	1	0	2	6

For each search depth d, the experiment was repeated for 5 times with the same robot starting pose. Each repetition was terminated at a cut-off time of approximately 4 minutes, or when the robot had completed exploration, as judged by the experimenter monitoring the quality of the map produced, or when a failure state occurred. Failure states include the algorithm getting stuck finding no feasible control trajectories, or, as judged by the experimenter, not making progress in exploration e.g. moving repeatedly around in the same location. A summary of qualitative results of the experiments is provided in Table 5.14. We note that most failures are due to no progress being made in exploration, while only 2 failures in the total of 20 experiments occurred due to the planner finding no feasible solutions. We believe this to be an indication of the robustness of the planning algorithm, as the current failure rate was achieved with a straightforward performance-oriented implementation of the planner. A typical cause for the "no progress" type failures was repeatedly moving around in the same area.

To evaluate the exploration performance of the algorithm, we compared the fraction of full map entropy remaining as a function of time and distance travelled. The results are shown in Figures 5.24 and 5.25, respectively.

When the remaining entropy value is 100% the occupancy information is $p(c) = 0.5$ for all cells c within the experimental area, and when the value is 0% the occupancy information is $p(c) = 0$ or $p(c) = 1$ for all cells, i.e. entropy is zero. The lines start at approximately 80% remaining entropy, after the robot observes the environment first at its starting pose. A dashed line in each subfigure indicates the performance of the human-operated reference run. The reference run lasted for 150 s and thus the reference line in Fig. 5.25 is flat after that time. Each of the other lines depicts the result of one of the five repetitions of an experiment. The lines indicate that the durations and lengths varied between experiments.

The performance of the planner improves significantly when applying non-myopic planning with $d > 1$. As in the simulation experiment, a longer search depth helps avoiding actions that seem informative at first but lead to poor information gain later. A typical example observed during the experiments was the robot turning to observe the area in that direction,

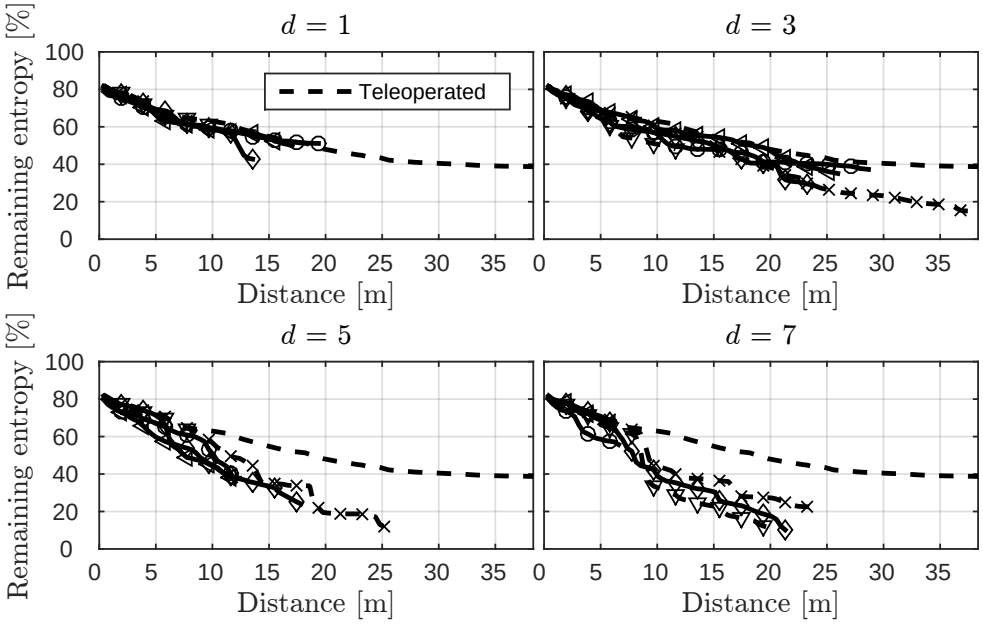


Figure 5.24: The percentage of entropy of a map of the whole area remaining as a function of the distance travelled, in all five experiments for each search depth d. The dashed line with no markers is the same in each axis, showing the reference result from manual teleoperation.

while ignoring that travelling in the current heading would lead it to a new area with greater potential information gain later. The improvement is seen comparing the two example trajectories for $d = 1$ and $d = 3$ shown in Figure 5.22. The robot travelling along the dashed line for $d = 1$ turns left back towards the start location instead of proceeding further ahead, shortly after which the experimenter terminated the experiment as no progress was being made. In contrast, for $d = 3$ the algorithm chooses in a similar situation to travel further ahead as indicated by the dotted line, which leads it to a new area to explore on the right hand side of the map.

The performance as a function of the search depth improved consistently, unlike in the simulation experiment. The improvement is seen from Figure 5.24, where longer search depths consistently lead to lower entropy at the same distance travelled. We believe this is primarily due to that in the experiment the sensor range is greater than the length of the trajectories considered by the planner. Thus, the trajectories are within the area currently observable by the robot's laser range finder. This reduces the need to draw samples from uniform prior map information in so far unobserved areas which was noted in the simulation experiment as one reason for increasingly poorer performance with longer search depths.

Performance as a function of distance was similar to a human operator for all search depths up to the first 10 meters travelled. However, beyond this distance performance starts to differ such that $d = 5$ and $d = 7$ have significantly better performance. For $d = 3$, the planner does not consistently outperform the human operator. As a function of time, performance of the planner was closer to that of the human operator. As seen from Figure 5.25, the human operator never performs consistently worse than the planner.

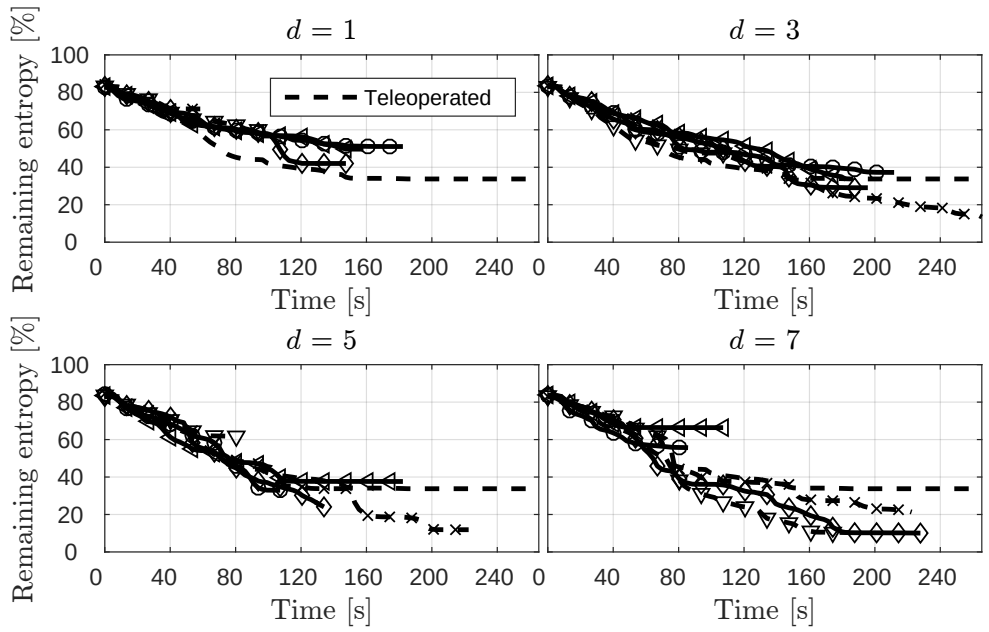


Figure 5.25: The percentage of entropy of a map of the whole area remaining as a function of time, in all five experiments for each search depth d. The dashed line with no markers is the same in each axis, showing the reference result from manual teleoperation.

