Information Gathering and Reward Exploitation of Subgoals for POMDPs

Hang Ma

Master of Science

School of Computer Science

McGill University

Montreal,Quebec

August 10, 2014

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

©Hang Ma, 2014

DEDICATION

To my parents and friends.

ACKNOWLEDGEMENTS

During my master's studies at McGill University, I feel lucky to have the chance to work in the Reasoning and Learning Lab. I am pleased and fortunate to have so many people supporting my study and research, and to whom I would always like to express my gratitude. I appreciate all of them for making this thesis possible.

First and foremost, I would like to express my deepest gratitude to my supervisor, Joelle Pineau, for providing the scholarship to support my graduate study and her most insightful guidance on my path of academic research. I have enjoyed the weekly meetings with her, where I have been inspired by her enthusiasm for research and great knowledge of computer science. She always encourages me to explore my own ideas and research topics, and continuously provides invaluable advice and encouragement to keep me on the right track. And her guidance has helped me improve my research and writing skills, and has further prepared me to be a better researcher. I have therefore been motivated to pursue PhD studies in computer science. I am most grateful for her time and efforts spent on reading and editing this thesis.

I gratefully acknowledge my thesis external examiner, Olivier Buffet, for contributing his time to review this thesis, and providing constructive comments and suggestions to improve the quality of this thesis.

I would also like to thank many researchers who provided their help and software packages for the experiments of this work, in particular Hanna Kurniawati, Blai Bonet and Guy Shani, and several other researchers I met in different universities who provided advice for my graduate study and research.

I would also like to thank Professor Luc Devroye and Doina Precup, from whom I have received helpful advice at different stages of my graduate study and academic research. It has always been a great pleasure to talk with Luc because of his humour and great knowledge of mathematics and computer science. Doina's Machine Learning course has introduced me to the fields of machine learning and artificial intelligence.

I would like to thank all members of Reasoning and Learning Lab, and it has been a memorable experience for us to share ideas and have fun together. We have been colleagues as well as good friends. I especially thank Gabriel Forgues for helping me translate the abstract of this thesis into French, and William Hamilton and Boyu Wang for reading my papers and providing helpful comments. I would also like to thank Gheorghe Comanici, who was really helpful when I had difficulty setting up the experimental environment on our lab machines.

I thank all my friends and schoolmates at McGill University for the wonderful times we shared, especially Mingzhou Yang who I know so well since we were also from the same bachelor's program, and Haowei Shi and Jinxu Jia with whom I took several courses together and had joy after school, and many others in the beautiful city of Montreal.

Finally, I would like to express my deep and sincere gratitude to my parents, for their continuous love which makes me who I am, and the encouragement and support for my studies. I wish to dedicate this thesis to them.

ABSTRACT

Partially observable Markov decision processes (POMDPs) have emerged as a principled framework for planning and decision making under uncertainty. Planning in large POMDPs is challenging especially when information-gathering efforts or long planning horizons are required. A few recent algorithms successfully tackle one of the above two cases but at the expense of a weaker capacity of tackling another based on the notion of point-based value iteration. To bridge the gap between the two classes of point-based approaches, this thesis proposes *Information Gathering and Reward Exploitation of Subgoals* (IGRES), a randomized POMDP planning algorithm that leverages information in the state space to automatically generate "macro-actions" that can tackle tasks with long planning horizons, while locally exploring the belief space to allow effective information gathering. Experimental results show that IGRES is an effective multi-purpose POMDP solver, providing state-of-the-art performance for both long horizon planning tasks and information-gathering tasks on benchmark domains. The successful application to a new challenge task indicates that IGRES is a promising tool for POMDP planning in real-world settings.

ABRÉGÉ

Les processus de décision Markoviens partiellement observables (POMDP) ont emergé comme un cadre pour la planification et la prise de decision dans l'incertitude. La planification dans les POMDPs de grande taille est difficile, en particulier lorsqu'une collecte d'information ou un grand horizon de planification est nécessaire. Certains algorithmes récents s'attaquent à un de ces deux cas, mais seulement en échange d'une faiblesse dans l'autre cas. Pour combler le manque entre les deux types d'approches, cette thése propose la collecte d'information et l'exploitation de récompense pour des sous-objectifs (IGRES), un algorithme de planification de POMDP randomisé qui exploite l'information dans l'espace d'états afin de générer automatiquement des "macro-actions" qui peuvent s'attaquer aux tâches à grand horizon de planification, tout en explorant localement l'espace de croyances, ce qui permet une collecte d'information efficace. Les résultats expérimentaux démontrent qu'IGRES est un solveur POMDP efficace et polyvalent qui produit une performance à l'état de l'art, à la fois pour les tâches de planification à grand horizon et pour les tâches de collecte d'information sur des domaines références. L'application avec succés à une nouvelle tâche indique qu'IGRES est un outil prometteur pour la planification POMDP pour une mise en application en situation réelle.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

In artificial intelligence, we consider complex situations where the intelligent agents perform actions in an uncertain and dynamic environment. One real-world example is an autonomous robot with noisy sensors and stochastic actions navigates, who needs to overcome the limited observability to act optimally and finish certain tasks. This sort of scenarios are often modeled as *partially observable Markov decision processes* (POMDPs).

In this chapter, we present a brief overview of the POMDP framework, various POMDP solutions, our contributions, and the outline of this thesis.

1.1 POMDP Framework

Partially observable Markov decision processes (POMDPs) have emerged as a principled mathematical framework for planning and decision making under uncertainty. In the POMDP framwork, partial observability is allowed and uncertainty of state information is modeled through observations and beliefs. The POMDP framework provides the capacities to capture a number of important planning aspects that appear in many real-world sequential decision tasks, such as the ability to handle stochastic actions, missing or noisy observations, and stochastic costs and rewards.

The POMDP framework is general enough for modeling a wide range of real-world sequential decision tasks. In practice, it has already been widely applied to

various complex situations [4], including mechanical grasping tasks [13], medical diagnosis [10], intelligent medical devices [22], assistance for people with disabilities [12], spoken dialogue systems [26], ecological adaptive management [7] etc.

1.2 POMDP Solutions

Despite the mathematical expressivity of the POMDP framework, solving large POMDPs is computational intractable [17] and their application in complex domains is also very limited due to two kinds of difficulty:

- For a POMDP problem modelled with n states, we must reason in an $(n-1)$-dimensional continuous belief space, which is called the "curse of dimensionality" [14].
- The complexity of POMDP planning also suffers an exponential increase with the length of planning horizon, which is the "curse of history" [23].

Because of the above challenges, different methods have been explored to refine the classic value iteration algorithm. Exact solutions with a variety of pruning strategies have been proposed to reduce the complexity [31, 28, 19, 8, 3, 36]. However, exact POMDP planning is in fact PSPACE-complete [17]. Thus, many small domains with limited number of states, actions and observations are computationally intractable, let alone applying the framework in real-world scenarios. Practically speaking, solving real-world tasks in this framework involves two challenges:

- How to carry out intelligent information gathering in a large high dimensional belief space. For example, robot navigation in a practical scenario involves dealing with complicated inner systems of the robot and a huge set of external states such as various types of topography, which results in a POMDP model

with large state space. However, it is often the case that only a small part of the information might be substantially helpful for finishing the goal. Thus, it is crucial for an algorithm to be able to gather useful information efficiently in a high dimensional continuous belief space for POMDP planning in a real-world situation.

- How to scale up planning with long sequences of actions and delayed rewards. Solving a real-world problem, such as a long-term medical treatment, adaptive management over a long period, navigation in a large map etc., often requires that the intelligent agent has the ability to reason about long sequences of actions and delayed rewards in order to obtain an optimal effect.

Researches have recently focused on approximate solutions to tackle the above challenges due to the limited scalability of the exact methods. Over the past decade, one set of successful approaches [23, 29, 30, 33, 27, 16] have been proposed relying on the notion of point-based value iteration. The latest point-based planning algorithms have made impressive improvements by tending to address one or the other of the above practical challenges, and can compute policies for problems with up to $100,000$ states or performs well in domains that have delayed reward after long sequences of actions. However, we still lack methods that can tackle problems of both types, i.e. substantial information gathering in a large state space and long planning horizons.