The human operator's performance differed from the planner due to two main factors. First, the human could view the entire experimental area and decide beforehand on a route to travel. This enables the human to easily exceed the performance of a myopic planner. The route the operator travelled covers the area well, but does not consider e.g. that a certain area could be observed as efficiently along a shorter route or that it might be useful to spend more time in some locations to observe them more accurately. The human operator might perform better by also viewing the current map online during the teleoperation task. However, dividing attention between multiple tasks may itself degrade performance, or may not always be safe. We remark that as the operator attempted to maximise a subjective, qualitative estimate of area coverage rather than quantitative information gain, concluding that algorithmic exploration outperformed the human operator is not possible.

The results of the experiments show that the exploration technique based on solving the robotic exploration problem applying the SMC algorithm works in practice in a real-world environment. The SMC algorithm samples action sequences adaptively based on current map information, making it more flexible than approaches relying on a fixed discretisation of the action space. Long search depths are useful if there is prior knowledge on the map and the environment dynamics are slow.

The exploration approach currently only takes into account the information gain. As concluded by Amigoni and Caglioti (2010), even better exploration performance might be achieved by simultaneously considering also the cost of traversing the possible trajectories, e.g. by comparing the time or energy required.

6

Discussion and conclusion

In this chapter we summarise and discuss the results obtained in the thesis. We also return to the research questions stated in Chapter 1, and provide answers based on the results obtained. Finally, discussion about the results and possible avenues for future work are given to conclude the thesis.

6.1 Summary of results

This thesis studied sequential decision-making in a POMDP framework applied to sensor management problems. The primary application area considered was mobile robotics.

In Chapter 2, a literature survey of state-of-the-art algorithms for solving POMDPs for sensor management tasks was provided. A set of canonical sensor management problems in mobile robotics was defined in Chapter 3, and the results of the literature survey were later applied in Chapter 4 to analyse which existing algorithms are applicable to the canonical problems.

The canonical problems were path planning, environmental monitoring, task support and robotic exploration. In Chapter 4, it was studied how to take advantage of the domain features in the canonical problems to solve the sensor management task more effectively.

For path planning problems on a graph with a partially observable environment state, heuristics and upper bounds for the optimal value were derived in Section 4.2.

For environmental monitoring problems that fulfil certain independence and partitioning conditions between the monitored variables and the actions of the monitoring agent, it was shown in Section 4.3 that when constraints on the robot's motion are relaxed, the relaxed POMDP problem with an information-theoretic reward function is equivalent to a multi-armed bandit (MAB) problem. It was shown that the optimal solution of the MAB problem is an upper bound on the optimal solution of the original, unrelaxed problem. The upper bound may be applied in a branch-and-bound pruning algorithm to optimally solve the unrelaxed problem. Furthermore, it was shown that the case of two-state MAB arms with mutual information (MI) as the reward

function has a monotonicity property that leads to an easily identifiable optimal policy for the MAB.

For task support problems with a continuous state space and finite action and observation spaces, a new error bound for a point-based value iteration algorithm was derived in Section 4.4. The error bound was applied to design an algorithm for sampling belief states for point-based value iteration such that the error is minimised.

For robotic exploration problems, the conditional independence properties between the robot state and environment state were exploited in Section 4.5 to derive a new Monte Carlo approximation for the MI of the state and observation in the problem. As the approximation applies random samples to estimate the MI, it can be integrated with sampling-based algorithms for approximately solving POMDPs.

In Chapter 5, exploiting the domain properties was studied in all problems via simulation experiments. Furthermore, a real-world implementation of a POMDP-based sensor management approach to robotic exploration was proposed and demonstrated.

6.2 Answering the research questions

The research questions of this thesis were stated in Section 1.4. In this section, we provide answers to the research questions in the context of the work performed in this thesis and the obtained results. For the reader's convenience, each research question is repeated below before discussing its answer.

RQ 1: How should sensor management problems be formulated as POMDPs?

POMDPs with a reward non-linear in the belief state were shown to represent a variety of relevant sensor management problems ranging from environment monitoring to exploration. Furthermore, the experimental results show that approximate solutions to such POMDPs work reasonably in practice and outperform e.g. greedy policies.

POMDPs with a state dependent reward function have been extensively studied in the literature and powerful algorithms exist for finding approximately optimal solutions to them. Although current approaches to finding approximately optimal solutions to POMDPs with rewards non-linear in the belief state are as of yet not as well studied, our results show that structural properties of the problem inputs can be exploited to effectively find approximate policies for such sensor management POMDPs.

RQ 2: Which structural properties of sensor management problems can be taken advantage of to tailor existing exact and approximate algorithms for solving POMDPs to be efficient in solving sensor management problems in mobile robotics?

A review of existing work on POMDP-based control for real robotic platforms was presented, and based on the review a set of canonical sensor management POMDPs relevant to mobile robotics was defined.

In sensor management POMDPs with a reward function non-linear in the belief state, taking advantage of mixed observability, underlying graph structures and independence properties in state transition and observation models and reward functions is useful. In the case of reward functions linear in the belief state, the results of case studies suggest that exploiting graph structures may not be worthwhile, neither from a point of view of solution quality nor computational effectiveness.

In case studies on the mixed observability path planning problem (Problem 3.4), the bounds found by taking advantage of the underlying graph structure in the problem are loose. Applying the bounds did not result in finding better approximate policies or saving computation time compared to a state-of-the-art sampling-based algorithms.

For the mixed observability environment monitoring problem (Problem 3.5), a relaxation was found by exploiting the underlying graph structure. If the relaxation can be shown to be a multi-armed bandit (MAB) problem, its solution can be exploited in a branch-and-bounding algorithm for the original unrelaxed problem. When computational savings due to the reduced number of search tree branches expanded are greater the computational cost of computing the bounds, shorter planning times are achieved without sacrificing optimality.

The robotic exploration problem (Problem 3.7) features conditional independence properties for the robot and environment state that were taken advantage of. In the case studies it was demonstrated that applying an approximation of MI based on exploiting these conditional independence properties allows implementing a sensor management approach for real-world robotic exploration.

RQ 3: For the approximate methods considered, what guarantees on the quality of the solution can be provided compared to the optimal solution?

Through the case studies it was found that the closer a problem formulation is to the real physical world, the more compromise is required on the quality guarantees for the produced solution. For finite state, action and observation spaces with mixed observability under certain independence assumptions, optimal policies could be found e.g. in the environment monitoring problem. In the robotic exploration problem, a particle-based simulated annealing algorithm was found to be the only feasible current method able to handle the continuous action space and large observation space.

To deal with large state spaces such as in the path planning problem, sampling-based approaches such as POMCP can be applied that provide an asymptotic convergence guarantee to the optimal solution (Silver and Veness, 2010). If the observation space is continuous, it may further be necessary to apply open loop approximations with the receding horizon control principle which in general do not perform as well as the closed

loop optimal solution (Bertsekas, 2005). To cope with continuous action spaces, the SMC algorithm applies particle-based simulated annealing, with asymptotic convergence properties (Kantas et al., 2009).