1.3 Contributions

In this thesis, we propose a POMDP solution method, called *Information Gathering and Reward Exploitation of Subgoals* (IGRES), that effectively tackles both

challenges by incorporating elements from the two families of point-based approaches: first, IGRES identifies potentially important states as subgoals and leverages insight of MiGS[15] algorithm to exploiting state structure in order to generate macro-actions for transitions to subgoals; second, IGRES only gathers information and exploits rewards with macro-actions in the neighbourhood of those subgoals, which avoids sampling unnecessary beliefs. Thus, IGRES is efficient in terms of computational time and space in the sense that it covers the belief space well with a much smaller size of belief points set for expensive backup operations while still maintaining good performance.

Promising experimental results show that IGRES outperforms state-of-the-art POMDP solvers on tasks that require significant information gathering and planning with long sequences of actions. We also show how IGRES can effectively tackle a new ecological adaptive management problem [20] in addition to the classical domains, thus providing evidence that IGRES is capable of solving tasks in useful real-world settings.

1.4 Outline

This thesis is organized as follows: In Chapter 2, we introduce the technical background on POMDPs; In Chapter 3, we present the motivation and cover technical details for IGRES, followed by the data structure used in the algorithm and analysis of some theoretical properties; The experimental results are reported in Chapter 4; Finally, we conclude with a discussion of our approach and the possible future work.

CHAPTER 2
Technical Background

Let us imagine an intelligent agent navigating in a complex environment where it could not fully localize itself. Instead, it keeps receiving observations from sensors and maintains a sufficient statistics of the history in order to choose actions that can maximize the total reward. A general framework for this kind of situations is partially observable Markov decision processes (POMDPs).

In this chapter, we introduce POMDPs as the general mathematical framework for planning and decision making under uncertainty. We cover the basic terminology and concepts within the scope of the POMDP framwork. Then we present the definition of POMDP planning and approaches of solving the problem. Finally we discuss the well-known and state-of-the-art algorithms that have been developed.

2.1 POMDP Framework

A partially observable Markov decision process (POMDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \Omega, T, O, R, b_0, \gamma \rangle$ [14], where

- $\mathcal{S}$ is a finite set of states of the environment, which model all possible configurations of the world.
- $\mathcal{A}$ is a finite set of actions which the agent can make in the environment.
- Ω is a finite set of observations about the environment which the agent can receive.

- $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ is the *state-transition function* mapping a state and an action to a probability distribution over states. $T(s, a, s') = Pr(s'|s, a)$ specifies the probability of transitioning to state s' from the current state s by taking action a. Thus the transitions are Markovian in the sense that the probability of reaching state s' from s depends only on the current state s but not on any of the earlier states.
- $O : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\Omega)$ is the observation function that maps a state and an action to a probability distribution over possible observations. $O(s', a, o) = Pr(o|s', a)$ specifies the probability that the agent observes observation o when it moves to state s' by taking action a.
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, where $R(s, a)$ specifies the reward for taking a in the current state s. The reward function models the prizes or costs by performing actions in the environment.
- b_0 is the initial belief state.
- $\gamma \in [0, 1)$ is the discount factor.

According to the framework, the agent takes an action $a_t \in \mathcal{A}$ at each time step t at the current state s_t, and transitions from s_t to s_{t+1}. The post-action state s_{t+1} is modeled with uncertainty as a conditional probability according to the transition function $T(s_t, a_t, s_{t+1}) = Pr(s_{t+1}|s_t, a_t)$.

In a POMDP, the current state s_t of the environment is a hidden variable which cannot be observed directly by the agent. Instead, at each time step, it receives an observation $o \in \Omega$ at the post-action state s_{t+1}. The observation $o \in \Omega$ is modeled

with uncertainty as a conditional probability according to the observation function $O(s_{t+1}, a_t, o) = Pr(o|s_{t+1}, a_t)$.

The agent receives reward $R(s_t, a_t)$ by taking action a_t at current state s_t. The desired optimality of the agent is to take a sequence of actions wisely in order to maximize its sum of total rewards gained. We limit our discussion to infinite-horizon POMDPs where the expected sum of discounted total rewards is

$$E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]. \tag{2.1}$$

2.2 Belief State

Since the agent is acting in an environment where the current state is not fully observable, it has to keep an internal *belief state* about the environment. A belief state $b \in \mathcal{B}$ is a probability distribution over all states of the environment. It is a sufficient statistic for the past history $h_t = \{a_0, o_1, \ldots, o_{t-1}, a_{t-1}, o_t\}$ in the sense that, given the properly computed current belief state of the agent, no additional data about its past actions or observations would provide any further information about the current state of the world [1, 28], which also means that the process over belief states is Markov, and that no additional data about the past history would help to increase the expected reward [14]. Thus, given the initial belief state b_0 and the entire past history h_t, the belief state $b_t(s)$ at time step t is defined as the posterior probability distribution assigned to state s:

$$b(s) = Pr(s_t = s|b_0, h_t). \tag{2.2}$$

The agent starts with its initial knowledge about the environment denoted as the initial belief state b_0. Then the new belief state $b' = b^{a,o}$ can be computed after taking action a and receiving observation o through the belief update function as follows:

$$\begin{aligned} b'(s') &= Pr(s'|o,a,b) \\ &= \frac{Pr(o|s',a,b)Pr(s'|a,b)}{Pr(o|a,b)} \\ &= \frac{Pr(o|s',a)\sum_{s\in\mathcal{S}} Pr(s'|a,b,s)Pr(s|a,b)}{Pr(o|a,b)} \\ &= \frac{O(s',a,o)\sum_{s\in\mathcal{S}} T(s,a,s')b(s)}{Pr(o|a,b)}. \end{aligned} \tag{2.3}$$

The denominator, $Pr(o|a,b) = \sum_{s\in\mathcal{S}} b(s) \sum_{s'\in\mathcal{S}} T(s,a,s')O(s',a,o)$, can be treated as a normalizing factor, independent of s', that makes sure $\sum_{s\in\mathcal{S}} b(s) = 1$.

2.3 POMDP Planning

After the agent receives an observation and updates its belief state, it must choose to take an action based on the current belief state. In POMDP, a policy $\pi : \mathcal{B} \to \mathcal{A}$ is a mapping from the current belief state $b \in \mathcal{B}$ to an action $a \in \mathcal{A}$.

A value function $\mathcal{V}_\pi(b)$ specifies the expected reward gained starting from b followed by policy π:

$$\mathcal{V}_\pi(b) = \sum_{s\in\mathcal{S}} b(s)R(s,\pi(b)) + \gamma \sum_{o\in\Omega} p(o|b,\pi(b))\mathcal{V}_\pi(b^{\pi(b),o}). \tag{2.4}$$

The goal of POMDP planning is to find an optimal policy π^* where its value function $\mathcal{V}^* = \mathcal{V}_{\pi^*}$ is maximized:

$$\mathcal{V}_\pi(b) = \max_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} b(s) R(s,a) + \gamma \sum_{o \in \Omega} p(o|b,a) \mathcal{V}_\pi(b^{a,o}) \right], \tag{2.5}$$

and thus the optimal policy π^* is defined as

$$\pi^*(b) = \operatorname*{argmax}_{a \in \mathcal{A}} \left[\sum_{s \in \mathcal{S}} b(s) R(s,a) + \gamma \sum_{o \in \Omega} p(o|b,a) \mathcal{V}_\pi(b^{a,o}) \right]. \tag{2.6}$$

In general, the value function $\mathcal{V}$ can be approximated arbitrarily closely by a piecewise-linear and convex function [32]:

$$\mathcal{V}(b) = \max_{\alpha \in \Gamma} (b(s)\alpha(s)), \tag{2.7}$$

where Γ is a finite set of hyper-planes called α-vectors. Since for each $\alpha \in \Gamma$, $\alpha(s)$ is just the expected total reward associated with the starting state s by choosing some specific action and acting according to the policy defined by Γ afterwards (more details in [14]), we know that each α-vector is associated with the chosen action. The best α-vector in the set Γ at any belief state is one that maximizes the value function. Then the policy execution is just to select the action such that the α-vector associated with it at the current belief state b is the best. Thus we see that a policy can be represented by a set of α-vectors, and computing a policy involves the construction of the α-vectors set Γ, and is performed offline in most cases.