6.3 Discussion and future work

Based on the experiences gained, computational complexity remains the most challenging aspect of applying POMDP based control in mobile robots. Accurate modelling of real-world problems often entails uncountable and continuous state, action, and observation spaces. To overcome the resulting computational challenges in planning, one may simplify the models to make them more tractable e.g. through discretisation. However, this results in a hard-to-quantify performance loss. Another option is to select a planning algorithm capable of directly handling the continous spaces. Such algorithms include ones based e.g. on reinforcement learning (RL) and use of function approximators. According to our best knowledge, RL algorithms for POMDPs have so far not been demonstrated in realistic scale robot control domains. Studying such applications for rewards both linear and non-linear in the belief state is a potential topic for future research.

In some of the experiments we carried out, such as the camera system operation case of Subsection 5.4.1, we did not always notice a clear performance improvement when applying non-myopic planning. Indeed, characterising the "adaptivity gap" in POMDP domains, i.e. the performance difference between the one-step greedy and a closed loop feedback policy, is another possible topic for future research.

Open-loop feedback control based on the receding horizon control principle was shown to be a computationally feasible way of implementing non-myopic POMDP-based control for real mobile robots. Deriving tighter bounds on the performance loss compared to the optimal closed-loop policy, especially for rewards non-linear in the belief state, remains to be explored.

Sampling-based planning methods, such as POMCP, might be further improved by applying application-specific rollout policies (Lauri et al., 2015). For the sampling approach presented for robotic exploration, sampling map hypotheses from a database as e.g. in Strom et al. (2015) seems promising. In the spirit of Theorem 4.17, we have recently analysed more general POMDP problems for sensor management. Structural results bounding the optimal policy from below and above may prove useful for increasing computational effectiveness (Lauri et al., 2016).

Finally, multi-agent decision-making presents another natural direction for future work. In a decision-theoretic context, applications such as information gathering by teams of robots could be handled in a principled manner in a decentralized POMDP framework (Goldman and Zilberstein, 2004; Oliehoek, 2010).

A

Information theory

This appendix provides a brief overview of some central concepts in information theory. The material is wholly based on (Cover and Thomas, 2006, Chap. 2). We have omitted subsequent references to this work within this appendix, unless referring to a particular result or theorem. The concepts and quantities introduced are properties of random variables and their PDFs. The general view adopted is that a PDF quantifies a state of knowledge or information.

A.1 Entropy and conditional entropy

The entropy of a random variable is a measure of its uncertainty.

Definition A.1 (Entropy)**.** *The entropy $H(X)$ of a discrete random variable $X \sim p(x)$ is defined as*

$$H(X) = \mathbb{E}_X\left[-\log p(x)\right] = -\sum_{x\in\mathcal{X}} p(x)\log p(x). \tag{A.1}$$

Entropy is measured in *nats* when the logarithm is base e, and in *bits* when the logarithm is base 2. For discrete random variables, it always holds that $H(X) \geqslant 0$, and the greater the uncertainty related to X, the greater the entropy. This is easily seen by an illustrative example considering a binary random variable, which assumes value 1 with probability p and, conversely, value 0 with probability $1-p$. The entropy of the binary random variable is derived according to (A.1) as

$$H(X) = -p\log p - (1-p)\log(1-p) \equiv H(p). \tag{A.2}$$

The definition of entropy may be extended to a pair of random variables (X, Y) with a joint PDF $p(x, y)$.

Definition A.2 (Joint entropy)**.** *The joint entropy $H(X, Y)$ of a pair of discrete random variables $(X, Y) \sim p(x, y)$ is defined as*

$$H(X,Y) = \mathbb{E}_{X,Y}\left[-\log p(x,y)\right] = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y). \tag{A.3}$$

Finally, the conditional entropy of a random variable Y given another random variable X is the expected value of the entropies of the conditional distributions, averaged over the conditioning random variable.

Definition A.3 (Conditional entropy)**.** *Given a pair of discrete random variables $(X, Y) \sim p(x, y)$, the conditional entropy $H(X \mid Y)$ is defined as*

$$\begin{aligned} H(X \mid Y) &= \mathbb{E}_Y \left[H(X \mid Y = y) \right] \\ &= -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x \mid y) \log p(x \mid y). \end{aligned} \tag{A.4}$$

A.2 Relative entropy and mutual information

Relative entropy is a quantification of the "distance" between two PDFs. Relative entropy is also often called the KL divergence.

Definition A.4 (Relative entropy)**.** *The relative entropy or Kullback-Leibler divergence between two probability mass functions $p(x)$ and $q(x)$ with the same support $x \in \mathcal{X}$ is defined*

$$D(p, q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{A.5}$$

It is readily apparent from the definition that the KL divergence is not symmetric, and hence not a true distance. The KL divergence is always non-negative and equal to zero if and only if $p = q$.

Mutual information (MI) is a measure of the amount of information that one random variable contains about another random variable. It can be viewed as a generalisation of the familiar correlation coefficient, which captures *linear* interdependence between random variables.

Definition A.5 (Mutual information)**.** *Let $(X, Y) \sim p(x, y)$ with marginal probability mass functions $X \sim p(x)$ and $Y \sim p(y)$. The mutual information (MI) of X and Y, denoted $I(X; Y)$, is the expected reduction in the entropy of X when it is conditioned on Y:*

$$I(X; Y) = H(X) - H(X \mid Y). \tag{A.6}$$

Equivalently, MI is the relative entropy between the joint distribution and the product of the marginals $p(x)p(y)$:

$$\begin{aligned} I(X; Y) &= D(p(x, y), p(x)p(y)) \\ &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\ &= \mathbb{E}_{X,Y} \left[\log \frac{p(x, y)}{p(x)p(y)} \right]. \end{aligned} \tag{A.7}$$

For any two random variables X, Y MI always satisfies $I(X; Y) \geqslant 0$ with equality if and only if X and Y are independent (i.e. $p(x)p(y) = p(x, y)$).

The following theorem (Cover and Thomas, 2006, Thm. 2.4.1) collects the key properties and connections between mutual information, entropy and conditional entropy.

Theorem A.6 (Properties of mutual information and entropy)**.**

$$I(X;Y) = H(X) - H(X \mid Y) \tag{A.8}$$
$$I(X;Y) = H(Y) - H(Y \mid X) \tag{A.9}$$
$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{A.10}$$
$$I(X;Y) = I(Y;X) \tag{A.11}$$
$$I(X;X) = H(X) \tag{A.12}$$

Conditional MI is the MI of two random variables given a third.

Definition A.7 (Conditional mutual information)**.** *The conditional MI of random variables X and Y given Z is defined*

$$\begin{aligned} I(X;Y \mid Z) &= H(X \mid Z) - H(X \mid Y,Z) \\ &= \mathbb{E}_{X,Y,Z}\left[\frac{p(x,y \mid z)}{p(x \mid z)p(y \mid z)}\right]. \end{aligned} \tag{A.13}$$

The conditional MI often appears together with the chain rule for mutual information.