Most of the exact POMDP approaches [31, 28, 19, 8, 3, 36] look for efficient strategies to prune dominated α-vectors, i.e. vectors that do not maximize Equation 2.7 at any belief state, to avoid maintaining α-vectors sets that are too large.

However, these algorithms are limited to solving small problems due to the extremely high complexity of exact approaches. Recent work on POMDP planning has focused on trying to find efficient approximation algorithms that can solve value function $\mathcal{V}$ with desired accuracy.

2.4 Point-based Algorithms

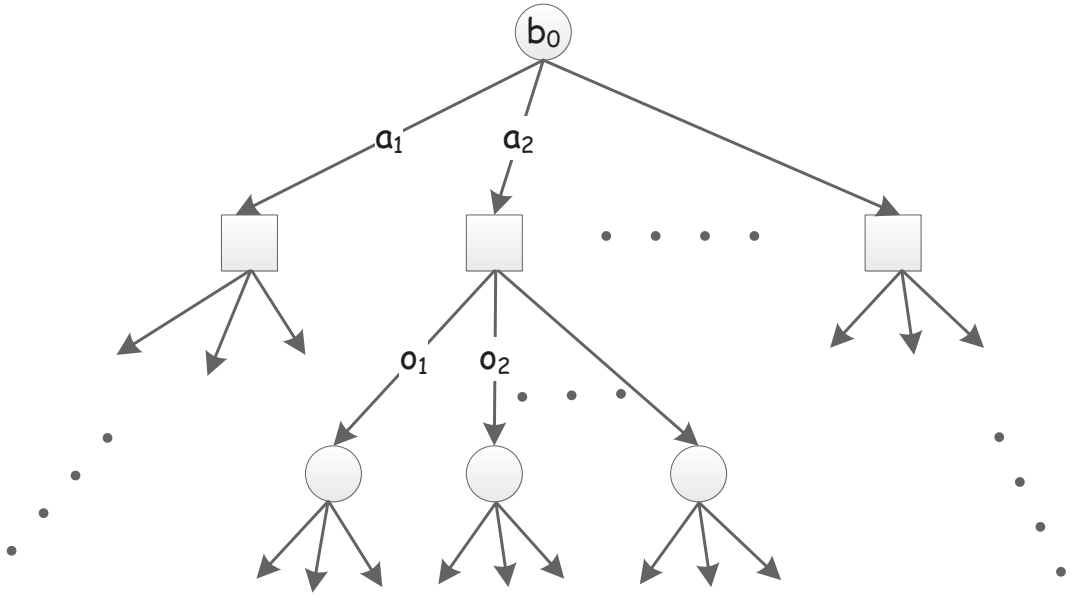
We have seen that the goal of POMDP planning is to find an optimal policy. However, computing policies for large POMDPs is not an easy task due to two famous reasons [14, 23]. One is known as "curse of dimensionality" in the sense that we must reason in a $(n-1)$-dimensional continuous belief space for a problem with n states. The other one is known as "curse of history" in the sense that the number of belief states computed from distinct action observation histories considered grows exponentially with the planning horizon in a complete search starting from an initial belief.

Point-based approximations are among the most successful approaches to approximate the value function in large POMDPs. Solutions are computed by applying iterative value function backups over a small set of belief points (see Algorithm 1). In this way, a set of points (belief states) from the belief space $\mathcal{B}$ can be sampled as an approximate representation of $\mathcal{B}$, rather than representing the exact belief space $\mathcal{B}$. To reduce computation, the choice of the candidate belief points at which to ap-

Algorithm 1 α-backup

$\alpha_{a,o} \leftarrow \text{argmax}_{\alpha \in \Gamma} \sum_{s \in \mathcal{S}} \alpha(s) b^{a,o}(s)$, $\forall a \in \mathcal{A}, o \in \Omega$;
$\alpha_a(s) \leftarrow R(s,a) + \gamma \sum_{o,s'} T(s,a,s') O(s',a,o) \alpha_{a,o}(s')$, $\forall a \in \mathcal{A}, s \in \mathcal{S}$;
$\alpha' \leftarrow \text{argmax}_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} \alpha'(s) b(s)$;
$\Gamma \leftarrow \Gamma \cup \{\alpha'\}$.

Figure 2–1: A belief tree rooted at b_0.



ply backups becomes crucial for this class of algorithms, since the backup operations are expensive. Different point-based approaches vary mainly in their belief point sampling strategies. PBVI [23], one of the first point-based algorithm, samples only belief points in a reachable space $R(b_0) \subseteq \mathcal{B}$ from an initial belief point b_0 as the representative set for backup operations rather than sampling from the high dimensional continuous $\mathcal{B}$. Many later point-based algorithms [29, 30, 33, 27, 16] are based on this idea. The sampled points typically form a tree $T_{\mathcal{R}}$ (Figure 2–1) rooted at the initial belief point b_0. Each node of $T_{\mathcal{R}}$ represents a sampled point.

Practically speaking, solving real-world tasks in the framework of POMDP involves solving two problems: how to carry out intelligent information gathering in a large high dimensional continuous belief space; and how to scale up planning with

long sequences of actions and delayed rewards. Recent point-based planning algorithms have made impressive improvements by tending to address one or the other of the above challenges, but we still lack methods that can simultaneously tackle problems of both types:

- PBVI [23], HSVI [29, 30], Perseus [33], FSVI [27] and SARSOP [16] work very successfully in problems that require gathering information with a large state space, using bias expansion of belief sampling to yield good approximations of value functions. In particular, HSVI keeps lower and upper bounds on the value function and use heuristics to sample belief states that reduce the gap between bounds. Perseus adapts a randomized belief sampling strategy and works well for highly explorative problems. FSVI uses only upper bound to guide the belief point sampling, and works well for domains that require only simple information gathering actions. SARSOP focuses sampling around the optimal reachable space, $R^*(b_0) \subseteq R(b_0)$ under optimal policies, and works well for various domains and can moderately scale to problems with up to $100,000$ states in reasonable computational time. However, when working in domains that require planning with long sequences of actions, these approaches often struggle because they fail to sample deeper belief states before having constructed belief trees that are too large to handle.
- MiGS [15] facilitates planning for problems that require long sequences of actions to reach a goal, by generating macro-actions and restricting the policy space. It has been shown to perform well in domains that require long planning

horizons with delayed reward, but we demonstrate how it can fail even in some very simple tasks that require certain information gathering efforts.

CHAPTER 3
Information Gathering and Rewards Exploitation

Now we introduce our approach, called *Information Gathering and Reward Exploitation of Subgoals* (IGRES), which attempts to bridge the gap between the two classes of point-based approaches, to produce a single POMDP solver with sufficient versatility to address both problems that require substantial information gathering actions, and problems that require long sequences of actions. To achieve this, IGRES identifies potentially important states as subgoals, and leverages MiGS's [15] insight of exploiting state structure to generate macro-actions for transitions to subgoals; Then for each belief associated with a subgoal, IGRES generates another macro-action to gather information and exploit rewards in the neighbourhood of those subgoals, which avoids sampling unnecessary beliefs. Thus, IGRES is efficient in terms of computational time and space in the sense that it covers the belief space well with a much smaller size of belief points set for expensive backup operations while still maintaining good performance.

In this chapter, we first present our strategy for sampling belief points, and give the overview of IGRES. Then we demonstrate the data structure used in the algorithm, and present the theoretical results on bounding the approximation error. Next we discuss the complexity reduced by the algorithm as well as the completeness of planning. Finally we provide a discussion on work related to our approach.

3.1 Sampling New Belief States

In this section, we present how IGRES uses the structure of state information to generate two types of macro-actions for sampling new belief states.

3.1.1 Capturing Important States

In general, actions that yield high rewards or gather significant information from the current belief state play an important role in constructing good policies [5, 11]. This observation suggests that states associated with high rewards or informative observations may also be important. IGRES leverages this structure by attempting to identify these potentially important states and bias the subgoal sampling towards them. Specifically, a state $s \in \mathcal{S}$ is sampled as subgoal using the softmax function:

$$p(s) \propto e^{\eta h(s)}, \tag{3.1}$$

where the pre-defined positive constant η serves as a normalizer and a controller for the smoothness of our random sampling, and $h(s)$ is a measure that indicates how important the state s is. The importance function $h(s)$ is defined as

$$h(s) = \frac{h_r(s)}{\sum h_r(s)} + \lambda \frac{h_i(s)}{\sum h_i(s)}, \tag{3.2}$$

where λ balances between the normalizing values defined by two heuristic functions $h_r(s)$ and $h_i(s)$ which describes the importance of a state $s \in \mathcal{S}$ in terms of the capacities of reward exploitation and information gathering respectively.

We calculate the importance of reward exploitation for each $s \in \mathcal{S}$:

$$h_r(s) = \max_{a \in \mathcal{A}} \frac{R(s,a) - R_{min}}{R_{max} - R_{min}}, \tag{3.3}$$

which captures the highest immediate reward we can get from state s over all actions, and is normalized by using minimum and maximum instantaneous rewards, which makes sure that $h_r(s)$ is strictly positive.

We then calculate the information gain for each $s \in \mathcal{S}$:

$$h_i(s) = \max_{a \in \mathcal{A}} \sum_{o \in \Omega} \left(-\frac{1}{|\Omega|} \log(\frac{1}{|\Omega|}) + O(s,a,o) \log\left(O(s,a,o)\right) \right). \tag{3.4}$$

which measures for state s the highest possible entropy of observation probabilities over all actions against a uniform distribution.

3.1.2 Leveraging State Structure

In this section, we follow MiGS [15] algorithm to exploit structure in the state space, which helps sampling in the belief space. We first consider the state graph $G_{\mathcal{S}}$ which is a weighted, directed multi-graph with each node corresponding to a state $s \in \mathcal{S}$, where edge $(\overline{ss'}, a)$ exists from node s to s' if and only if $T(s,a,s') > 0$. We associate with each edge $(\overline{ss'}, a)$ a weight that measures the difference between the expected total reward of state s and destination s' via action a:

$$\alpha_w(s) - \alpha_w(s') = R(s,a) + \gamma \sum_{s'' \in \mathcal{S}/\{s'\}} T(s,a,s') \times \sum_{o \in \Omega} O(s'',a,o)(\alpha_w(s'') - \alpha_w(s')), \tag{3.5}$$

where $\alpha_a(s) = R(s,a) + \gamma \sum_{o,s'} T(s,a,s') O(s',a,o) \alpha_w(s')$ approximates the expected total reward from state $s \in \mathcal{S}$ with the inspiration from Algorithm 1. Since solving all equations for all unknown $\alpha_w(s)$ is expensive if $|\mathcal{S}|$ is large, we need another approximation. If we consider the fact that s'' can be reached from s with only one single action, we can expect the difference between $\alpha_w(s'')$ and $\alpha_w(s)$ is small. Thus

we get the following equation by replacing $\alpha_w(s'')$ with $\alpha_w(s)$:

$$\alpha_w(s) - \alpha_w(s') = \frac{R(s,a)}{1-\gamma+\gamma T(s,a,s')}, \tag{3.6}$$

If we only consider costs (negative rewards) as an analogue of distance, we define the weight for an edge $(\overline{ss'}, a)$ as:

$$w(\overline{ss'}, a) = \frac{-R(s,a)\cdot \mathbb{1}_{R(s,a)\leq 0}}{1-\gamma+\gamma T(s,a,s')}, \tag{3.7}$$

such that the weights for all edges in $G_{\mathcal{S}}$ are non-negative and can be used to construct a distance measure. This definition of edge weights properly captures the cost of selecting action a transitioning from s to s' and takes both immediate cost and future regret into account.

Now we define the distance as

$$d_{\mathcal{S}}(s,s') = \min_{a\in\mathcal{A}:T(s,a,s')>0} w(\overline{ss'}, a). \tag{3.8}$$

We further extend the notion of distance $d_{\mathcal{S}}$ such that distance from s to s' is just the shortest path by the distance measure above. Thus we have reduced the directed multigraph $G_{\mathcal{S}}$ to a weighted directed graph.

If we assume strong connectivity of the state graph, an inward Voronoi partitioning [9] can then be used to partition the states based on $d_{\mathcal{S}}$ into a partitioning

$$\mathcal{K} = \{K(m) | m \in \mathcal{M}\}, \tag{3.9}$$

where $\mathcal{M} \subseteq \mathcal{S}$ is a set of subgoals and $K(m) = \{s \in \mathcal{S} | d_{\mathcal{S}}(s, m) < d_{\mathcal{S}}(s, m'), \forall m' \neq m \text{ and } m, m' \in \mathcal{M}\}$ is the set of states whose distance to subgoal m is less than the distance to any other subgoal in $\mathcal{M}$.

We build a roadmap graph $G_{\mathcal{M}}$ where each node corresponds to a sampled subgoal. Edge $\overline{mm'}$ from subgoal m to m' is present if a path is found from m to m' in graph $G_{K(m) \cup K_{m'}}$, and the edge is labeled with both a sequence of actions and a sequence of states according to the path. Then edge $\overline{mm'}$ is also associated with the weight of the shortest path. In this way, we partition the state space into regions where states in each region lead towards a subgoal inducing that region, and the connectivity between regions is also well structured by the edges of $G_{\mathcal{M}}$.

3.1.3 Sampling Belief States Using Macro-actions

A macro-action is constructed by a sequence of actions. IGRES adapts the notion of sampling new belief points using macro-actions instead of single actions as well as samples one observation sequence for each macro-action, which facilitates planning with long sequences of actions and avoids unnecessary backup operations.

Now we describe the details of IGRES's belief sampling strategy using macro-actions. Just like most of the point-based algorithms, IGRES only performs backup operations at a set R of belief points sampled from $\mathcal{B}$ rather than the entire belief space $\mathcal{B}$. To sample a new belief point, we choose an existing belief point $b \in R$ and suppose that an estimate of current state s is also given (how to obtain the estimate of current state will be discussed in the later sections). A new belief state b' is then sampled from the current belief state b according to a macro-action $(a_1, a_2, \ldots, a_l)$ and a state sequence $(s_0, s_1, s_2, \ldots, s_l)$ where $s_0 = s$.

To achieve this, we first generate an observation sequence $(o_1, o_2, \ldots, o_l)$ where o_i is sampled according to the probability

$$Pr(o_i) \propto O(s_i, a_i, o_i) = Pr(o_i|s_i, a_i). \tag{3.10}$$

In this way, the observation sequence $(o_1, o_2, \ldots, o_l)$ being generated is consistent with the state sequence $(s_0, s_1, s_2, \ldots, s_l)$ in the sense that it is indeed possible to receive observation o_i when transitioning from state s_{i-1} to s_i via action a_i, for $1 \leq i \leq l$. This consistency limits the number of possible observations. The motivation for sampling observation sequences is to avoid exploring many similar paths in belief space since many sequences provide similar information, which helps to reduce the complexity of long horizon due to observations as well.

After the observation sequence is sampled, by applying the updating rule of belief states (2.3), we immediately get a sequence of belief states $(b_1, b_2, \ldots, b_l)$ such that

$$\begin{aligned} b_1 &= b^{a_1,o_1}, \\ b_i &= b_{i-1}^{a_{i-1},o_{i-1}}, \text{ for } 2 \leq i \leq l. \end{aligned} \tag{3.11}$$

Thus the new belief point $b' = b_l$ is sampled, associated with the estimate state $s' = s_l$.

As the core of our algorithm, two types of macro-actions:

1. subgoal-oriented macro-actions,
2. exploitation macro-actions,

are generated, which splits the planning strategy into subgoal transitioning phase and exploitation phase respectively.