Definition A.8 (Chain rule for mutual information)**.** *The chain rule for mutual information is*

$$I(X;Y,Z) = I(X;Z) + I(X;Y \mid Z). \tag{A.14}$$

B

Gaussian mixture model simplification

Gaussian mixture models (GMMs) are linear combinations of Gaussian PDFs, weighted such that the resulting function is a proper PDF. A GMM $p(x)$ with n components is

$$p(x) = \sum_{i=1}^{n} w_i N(x; \mu_i, \Sigma_i), \tag{B.1}$$

where $N(x; \mu_i, \Sigma_i)$ is the ith Gaussian component with mean vector μ_i and covariance matrix Σ_i, and $w_i \geqslant 0$ is the weight of the ith Gaussian component. The weights must additionally satisfy $\sum_{i=1}^{n} w_i = 1$. GMMs are a powerful choice for modelling, as they can accurately approximate arbitrary PDFs when the number of components is increased (MacLachlan and Peel, 2000).

In the following, we denote Gaussian components by the shorthand notation $p_i(x) \equiv N(x; \mu_i, \Sigma_i)$. Given a GMM $p(x) = \sum_{i=1}^{n} w_i p_i(x)$, we review methods that find an approximation $q(x) = \sum_{j=1}^{k} c_j q_j(x)$ with $k < n$ that is also a GMM. The general approach is to minimise some distance $d(p, q)$ between the two PDFs. The functions considered here are the KL divergence and two norms of $p - q$; the infinity norm or the L^2 norm. We consider approximation of both properly normalised GMMs and unnormalised GMMs that may be encountered as so-called α-function in context of point-based value iteration Porta et al. (2006).

B.1 KL divergence

This section considers an approximation q that minimises the KL divergence $D(p, q)$ (see Equation (A.5)) between the original function p and q. Unfortunately, the KL divergence for two GMMs cannot be solved in closed form. Goldberger and Roweis (2005) present an alternative method using a

distance metric that minimises the weighted sum of KL divergences between individual components of p and q. The method attempts to identify an approximation q such that

$$d_{\mathrm{GR}}(p,q) = \sum_{i=1}^{n} w_i \min_{j\in\{1,2,\ldots,k\}} D(p_i, q_j) \tag{B.2}$$

is minimised. The approximating GMM q is initialised by including in it the k components with the largest weights from p. Regrouping and combination stages for components are then iteratively applied to modify q to minimise $d_{\mathrm{GR}}(p,q)$. For approximation of α-functions, the weighted sums of Gaussians must be normalised so that the weights sum to unity and are non-negative. After finding the approximation, the original weights are returned by rescaling.

B.2 Infinity norm

As the KL divergence is a quantification of the difference between two PDFs, it is not clear whether it is well suited for finding approximations to the α-functions which are unnormalised GMMs. An alternative that minimises the overall function approximation error was suggested by Brunskill et al. (2010), based on minimising the infinity norm

$$d_\infty(p,q) = ||p-q||_\infty = \sup_{x\in\mathcal{X}} |p(x) - q(x)|. \tag{B.3}$$

Minimising $d_\infty(p,q)$ is equivalent to minimising the worst case overall approximation error.

As Equation (B.3) cannot be evaluated in closed form for two GMMs, Brunskill et al. (2010) suggest a greedy method for minimising it. The domain $\mathcal{X}$ is sampled uniformly and p is evaluated at the sampled points, providing the initial approximation residuals. At iteration j, a Gaussian component q_j is inserted into q at the location of the greatest absolute value of the residual, with a variance estimated by the slope of the residual around this location. The weight c_j for the component is estimated based on the value of the residual at the current location. The residual is updated by subtracting from it the contribution of $c_j \cdot q_j$, and the process is repeated until k components have been inserted into q. The approximation can be directly applied to both normalised an unnormalised GMMs.

B.3 L^2 norm

Zhang and Kwok (2010) propose a mixture model approximation scheme based on minimising an upper bound on the L^2 distance between the original function and its approximation. They derive an upper bound

$$||p-q||_2^2 \leqslant d_{\mathrm{ZK}}(p,q) = k \sum_{j=1}^{k} \int_{x\in\mathcal{X}} \left(\sum_{i\in S_j} w_i p_i(x) - c_j q_j(x) \right)^2 \mathrm{d}x, \tag{B.4}$$

where $\{S_j\}_{j=1}^k$ is a partition of the n components of p into k disjoint sets. The approximation q that minimises (B.4) approximates the components in the set S_j by the jth component in q.

The approximation is found by an iterative procedure. An initial guess with a random partition is made. Steps of local approximation and regrouping of the components across the sets is then applied iteratively to minimise the upper bound.

To deal with positive and negative weights, Zhang and Kwok (2010) suggest splitting p into p_+ and p_- containing only the components with positive or negative weights, respectively. Then p_+ and p_- are approximated separately to form q_+ and q_-, and the results are then combined to obtain the overall approximation q.

C

A lazy sampling scheme for robotic exploration

This appendix presents a lazy sampling method for drawing observation samples in the context of the robotic exploration problem (Problem 3.7). The sampling scheme is presented assuming an occupancy grid map representation and a LRF sensor. The idea is to only propagate map occupancy information for the subset of cells that are observed by the sensor, saving computational resources. The sampling algorithm can be easily integrated with the POMCP (Algorithm 2.4) or SMC (Algorithm 2.2) planning algorithms.

Let M denote the set of cells in the occupancy grid map, and let $\tilde{M}(x_t^{(i)})$ denote the set of cells that can potentially be perceived by the robot's sensor when its pose is $x_t^{(i)}$. Given a search depth d at decision epoch t, Algorithm C.1 works by constructing step-by-step the observations $z_{t+1:t+d}^{(i)}$ as functions $z_t^{(i)} : \tilde{M}(x_t^{(i)}) \to \{1, 0, -1\}$ which assign to each potentially sensed cell one of three values, a hit (1), a miss (0), or indication that the cell was not observed (-1). We define a persistent map sample $m^p : M \to \mathbb{N} \times \{0, 1\}$ as a mapping from cells to a pair (k, o) consisting of an integer k denoting the time up to which the cell's state has been propagated, and a binary-valued cell state o, free (0) or occupied (1).

Algorithm C.1 A lazy sampling scheme.

Input: The POMDP model, current belief state $b_t = p(x_t, m_t)$, and a search depth d.

function SAMPLE(b_t, d)
  Sample $x_t^{(i)} \sim p(x_t)$
  for $k = t, \ldots t + d - 1$ **do**
    Sample $x_{k+1}^{(i)} \sim p(x_{k+1} \mid x_k^{(i)}, a_k)$
    Sample $z_{k+1}^{(i)} \sim p(z_{k+1} \mid x_{k+1}^{(i)}, a_k)$ (use e.g. RAYRACE
    Calculate $p(m_{k+1} \mid m_k, x_{k+1}^{(i)}, z_{k+1}^{(i)})$
  end for
  return $z_{t+1:t+d}^{(i)}$
end function

In a computer implementation, both observations and the persistent map sample can be efficiently represented as hash tables that are constructed and updated piece by piece. Initially, both are defined as empty.

A pose $x_t^{(i)}$ is sampled from the marginal PDF $p(x_t)$ obtained from the current belief state $b_t = p(x_t, m_t)$ (Line 2). At each step k, a new pose $x_{k+1}^{(i)}$ is sampled from $p(x_{k+1} \mid x_k^{(i)}, a_k)$ (Line 4). For the POMCP algorithm, the action a_k is obtained either from the tree policy or from the rollout policy. For the SMC algorithm, the action in the sequence of the current particle is applied.