Generating Subgoal-Oriented Macro-actions

With the help of roadmap $G_{\mathcal{M}}$, we first extract a path transitioning to the closest $m \in \mathcal{M}$ from current state s. Recall that each path in $G_{\mathcal{M}}$ is an action sequence associated with the edges along the path. To choose a path, we store a circular list of all outgoing edges with increasing weights for each subgoal $m \in \mathcal{M}$ according to $G_{\mathcal{M}}$. If the current state s is not a subgoal, the path will be the only path from s to the subgoal in the same partition; otherwise s is a subgoal and the path is just the next outgoing edge in the list. From this path, we obtain a sequence of actions as our subgoal oriented macro-action and a sequence of states that are visited along the path. According to the belief sampling strategy (Eqn 3.10, 3.11), we get a new belief state b' associated with m. Since the subgoal $m \in \mathcal{M}$ is potentially important, we then introduce an exploitation macro-action next, which is restricted to the neighbourhood of m.

Generating Exploitation Macro-actions

Given an estimate of current state s at each iteration, we sample an action a with probability $Pr(a)$ proportional to some heuristic function $\tilde{a}(s)$. Many heuristics can be adopted here, such as choosing the action that maximizes long term reward, or just uniformly at random. We define $\tilde{a}(s) = e^{\mu(T(s,a,s')\mathbf{1}_{s,s' \in K(m)})}$ in order to favor actions that are limited in the partition where the subgoal is located. For simplicity and computational efficiency, we use $\tilde{a}(s) = e^{\mu T(s,a,s)}$ in our algorithm to favor the action that could gather information explicitly without changing the estimate of state, which works well empirically. And we sample a state s' to be the updated estimate of state with probability $Pr(s') \propto T(s, a, s')$. We stop exploiting with probability

$(1 - p_{ex})$ at each iteration, where p_{ex} is an exploitation probability parameter to control the length of this macro-action. At the end of the exploitation, we add an action that maximizes the reward from the current estimate of state if necessary, which is consistent with the heuristic function (Eqn 3.3) for choosing subgoals. Given the sequence of actions and the sequence of states generated in this way, we again get a new belief state according to the belief sampling strategy (Eqn 3.10, 3.11).

3.2 Overview of IGRES

IGRES maintains a belief tree rooted at the initial belief node u_{b_0} which consists of all belief states that are sampled from $\mathcal{B}$. It also stores all belief states that are associated with subgoal states in a set $\mathcal{R}_\delta$ to judge whether to exploit a subgoal or not. In IGRES, adding new belief nodes to $T_\mathcal{R}$ is a three-steps procedure:

1. Choose an existing node u_b in $T_\mathcal{R}$.
2. Generate a subgoal-oriented macro-action for b and obtain new belief b'. Then $u_{b'}$ is inserted into $T_\mathcal{R}$ as a child of u_b.
3. Generate an exploitation macro-action for b' and obtain new belief b'', if the algorithm decides to exploit. Then $u_{b''}$ is inserted into $T_\mathcal{R}$ as a child of $u_{b'}$.

Then, as a point based algorithm, IGRES performs backup operations (Algorithm 2) for all the belief nodes after each of them is inserted in the tree.

Algorithm 2 Backup(Γ,b)

$\alpha_{a,o} \leftarrow \text{argmax}_{\alpha \in \Gamma} \sum_{s \in \mathcal{S}} \alpha(s) b^{a,o}(s)$, $\forall a \in \mathcal{A}, o \in \Omega$;
$\alpha_a(s) \leftarrow R(s,a) + \gamma \sum_{o,s'} T(s,a,s') O(s',a,o) \alpha_{a,o}(s')$, $\forall a \in \mathcal{A}, s \in \mathcal{S}$;
$\alpha' \leftarrow \text{argmax}_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} \alpha'(s) b(s)$;
$\Gamma \leftarrow \Gamma \cup \{\alpha'\}$.

The full description of IGRES is given in Algorithm 3.

Algorithm 3 IGRES

Input: $b_0, G_{\mathcal{M}}, \delta$
Output: $\mathcal{V}(b_0)$
$T_{\mathcal{R}} \leftarrow \{u_{b_0}\}, R_\delta \leftarrow \{b_0\}$;
while number of rounds $\mathcal{V}(b_0)$ not improving $<$ limit **do**
Sample node $u_b \in T_{\mathcal{R}}$ probability $\propto \frac{1}{|K_c(b)|}$;
if $b = b_0$ **then**
Sample $u_b.state$ according to $b(s)$;
end if
Current estimate of state $s \leftarrow u_b.state$;
Generate subgoal oriented macro-action for b with state estimate s, which gives updated belief b' and s';
if $\min_{\hat{b} \in \mathcal{R}_\delta} d_{\mathcal{K}}(b', \hat{b}) > \delta$ **then**
BackUpAtLeaf(b',s',$T_{\mathcal{R}}$,u_b); ▷ Algorithm 4.
$R_\delta \leftarrow R_\delta \cup \{b'\}$;
Generate exploitation macro-action for b' with state estimate s', which gives updated belief b'' and s'';
BackupAtLeaf(b'',s'',$T_{\mathcal{R}}$,$u_{b'}$); ▷ Algorithm 4.
end if
end while

3.3 Data structure of IGRES

At each iteration, the algorithm chooses a belief node u_b in the tree $T_{\mathcal{R}}$ with probability proportional to $\frac{1}{|K_c(b)|}$ to start the belief expansion, where the c-neighbourhood $K_c(b)$ is defined as

$$K_c(b) = \{\hat{b} \in \mathcal{R} | d_{\mathcal{K}}(b, \hat{b}) \leq c\}, \tag{3.12}$$

Algorithm 4 BackupAtLeaf(b,s,$T_{\mathcal{R}}$,u^{parent})

Create a node u_b;
$u_b.state \leftarrow s$;
Insert u_b to $T_{\mathcal{R}}$ as a child of node u^{parent};
Backup for each belief inducing the node along the path from u_b back to root u_{b_0}; ▷ Algorithm 2.

to favor belief states that have fewer neighbours. A similar belief expansion strategy is adopted by PBVI [23], except that instead of using L_1 distance, we use a more relaxed $\mathcal{K}$-distance $d_{\mathcal{K}}$, defined based on $\mathcal{K}$-partitioning (Eqn 3.9):

$$d_{\mathcal{K}}(b, b') = \sum_{m \in \mathcal{M}} \left| \sum_{s \in K(m)} b(s) - \sum_{s \in K(m)} b'(s) \right|, \tag{3.13}$$

which measures the differences between probability sums for partitions rather than probabilities for single states. Besides the associated belief state b, we also store the estimate of current state s in each node u_b.

Then, a new belief state b' is sampled by generating a subgoal-oriented macro-action from b, and the estimate of current state s' is also updated to be the subgoal state (see Section 3.1.3).

To control the total number of belief states inserted in the tree, we calculate the $\mathcal{K}$-distance from the new belief state b' to the belief set $\mathcal{R}_\delta$ to decide whether we will exploit this subgoal or not. If the $\mathcal{K}$-distances from b' to all belief states in $\mathcal{R}_\delta$ are less than δ, we decide to do exploitation. To exploit the belief b' with state estimate s', a new belief state b'' is sampled by generating an exploitation macro-action from b' (see Section 3.1.3).

Once a new belief node is inserted in the tree, an α-backup is done for every belief state along the path from that node back to the root. The corresponding estimate of current state is also stored in each belief node to facilitate sampling deeper belief points: if we choose to sample new beliefs from that node later, we can start immediately from the estimate of current state rather than sampling a new state as we do for the root. This strategy enables reasoning about long sequences

of actions without sampling unnecessary belief states, and thus avoids maintaining a belief tree that is too large.

As the algorithm describes, IGRES considers using different macro-actions for transitioning to different subgoals in a circular way. So after a sufficient number of iterations, several belief states associated with the same subgoal will be sampled. This might degrade the performance of the algorithm because those belief states may be sampled using the same subgoal-oriented macro-action which provides similar observation information even though a different observation sequence is sampled each time. To overcome this, IGRES is implemented in an any-time manner such that after the value of initial belief state is not improved for some number of rounds, new states are sampled as subgoals and added to $\mathcal{M}$. In this way, a new $G_{\mathcal{M}}$ is constructed accordingly, which introduces new subgoals for gathering information and exploiting rewards.