A sensor-dependent method is used (Line 5) to sample an observation. For a LRF, the sample is drawn by applying the function RAYTRACE (Algorithm C.2). For each incidence angle α in which a LRF beam is sent from the current pose $x_{k+1}^{(i)}$, cells $c \in \tilde{M}(x_t^{(i)})$ which are on the path of the beam are found starting from the nearest. If the cell is not already in the persistent map, we set $m^p(c) = (k+1, o_c)$, sampling o_c from $p(c_{k+1})$, which is obtained by propagating initial information $p(c_t)$ from the belief $p(x_t, m_t)$ appropriately via the cell's state transition model (Line 5). If the cell is already in m^p, we propagate its state in m^p to time $(k+1)$ (Line 7).

Algorithm C.2 Raytracing method for sampling a LRF observation.

Input: The POMDP model, the map sample m^p, set of incidence angles $\{\alpha\}$, the belief state $b_t = p(x_t, m_t)$, and the current pose $x_{k+1}^{(i)}$.

```
1:  function RAYTRACE(m^p, {α}, b_t, x_{k+1}^(i))
2:      for all α ∈ {α} do
3:          for all cells c along a beam starting at x_k^(i) in direction α do
4:              if c ∉ m^p then
5:                  Set m^p(c) = (k + 1, o_c) with o_c ~ p(c_{k+1})
6:              else
7:                  Propagate m^p(c) to time (k + 1) via p(c_{t+1} | c_t)
8:              end if
9:              Sample z_{k+1}^(i)(c) ~ p(z_{k+1}(c) | m^p(c), x_k^(i))
10:             if z_{k+1}^(i)(c) = 1 then
11:                 break
12:             end if
13:         end for
14:     end for
15:     return z_{k+1}^(i)
16: end function
```

Once the state of the map in the cell c has been sampled at the current time step, we sample the observation for the particular cell as $z_{k+1}^{(i)}(c) \sim p(z_{k+1}(c) \mid o_c, x_k^{(i)})$ (Line 9). For a LRF sensor, $z_{k+1}^{(i)}(c) = 1$ indicates that the beam hit an object and the ray tracing for the current angle is terminated (Line 11).

Finally, once the observation sample is returned, we can compute the posterior belief over the map m_{k+1} (Algorithm C.1, Line 6). At this point, it is also convenient to calculate quantities such as map entropy or mutual information.

Bibliography

In the electronic version of the thesis, the highlighted parts in the bibliography contain hyperlinks to an online resource relevant to the reference if available.

Ahuja, R. K., Orlin, J. B., Pallottino, S., and Scutellà, M. G. (2003). Dynamic Shortest Paths Minimizing Travel Times and Costs. *Networks*, 41(4):197–205.

Alami, R., Chatila, R., Fleury, S., Ghallab, M., and Ingrand, F. (1998). An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337.

Albin, P. (2003). *Stokastiska Processer*. Studentlitteratur, Lund, Sweden.

Amigoni, F. and Caglioti, V. (2010). An Information-based Exploration Strategy for Environment Mapping with Mobile Robots. *Robotics and Autonomous Systems*, 58(5):684–699.

Araya-López, M., Buffet, O., Thomas, V., and Charpillet, F. (2010). A POMDP Extension with Belief-dependent Rewards. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 64–72, Vancouver, Canada.

Åström, K. J. (1969). Optimal Control of Markov Processes with Incomplete State-Information II. The Convexity of the Lossfunction. *Journal of Mathematical Analysis and Applications*, 26(2):403 – 406.

Åström, K. J. (1970). *Introduction to Stochastic Control Theory*. Academic Press, Inc., New York, NY, 1st edition.

Atanasov, N., Le Ny, J., Daniilidis, K., and Pappas, G. J. (2014a). Information Acquisition with Sensing Robots: Algorithms and Error Bounds. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 6447–6454, Hong Kong, China.

Atanasov, N., Sankaran, B., Le Ny, J., Pappas, G. J., and Daniilidis, K. (2014b). Nonmyopic View Planning for Active Object Classification and Pose Estimation. *IEEE Transactions on Robotics*, 30(5):1078–1090.

Athans, M. (1971). The Role and Use of the Stochastic Linear-quadratic-Gaussian Problem in Control System Design. *IEEE Transactions on Automatic Control*, 16(6):529–552.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256.

Bai, H., Cai, S., Ye, N., Hsu, D., and Lee, W. S. (2015). Intention-aware online POMDP planning for autonomous driving in a crowd. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 454–460, Seattle, WA.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, New Jersey.

Berry, D. A. and Fristedt, B. (1985). *Bandit Problems: Sequential Allocation of Experiments*. Chapman and Hall Ltd., London, UK.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, MA.

Bertsekas, D. P. (2001). *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, Belmont, MA, 2nd edition.

Bertsekas, D. P. (2005). Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334.

Bertsekas, D. P. and Castañon, D. A. (1999). Rollout Algorithms for Stochastic Scheduling Problems. *Journal of Heuristics*, 5:89–108.

Bertsekas, D. P. and Shreve, S. E. (1996). *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, Belmont, MA. Reprint of 1978 original.

Besse, C. and Chaib-draa, B. (2009). Quasi-Deterministic Partially Observable Markov Decision Processes. In Leung, C. S., Lee, M., and Chan, J. H., editors, *Neural Information Processing*, volume 5863 of *Lecture Notes in Computer Science*, pages 237–246. Springer.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, New York, NY.

Blanco, J., Fernandez-Madrigal, J., and Gonzalez, J. (2008). A Novel Measure of Uncertainty for Mobile Robot SLAM with Rao Blackwellized Particle Filters. *International Journal of Robotics Research*, 27(1):73–89.

Bnaya, Z., Felner, A., and Shimony, S. (2009). Canadian Traveler Problem with Remote Sensing. In *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 437–442, Pasadena, CA.

Bonet, B. (2009). Deterministic POMDPs Revisited. In *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 59–66, Montreal, Canada.

Bonet, B. and Geffner, H. (2009). Solving POMDPs: RTDP-Bel vs. Point-based Algorithms. In *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1641–1646, Pasadena, CA.

Brooks, A., Makarenko, A., Williams, S., and Durrant-Whyte, H. (2006). Parametric POMDPs for Planning in Continuous State Spaces. *Robotics and Autonomous Systems*, 54(11):887–897.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43.

Brunskill, E., Kaelbling, L. P., Lozano-Pérez, T., and Roy, N. (2010). Planning in Partially-observable Switching-mode Continuous Domains. *Annals of Mathematics and Artificial Intelligence*, 58(3-4):185–216.

Brzeźniak, Z. and Zastawniak, T. (1999). *Basic Stochastic Processes*. Springer-Verlag, London, UK.

Buşoniu, L., Babuška, R., De Schutter, B., and Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Taylor & Francis Group, LLC, Boca Raton, FL.

Carlone, L., Ng, M. K., Bona, B., and Indri, M. (2010). An application of Kullback-Leibler Divergence to Active SLAM and Exploration with Particle Filters. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 287–293, Taipei, Taiwan.

Cassandra, A., Littman, M., and Zhang, N. (1997). Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. In *Proc. of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 54–61, Providence, RI.

Chang, H. S., Givan, R., and Chong, E. K. P. (2004). Parallel Rollout for Online Solution of Partially Observable Markov Decision Processes. *Discrete Event Dynamic Systems: Theory and Applications*, 14(3):309–341.

Charrow, B., Kumar, V., and Michael, N. (2014). Approximate Representations for Multi-robot Control Policies that Maximize Mutual Information. *Autonomous Robots*, 37(4):383–400.

Charrow, B., Liu, S., Kumar, V., and Michael, N. (2015). Information-theoretic mapping using Cauchy-Schwarz Quadratic Mutual Information. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4791–4798, Seattle, WA.

Cheng, H.-T. (1988). *Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, University of British Columbia.

Chong, E. K. P., Kreucher, C. M., and Hero, A. O. (2009). Partially Observable Markov Decision Process Approximations for Adaptive Sensing. *Discrete Event Dynamic Systems*, 19(3):377–422.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. John Wiley & Sons, Inc., Hoboken, NJ.

DeGroot, M. H. (2004). *Optimal Statistical Decisions*. John Wiley & Sons, Inc., Hoboken, New Jersey. Wiley Classics Library edition.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo Samplers. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 68(3):411–436.

Dreyfus, S. D. (1965). *Dynamic Programming and the Calculus of Variations*. Academic Press, Inc., New York, NY.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous Localization and Mapping: Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110.

Eddy, S. R. (1996). Hidden Markov Models. *Current Opinion in Structural Biology*, 6(3):361 – 365.

Emery-Montemerlo, R. (2005). *Game-Theoretic Control for Robot Teams*. PhD thesis, The Robotics Institute, Carnegie Mellon University.

Endo, G., Morimoto, J., Matsubara, T., Nakanishi, J., and Cheng, G. (2008). Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot. *The International Journal of Robotics Research*, 27(2):213–228.

Eyerich, P., Keller, T., and Helmert, M. (2010). High-Quality Policies for the Canadian Traveler's Problem. In *Third Annual Symposium on Combinatorial Search*, pages 51–58, Atlanta, GA, USA.

Feinberg, E., Kasyanov, P., and Zgurovsky, M. (2013). Optimality Conditions for Total-cost Partially Observable Markov Decision Processes. In *Proc. of the 52nd Annual Conference on Decision and Control (CDC)*, pages 5716–5721, Florence, Italy.

Feinberg, E. A. (2002). Total reward criteria. In Feinberg, E. A. and Shwartz, A., editors, *Handbook of Markov Decision Processes*. Springer Science + Business Media, New York, NY. Originally by Kluwer Academic Publishers in 2002.

Foka, A. and Trahanias, P. (2007). Real-time hierarchical POMDPs for autonomous robot navigation. *Robotics and Autonomous Systems*, 55(7):561–571.

French, S. and Ríos Insua, D. (2000). *Statistical Decision Theory*. Arnold Publishers, London, UK.

Fujishige, S. (2005). *Submodular Functions and Optimization*, volume 58 of *Annals of discrete mathematics*. Elsevier B.V., Amsterdam, Netherlands, 2nd edition.

Geffner, H. and Bonet, B. (1998). Solving Large POMDPs using Real Time Dynamic Programming. In *Proceedings of the AAAI Fall Symposium on POMDPs*, pages 61–68, Orlando, FL.

Gittins, J., Glazebrook, K., and Weber, R. (2011). *Multi-armed Bandit Allocation Indices*. John Wiley & Sons, Ltd., Chichester, UK, 2nd edition.

Gittins, J. C. (1979). Bandit Processes and Dynamic Allocation Indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177.

Goldberger, J. and Roweis, S. (2005). Hierarchical Clustering of a Mixture Model. In *Advances in Neural Information Processing Systems 18*, volume 17, pages 505–512, Vancouver, Canada.

Goldman, C. V. and Zilberstein, S. (2004). Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis. *Journal of Artificial Intelligence Research*, 22:143–174.

Golovin, D. and Krause, A. (2011). Adaptive Submodularity: A New Approach to Active Learning and Stochastic Optimization. *Journal of Artificial Intelligence Research*, 42:427–486.

Grimmett, G. R. and Stirzaker, D. R. (2001). *Probability and Random Processes*. Oxford University Press, Inc., New York, NY, 3rd edition.

Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46.

Hansen, E. A. (1998a). An Improved Policy Iteration Algorithm for Partially Observable MDPs. In Kearns, M. J., Solla, S. A., and Cohn, D. A., editors, *Advances in Neural Information Processing Systems 11*, pages 1015–1021, Denver, CO.

Hansen, E. A. (1998b). Solving POMDPs by Searching in Policy Space. In *Proc. of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, Madison, WI.

Hansen, E. A. (2007). Indefinite-Horizon POMDPs with Action-Based Termination. In *Proc. of the 22nd National Conference on Artificial Intelligence*, pages 1237–1242, Vancouver, Canada.

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic Programming for Partially Observable Stochastic games. In *Proc. of the 19th National Conference on Artificial Intelligence (AAAI)*, pages 709–715, San Jose, CA.

Hauskrecht, M. (2000). Value-function Approximations for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 13(1):33–94.

Hero, A. and Cochran, D. (2011). Sensor Management: Past, Present, and Future. *IEEE Sensors Journal*, 11(12):3064–3075.

Hitz, G., Gotovos, A., Eve, M., Krause, A., and Siegwart, R. Y. (2014). Fully Autonomous Focused Exploration for Robotic Environmental Monitoring. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2658–2664, Hong Kong, China.

Hoey, J. and Poupart, P. (2005). Solving POMDPs with Continuous or Large Discrete Observation Spaces. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1332–1338, Edinburgh, Scotland.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. The Technology Press of The Massachusetts Institute of Technology and John Wiley & Sons, Inc., New York, NY.

Ji, S., Parr, R., Li, H., Liao, X., and Carin, L. (2007). Point-based Policy Iteration. In *Proc. of the 22nd National Conference on Artificial Intelligence (AAAI)*, pages 1243–1249, Vancouver, Canada.

Johansen, A. M., Doucet, A., and Davy, M. (2008). Particle Methods for Maximum Likelihood Estimation in Latent Variable Models. *Statistics and Computing*, 18(1):47–57.

Kaelbling, L., Littman, M., and Cassandra, A. (1998). Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1-2):99–134.

Kantas, N., Maciejowski, J., and Lecchini-Visintini, A. (2009). Sequential Monte Carlo for Model Predictive Control. In Magni, L., Raimondo, D., and Allgöwer, F., editors, *Nonlinear Model Predictive Control*, volume 384 of *Lecture Notes in Control and Information Sciences*, pages 263–273. Springer, Berlin; Heidelberg, Germany.

Khatib, O. (1986). Real Time Obstacle Avoidance for Manipulators and Mobile Robots. *International Journal of Robotics and Research*, 5(1):90–98.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Kocsis, L. and Szepesvári, C. (2006). Bandit Based Monte-Carlo Planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Proc. of the 17th European Conference on Machine Learning (ECML)*, pages 282–293, Berlin, Germany.

Koenig, S. (2001). Agent-centered Search. *AI Magazine*, 22(4):109–131.

Kollar, T. and Roy, N. (2008). Trajectory Optimization using Reinforcement Learning for Map Exploration. *International Journal of Robotics Research*, 27(2):175–196.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-Optimal Sensor Placements in Gaussian Processes-Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284.

Krishnamurthy, V. and Evans, R. J. (2001). Hidden Markov Model Multiarm Bandits: a Methodology for Beamscheduling in Multitarget Tracking. *IEEE Transactions on Signal Processing*, 49(12):2893–2908.