3.4 Analysis

In this section, we present the theoretical results on bounding the approximation error. Then we discuss the complexity reduced by the algorithm as well as the completeness of planning.

3.4.1 $\mathcal{K}$-Distance Threshold

The $\mathcal{K}$-distance threshold δ is used to control the number of belief states associated with subgoals for backup operations. It is desirable that the algorithm does not lose too much performance because of δ, since intuitively two belief states within a $\mathcal{K}$-distance of δ have similar values. Next we demonstrate this idea by showing how the difference of the values of these two belief states can be bounded.

The partitioning $\mathcal{K}$ makes sure that all underlying state sequences starting from $s \in K(m)$ must travel to m before entering other $K(m')$, where $m' \neq m$. If we assume sampling a proper number of subgoals and a proper η (see 3.1), all states with positive reward will then be included in $\mathcal{M}$ with large probability. Under this assumption and assuming from Equation 3.7 that α_w approximate the value α properly, we know

$$\alpha(s) - \alpha(s') \leq \alpha(m) - \alpha(s) \leq \frac{\max_{a \in \mathcal{A}} -R(s,a)}{1-\gamma} l_{max} \leq \frac{-R_{min}}{1-\gamma} l_{max}. \tag{3.14}$$

where $s, s' \in K(m)$, and $l_{max} = \max_{m \in \mathcal{M}} |K(m)|$, which yields the following:

Theorem 1. *If $d_{\mathcal{K}}(b, b') \leq \delta$, then $|\mathcal{V}^*(b) - \mathcal{V}^*(b')| \leq \frac{1}{1-\gamma}(\delta R_{max} - 2 l_{max} R_{min}), \forall b, b' \in \mathcal{B}$.*

Proof. Recall from Equation 2.7 that the optimal value function can be approximated arbitrarily closely by $\mathcal{V}^*(b) = \max_{\alpha \in \Gamma}(b(s)\alpha(s))$. Let α be the maximizing α-vector for belief b and α' for b'. Assume w.l.o.g. that $\mathcal{V}^*(b) \geq \mathcal{V}^*(b')$, or $\mathcal{V}^*(b) - \mathcal{V}^*(b') \geq 0$. Since α' maximizes $\mathcal{V}^*$ at b', we get $\alpha' \cdot b' \geq \alpha \cdot b'$ and $\mathcal{V}^*(b) - \mathcal{V}^*(b') = \alpha \cdot b - \alpha' \cdot b' \leq \alpha \cdot b - \alpha \cdot b' \leq \alpha \cdot (b - b')$. Thus we have

$$|\mathcal{V}^*(b) - \mathcal{V}^*(b')| \leq |\alpha \cdot (b - b')|.$$

We next introduce the $\mathcal{K}$-partitioning in our analysis:

$$\begin{aligned} |\mathcal{V}^*(b) - \mathcal{V}^*(b')| &\leq \left| \sum_{s \in \mathcal{S}} \alpha(s)(b(s) - b'(s)) \right| \\ &\leq \left| \sum_{K \in \mathcal{K}} \sum_{s \in K} \alpha(s)(b(s) - b'(s)) \right|. \end{aligned}$$

For any $s, s' \in K$, we have

$$
\begin{aligned}
|\mathcal{V}^*(b) - \mathcal{V}^*(b')| &\leq \left| \sum_{K \in \mathcal{K}} \sum_{s \in K} (\alpha(s) - \alpha(s') + \alpha(s'))(b(s) - b'(s)) \right| \\
&\leq \left| \sum_{K \in \mathcal{K}} \sum_{s \in K} (\alpha(s) - \alpha(s'))(b(s) - b'(s)) \right| + \left| \sum_{K \in \mathcal{K}} \sum_{s \in K} \alpha(s')(b(s) - b'(s)) \right|.
\end{aligned} \tag{3.15}
$$

Let e_1 and e_2 denote the two error terms on the right hand side in above inequality (3.15). We know from the inequality (3.14) that

$$
\begin{aligned}
e_1 &\leq \left| \sum_{K \in \mathcal{K}} \sum_{s \in K} \frac{-l_{max} R_{min}}{1 - \gamma} (b(s) - b'(s)) \right| \\
&= \frac{-l_{max} R_{min}}{1 - \gamma} \sum_{s \in \mathcal{S}} |b(s) - b'(s)| \\
&\leq 2 \frac{-l_{max} R_{min}}{1 - \gamma}.
\end{aligned} \tag{3.16}
$$

Since the values of $\alpha(s) \leq \frac{R_{max}}{1-\gamma}$ and $\sum_{K \in \mathcal{K}} \sum_{s \in K} |b(s) - b'(s)| = d_{\mathcal{K}}(b, b') \leq \delta$ it follows that

$$
\begin{aligned}
e_2 &\leq \sum_{K \in \mathcal{K}} |\alpha(s')| \left| \sum_{s \in K} (b(s) - b'(s)) \right| \\
&\leq \sum_{K \in \mathcal{K}} \frac{R_{max}}{1 - \gamma} \left| \sum_{s \in K} (b(s) - b'(s)) \right| \\
&\leq \frac{R_{max}}{1 - \gamma} \delta.
\end{aligned} \tag{3.17}
$$

Combining (3.16) and (3.17) yields the theorem. □

This theorem implies that $\mathcal{V}^*(b')$ can be approximated by $\mathcal{V}^*(b)$ with a small error controlled by the $\mathcal{K}$-distance threshold δ and an extra error that depends on the number of states in each partition.

3.4.2 Reduced Complexity

Since the backup operations are often the most expensive steps in a point-based algorithm, the choice of nodes inserted in the belief tree (see Figure 2–1) becomes crucial for reducing the planning complexity. IGRES reduces the planning complexity for the following two reasons:

1. Belief points are sampled using macro-actions instead of single actions. So the belief tree grows much deeper with the same number of nodes.
2. Only one observation sequence is sampled each time for each macro-action instead of generating all possible observation sequences exhaustively.

If we consider a POMDP problem that requires planning with horizon length h, an exhaustive approach yields a belief tree with a total number of $\Theta((|\mathcal{A}||\Omega|)^h)$ belief nodes.

IGRES significantly alleviates the effect of long planning horizon by adapting macro-actions. Let l be the minimum length of a macro-action generated by IGRES, then the size of belief tree constructed is reduced to $\mathcal{O}((d|\Omega|^l)^{\frac{h}{l}}) = \mathcal{O}(d^{\frac{h}{l}}|\Omega|^h)$ for the same planning length h, where the maximum branching number d is the maximum among the out degrees of all subgoals in $G_{\mathcal{M}}$.

IGRES also carries out effective information gathering in the sense that it does not exhaustively generate all possible observation sequences for one macro-action because many of these sequences give similar information about the hidden current states. This strategy further reduces the complexity introduced by branching out exhaustively different observations.

3.4.3 Completeness of Planning

IGRES approximates the reachable space R_{b_o} from the initial belief b_0 instead of sampling the whole space $\mathcal{B}$ exhaustively in the sense that:

1. IGRES samples states as subgoals, and generates new belief states with macro-actions. It only gathers information and exploits rewards at belief states associated with subgoal states, and excludes other belief states generated along the way transitioning to subgoals for exploitation.
2. IGRES uses the $\mathcal{K}$-distance threshold $\delta = 0$ to avoid reasoning belief states that are too close to the existing belief point set.

However, IGRES is still planning in a complete policy space under some conditions:

In the case where the distance threshold $\delta = 0$, IGRES inserts every belief state that has been sampled. Then we consider the fact that $G_{\mathcal{M}}$ is updated by adding without replacement new subgoals sampled from a finite set of states $\mathcal{S}$. Thus, even if we start with a small number of subgoals, all states would be sampled as subgoals given enough running time for IGRES. As new edges are inserted to $G_{\mathcal{M}}$, including multiple edges labeled with different actions between same pair of subgoals, all possible edges between each pair of states will finally be added to $G_{\mathcal{M}}$ (subgoal-oriented macro-actions become single actions in this case). If we also consider the fact that for a non-zero exploitation probability parameter $p_e x$, self-loop action sequences of any length could be generated with non-zero probability. In this way, any transitioning action between any pair of states with non-zero probability or any possible self-loop action of any state could be taken to sample new belief. Since observation is

sampled according to observation function O, with non-zero probability all possible action-observation sequences could be generated, which covers the whole reachable space $R(b_0)$. Local exploitation in principle ensures completeness over the reachable space.

Therefore IGRES is complete in a probability sense. Although practically speaking, this might not be necessary for most of POMDP problems.

3.5 Related Work and Discussion

Planning with macro-actions in fully-observable environments was first introduced in the semi-Markov Decision Processes and the options framework [34]. It was then adapted to partially-observable domains with a dynamic grid approximation approach [35].

More recently, many algorithms that are based on a generic online forward search for POMDPs [25] (e.g. PUMA [11]) provide good performance for domains that require long planning horizons by restricting the policy space with macro-actions. However, they do not compute full policies for the problems.

MiGS [15] adapts a similar approach as that of IGRES by identifying important states as subgoals, and automatically constructing macro-actions to tackle POMDP tasks with long planning horizons. However, MiGS does poorly in gathering information, and thus struggles in generic POMDPs including domains like Tiger [6] and RockSample [29] which require sensing action sequences. We will demonstrate this in the next chapter.

The key idea of point-based POMDP algorithms is to sample a set of points from $\mathcal{B}$ to approximate the representation of entire $\mathcal{B}$ because the backup operations

for the sampled points are expensive. Thus, the strategy of sampling belief points is crucial for a point-based algorithm. IGRES performs point-based value function backup operations and leverages the structure of state space and adapts macro-actions for sampling belief points to reduce the complexity of planning, which helps to scale better than other point-based approaches for general POMDP problems that require effective information gathering in large state space and long action sequences of planning.

In IGRES, the calculation of heuristic values for all states and the construction of $G_{\mathcal{M}}$ are done only once before the process of belief sampling starts, which makes the generation of two types of macro-actions simple and effective. So in this sense, IGRES is efficient in sampling useful belief points.

The generation of macro-actions in IGRES is relatively simple and random, where the choices of actions are not conditional on the observation received or current belief states. This strategy ensures IGRES performs well for general POMDP domains. However, we expect that different domain heuristics applied to the construction of macro-actions will lead to better performance for specific POMDP domains.

CHAPTER 4
Experiments

In this chapter, we present the testing results of IGRES on different tasks that require either information gathering efforts or long planning horizons.

We first consider classic POMDP benchmark problems of various sizes and types. We compare performance of IGRES to existing POMDP solvers RTDP-Bel [2], HSVI2 [30], FSVI [27], SARSOP [16] and MiGS [15]. RTDP-Bel is an adaptation of real time dynamic programming to POMDPs, and has demonstrate strong performance on some POMDP domains. FSVI is a trial-based algorithm that finds sequences of useful backups, and performs well on most of POMDPs. HSVI2 uses heuristics to guide the sampling, and has good performance in general. SARSOP is a state-of-the-art POMDP solver with demonstrated strong performance in a variety of domains, especially problems with large state space. MiGS is a state-of-the-art solver specifically aimed at domains with long planning horizons and delayed rewards.

We then present the results of applying IGRES to a real-world ecological adaptive management domain, and demonstrate its behavior with different parameters.

4.1 Benchmark Problems

POMDP as a model for planning differs largely from other fully observable models due to the uncertainty of state information. Thus, algorithms that sample belief states by using pure state information to generate macro-actions (e.g. MiGS) might

not work well on some problems, largely because they fail to take advantages of observation information to sample beliefs which are more focused to a single important state.

On the other hand, for problems that require long action sequences, algorithms that adapt certain heuristics to gather information (e.g. SARSOP) might improve very slowly when they maintain a large set of sampled beliefs or they might even get stuck locally on some problems because they fail to sample deeper belief points.

IGRES constructs macro-actions considering both informative observations and rewarding actions, and gathers information and exploits subgoals in a probabilistic manner without making assumptions about the domain. Thus, IGRES has the potential to provide good solutions to problems which require either information gathering to compute good policies or long sequences of actions to exploit delayed rewards. To demonstrate this, we carried out experiments on both types of benchmark problems.

4.1.1 Introduction to Domains Studied

First, we introduce the POMDP domains on which we ran IGRES and the other five algorithms as comparison.

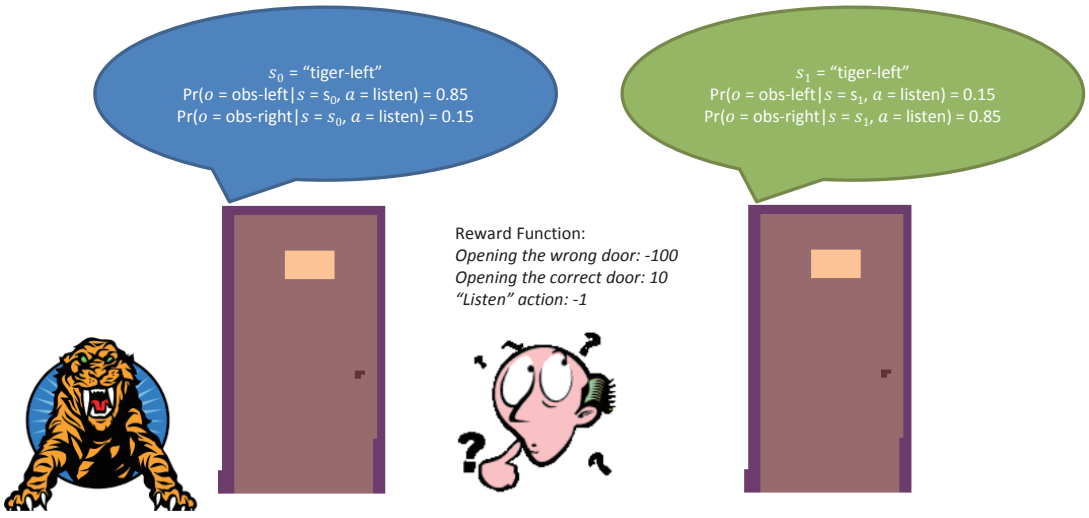
Tiger and Noisy Tiger

Tiger [6] (see Figure 4–1) is the most classic task to demonstrate the information gathering perspective of a POMDP problem. In this problem, two closed doors are given. A positive reward is behind one of the doors, while a huge cost is behind the other door which represents a tiger. The agent can use small-cost "listen" action before opening the right or left door. The "listen" action gives observations indicating the correct location of the tiger with 15% noise.

Noisy-tiger [11] is a modified version of Tiger where the noise of the *listen* action is increased from 15% to 35% in order to make the information gathering more difficult, thus yielding a longer planning horizon.

Typically successful policies for these two problems will sample belief states that are close to the "corner" of state representing the correct door, which requires several "listen" efforts.

Figure 4–1: Tiger Problem.



RockSample

RockSample [29] is another information gathering problem, which models a rover exploring a grid map and trying to sample valuable rocks. In RockSample(n, k), the grid map of size $n \times n$ as well as k rocks are given. The rover knows the exact positions of itself and the k rocks. But it does not have the prior knowledge of whether a rock is valuable or not. The rover moves deterministically, and can perform k types

of "check" actions to obtain noisy information about whether the corresponding rock is good. The accuracy of each "check" action depends on the distance to the corresponding rocks. The rover gets +10 reward if it correctly samples a good rock and is punished with -10 if the sampling is not correct.

Hallway2

Hallway2 is a maze domain introduced in [18], where a robot navigates in a map with 22 grid rooms plus a goal and it wants to reach the designated goal as quickly as possible. The robot observes each combination of the presence of a wall in four directions and a "star" for the goal. The robot can move towards four directions or stay in the same place for each time step. The transitions and observations are both extremely noisy.