Krishnamurthy, V. and Wahlberg, B. (2009). Partially Observed Markov Decision Process Multiarmed Bandits – Structural Results. *Mathematics of Operations Research*, 34(2):287–302.

Kuipers, B., Modayil, J., Beeson, P., MacMahon, M., and Savelli, F. (2004). Local Metrical and Global Topological Maps in the Hybrid Spatial Semantic Hierarchy. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4845–4851, New Orleans, LA.

Kurniawati, H., Hsu, D., and Lee, W. (2008). SARSOP: Efficient Point-based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In Brock, O., Trinkle, J., and Ramos, F., editors, *Proc. Robotics: Science and Systems*, Zürich, Switzerland.

Lauri, M., Atanasov, N., Pappas, G., and Ritala, R. (2015). Active Object Recognition via Monte Carlo Tree Search. In *ICRA Workshop: Beyond Geometric Constraints: Planning for Solving Complex Tasks, Reducing Uncertainty, and Generating Informative Paths & Policies*, Seattle, WA.

Lauri, M., Atanasov, N., Pappas, G., and Ritala, R. (2016). Myopic Policy Bounds for Information Acquisition POMDPs. Submitted to *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden.

Lauri, M. and Ritala, R. (2012). Receding Horizon Control for Selection of Focus of Attention. In *Proc. of the 20th IMEKO World Congress*, Busan, Republic of Korea.

Lauri, M. and Ritala, R. (2013a). Path Planning in Dynamic Environments with the Partially Observable Canadian Traveller's Problem. In *Proc. of the 1st Workshop on Planning and Robotics (PlanRob)*, pages 89–95, Rome, Italy.

Lauri, M. and Ritala, R. (2013b). Planning for Multiple Measurement Channels in a Continuous-state POMDP. *Annals of Mathematics and Artificial Intelligence*, 67:283–317.

Lauri, M. and Ritala, R. (2014). Stochastic Control for Maximizing Mutual Information in Active Sensing. In *ICRA Workshop on Robots in Homes and Industry: Where to Look First?*, Hong Kong, China.

Lauri, M. and Ritala, R. (2015a). Optimal Sensing via Multi-armed Bandit Relaxations in Mixed Observability Domains. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4807–4812, Seattle, WA.

Lauri, M. and Ritala, R. (2015b). Planning for Robotic Exploration Based on Forward Simulation. arXiv:1502.02474. Preprint.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, UK.

Likhachev, M. and Stentz, A. (2009). Probabilistic Planning with Clear Preferences on Missing Information. *Artificial Intelligence*, 173(5-6):696–721.

Littman, M. L. (1996). *Algorithms for Sequential Decision Making*. PhD thesis, Brown University.

Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning Policies for Partially Observable Environments: Scaling up. In *Proc. of the 12th International Conference on Machine Learning (ICML)*, Tahoe City, CA.

Liu, J. and Chen, R. (1998). Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association*, 93(443):1032–1044.

Lovejoy, W. (1991a). A Survey of Algorithmic Methods for Partially Observed Markov Decision Processes. *Annals of Operations Research*, 28(1):47–65.

Lovejoy, W. (1991b). Computationally Feasible Bounds for Partially Observed Markov Decision Processes. *Operations research*, 39(1):162–175.

Lovejoy, W. S. (1987). Some Monotonicity Results for Partially Observed Markov Decision Processes. *Operations Research*, 35(5):736–743.

Maciejowski, J. M. (2002). *Predictive Control with Constraints*. Pearson Education Ltd., Harlow, UK.

MacLachlan, G. and Peel, D. (2000). *Finite Mixture Models*. John Wiley & Sons.

Madani, O., Hanks, S., and Condon, A. (1999). On the Undecidability of Probabilistic Planning and Infinite-horizon Partially Observable Markov Decision Problems. In *Proc. of the 16th National Conference on Artificial Intelligence (AAAI)*, pages 541–548, Orlando, FL.

Madani, O., Hanks, S., and Condon, A. (2003). On the Undecidability of Probabilistic Planning and Related Stochastic Optimization Problems. *Artificial Intelligence*, 147(1–2):5–34.

Mahajan, A. and Teneketzis, D. (2008). Multi-armed Bandit Problems. In Hero, A. O., Castañón, D., Cochran, D., and Kastella, K., editors, *Foundations and Applications of Sensor Management*. Springer Science + Business Media, New York, NY.

Martinez-Cantin, R., De Freitas, N., Brochu, E., Castellanos, J., and Doucet, A. (2009). A Bayesian Exploration-exploitation Approach for Optimal Online Sensing and Planning with a Visually Guided Mobile Robot. *Autonomous Robots*, 27(2):93–103.

McAllester, D. A. and Singh, S. (1999). Approximate Planning for Factored POMDPs Using Belief State Simplification. In Laskey, K. B. and Prade, H., editors, *Proc. of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 409–416, Stockholm, Sweden.

Meier, L., Peschon, J., Dressler, R., and Tuel, W. G. (1967). Optimal Control of Measurement Subsystems. *IEEE Transactions on Automatic Control*, 12(5):528–536.

Melin, J., Lauri, M., Kolu, A., Koljonen, J., and Ritala, R. (2015). Cooperative Sensing and Path Planning in a Multi-vehicle Environment. In *IFAC Workshop on Advanced Control and Navigation for Autonomous Aerospace Vehicles (ACNAAV)*, pages 198–203, Seville, Spain.

Meyer-Delius, D., Beinhofer, M., and Burgard, W. (2012). Occupancy Grid Models for Robot Mapping in Changing Environments. In *Proc. of the 26th National Conference on Artificial Intelligence (AAAI)*, pages 2024–2030, Toronto, Canada.

Monahan, G. (1982). A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms. *Management Science*, 28(1):1–16.

Monso, P., Alenya, G., and Torras, C. (2012). POMDP approach to robotized clothes separation. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1324–1329, Vilamoura, Algarve, Portugal.

Moran, W., Suvorova, S., and Howard, S. (2008). Application of Sensor Scheduling Concepts to Radar. In Hero, A. O., Castañón, D., Cochran, D., and Kastella, K., editors, *Foundations and Applications of Sensor Management*. Springer Science + Business Media, New York, NY.

Moravec, H. P. (1988). Sensor Fusion in Certainty Grids for Mobile Robots. *AI Magazine*, 9(2):61–74.

Neapolitan, R. E. (2004). *Learning Bayesian Networks*. Pearson Education, Inc., Upper Saddle River, NJ.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An Analysis of Approximations for Maximizing Submodular Set Functions - I. *Mathematical Programming*, 14(1):265–294.

von Neumann, J. and Morgenstern, O. (1953). *Theory of Games and Economic Behaviour*. Princeton University Press, Princeton, New Jersey.

Ng, A. Y. and Jordan, M. (2000). PEGASUS: A Policy Search Method for Large MDPs and POMDPs. In *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 406–415, Stanford, CA.

Nikandrova, E., Laaksonen, J., and Kyrki, V. (2014). Towards informative sensor-based grasp planning. *Robotics and Autonomous Systems*, 62(3):340–354.

Oliehoek, F. A. (2010). *Value-Based Planning for Teams of Agents in Stochastic Partially Observable Environments*. PhD thesis, Informatics Institute, University of Amsterdam.

Ong, S. C. W., Png, S. W., Hsu, D., and Lee, W. (2010). Planning under Uncertainty for Robotic Tasks with Mixed Observability. *International Journal of Robotics Research*, 29(8):1053–1068.

Pajarinen, J. and Kyrki, V. (2015). Robotic manipulation of multiple objects as a POMDP. *Artificial Intelligence*, pages –. (In press).

Papadimitriou, C. H. (1985). Games against Nature. *Journal of Computer and System Sciences*, 31(2):288 – 301.

Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12:441–450.

Papadimitriou, C. H. and Yannakakis, M. (1991). Shortest Paths without a Map. *Theoretical Computer Science*, 84(1):127–150.

Paquet, S., Chaib-Draa, B., and Ross, S. (2006). Hybrid POMDP Algorithms. In *Proc. of the Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, pages 133–147.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Pineau, J., Gordon, G., and Thrun, S. (2006). Anytime Point-based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research*, 27(1):335–380.

Pineau, J., Montemerlo, M., Pollack, M., Roy, N., and Thrun, S. (2003). Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3-4):271–281.

Platt, R., Tedrake, R., Kaelbling, L., and Lozano-Perez, T. (2010). Belief space planning assuming maximum likelihood observations. In *Proc. Robotics: Science and Systems*, Zaragoza, Spain.

Porta, J., Vlassis, N., Spaan, M., and Poupart, P. (2006). Point-based Value Iteration for Continuous POMDPs. *Journal of Machine Learning Research*, 7:2329–2367.

Poupart, P. (2005). *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, Department of Computer Science, University of Toronto.

Poupart, P. and Boutilier, C. (2002). Value-directed Compression of POMDPs. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1547–1554, Vancouver, Canada.

Psaraftis, H. and Tsitsiklis, J. (1993). Dynamic Shortest Paths in Acyclic Networks with Markovian Arc Costs. *Operations Research*, 41(1):91–101.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an Open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

Raiffa, H. and Schlaifer, R. (2000). *Applied Statistical Decision Theory*. John Wiley & Sons, Inc., Hoboken, New Jersey. Wiley Classics Library edition.

Robert, C. P. and Casella, G. (1999). *Monte Carlo Statistical Methods*. Springer-Verlag, Inc., New York, NY.

Ross, S. (1983). *Introduction to Stochastic Dynamic Programming*. Academic Press, Inc., New York, NY.

Ross, S. and Chaib-draa, B. (2007). AEMS: An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs. In *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2592–2598, Hyderabad, India.

Ross, S., Chaib-draa, B., and Pineau, J. (2007). Bayes-adaptive POMDPs. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, pages 1225–1232, Vancouver, Canada.

Ross, S., Pineau, J., Paquet, S., and Chaib-Draa, B. (2008). Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32(1):663–704.

Roy, N., Gordon, G., and Thrun, S. (2005). Finding Approximate POMDP Solutions through Belief Compression. *Journal of Artificial Intelligence Research*, 23(1):1–40.

Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc. (As Prentice Hall), Upper Saddle River, NJ.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press, Cambridge, UK.

Satia, J. K. and Lave, R. E. (1973). Markovian Decision Processes with Probabilistic Observation of States. *Management Science*, 20(1):1–13.

Satsangi, Y., Whiteson, S., and Oliehoek, F. (2015). Exploiting submodular value functions for faster dynamic sensor selection. In *Proc. of the 29th National Conference on Artificial Intelligence (AAAI)*, Austin, TX.

Shani, G., Pineau, J., and Kaplow, R. (2013). A Survey of Point-based POMDP Solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51.

Silver, D. and Veness, J. (2010). Monte-Carlo Planning in Large POMDPs. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 2164–2172, Vancouver, Canada.

Sim, R. and Roy, N. (2005). Global A-Optimal Robot Exploration in SLAM. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 661–666, Barcelona, Spain. IEEE.

Smallwood, R. and Sondik, E. (1973). The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Operations Research*, 21(5):1071–1088.

Smith, T. and Simmons, R. (2004). Heuristic Search Value Iteration for POMDPs. In Meek, C. and Halpern, J., editors, *Proc. of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 520–527, Banff, Canada.

Smith, T. and Simmons, R. (2005). Point-based POMDP Algorithms: Improved Analysis and Implementation. In Bacchus, F. and Jaakkola, T., editors, *Proc. of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 542–549, Edinburgh, Scotland.

Somani, A., Ye, N., Hsu, D., and Lee, W. S. (2013). Despot: Online pomdp planning with regularization. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 26*, pages 1772 – 1780. Curran Associates, Inc.

Sondik, E. J. (1971). *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University.

Sondik, E. J. (1978). The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs. *Operations Research*, 26(2):282 – 304.

Spaan, M., Veiga, T., and Lima, P. (2010). Active cooperative perception in network robot systems using POMDPs. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4800 – 4805, Taipei, Taiwan.

Spaan, M. and Vlassis, N. (2005). Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24(1):195 – 220.

Spaan, M. T. J., Veiga, T. S., and Lima, P. U. (2015). Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems*, 29(6):1157 – 1185.

Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information Gain-based Exploration using Rao-Blackwellized Particle Filters. In *Proc. Robotics: Science and Systems (RSS)*, Cambridge, MA, USA.

Strom, D. P., Nenci, F., and Stachniss, C. (2015). Predictive Exploration Considering Previously Mapped Environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2761 – 2766, Seattle, WA, USA.

Thrun, S. (2000). Monte Carlo POMDPs. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 12*, pages 1064 – 1070, Denver, CO.

Thrun, S., Burgard, W., and Fox, D. (2006). *Probabilistic Robotics*. The MIT Press, Cambridge, MA.

Tversky, A. and Kahneman, D. (1974). Judgment Under Uncertainty: Heuristics and Biases. *Science*, 185(4157):1124 – 1131.

Varaiya, P., Walrand, J., and Buyukkoc, C. (1985). Extensions of the Multiarmed Bandit Problem: The Discounted Case. *IEEE Transactions on Automatic Control*, 30(5):426 – 439.

Warnquist, H., Kvarnström, J., and Doherty, P. (2013). Exploiting Fully Observable and Deterministic Structures in Goal POMDPs. In *Proc. of the 23rd International Conference on Automated Planning and Scheduling, (ICAPS)*, pages 242 – 250, Rome, Italy.

Washburn, R. B. (2008). Application of Multi-Armed Bandits to Sensor Management. In Hero, A. O., Castañón, D., Cochran, D., and Kastella, K., editors, *Foundations and Applications of Sensor Management*. Springer Science + Business Media, New York, NY.

Washington, R. (1997). BI-POMDP: Bounded, Incremental Partially-Observable Markov-Model Planning. In *Proc. of the 4th European Conference on Planning*, pages 440 – 451, Toulouse, France.

Zhang, K. and Kwok, J. T. (2010). Simplifying Mixture Models Through Function Approximation. *IEEE Transactions on Neural Networks*, 21(4):644–658.

Zhou, E., Fu, M. C., and Marcus, S. I. (2010). Solving Continuous-state POMDPs via Density Projection. *IEEE Transactions on Automatic Control*, 55(5):1101–1116.

Zhou, R. and Hansen, E. A. (2001). An Improved Grid-based Approximation Algorithm for POMDPs. In *Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 707–716, Seattle, WA.

Zilberstein, S. (1996). Using Anytime Algorithms in Intelligent Systems. *AI magazine*, 17(3):73–83.

Index

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-3676-2
ISSN 1459-2045