Tag

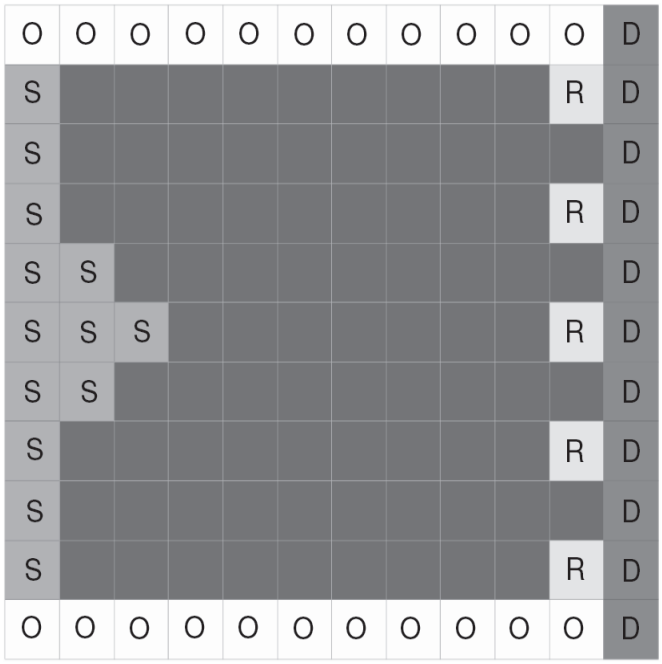
Tag is a domain introduced in [23] where a robot searches for and tags a moving opponent. The position of the robot is fully observable, and the "move" actions are deterministic. However, the robot cannot observe the opponent until they meet at the same position, when it should take the "tag" action to get a reward of 10 and win the game. The robot gets -1 for each "move" action, and -10 for a wrong "tag" action.

Underwater Navigation

Underwater Navigation [16] (Figure 4–2) models an autonomous underwater vehicle (AUV) navigating in a static 51×52 grid map from the left border to the right. The AUV knows the environment. At each time step, it chooses to stay or move to the adjacent grids. The difficulty of this domain is due to its large state

space and the relatively long planning horizons. A successful solution will guide the AUV to the top or bottom borders and localize itself around the landmarks there in order to avoid dangerous rocks near goals.

Figure 4–2: Underwater Navigation Problem. The figure shows a reduced instance of Underwater Navigation Problem, shown on a reduced map with a 11×12 grid for coastal navigation. "S" marks the possible starting positions with uniform probabilities. "D" marks the destinations. "R" marks the rocks. "O" marks the landmarks where the robot can fully localize itself. (This figure is reproduced from [16] with the permission from the authors.)

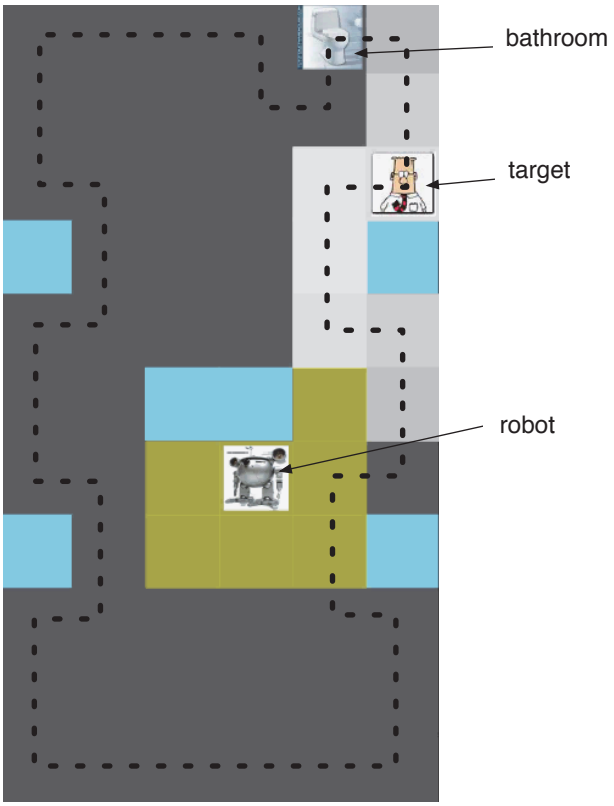


Homecare

Homecare [16] (Figure 4–3) is a domain closely related to Tag [23] but with much more states and more complex dynamics. It models a robot that tries to take care

of an elderly person who moves on a certain path non-deterministically. The person will proceed or stay along the dash path with uniform probabilities each time. And he may stay for a long duration when he reaches the bathroom. To achieve high reward, the robot needs to keep close to the person so that it could sense the person. The robot gets a reward only if it serves in time when the person calls for help.

Figure 4–3: Homecare Problem. This figure shows the robot tracking a moving person. The light blue areas are obstacles, and the black dashed curve indicates the targets path. The green area around the robot marks the the visible region of the sensors of the robot. The gray shades show the belief states of the current target position. (This figure is reproduced from [16] with the permission from the authors.)

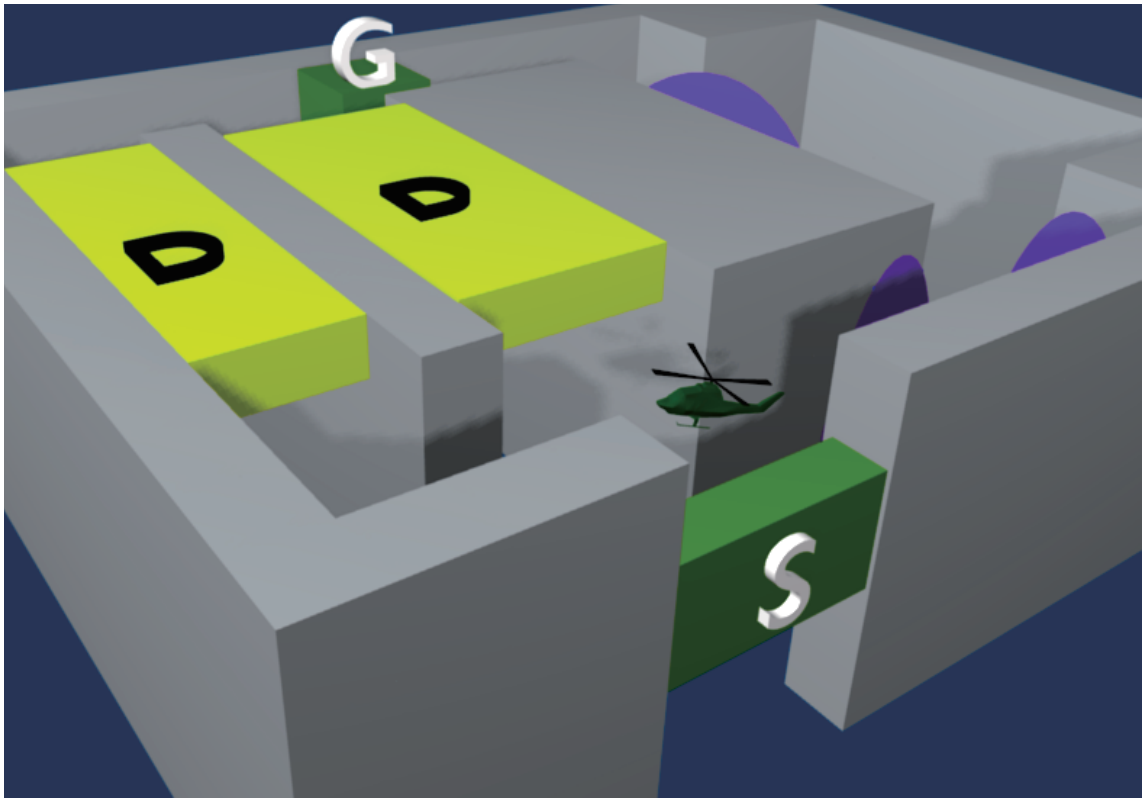


3D-Navigation

3D-Navigation [15] (Figure 4–4) models an unmanned aerial vehicle (UAV) navigating in a 3D-indoor environment with 18×14 horizontal grids and 5 height levels. The UAV moves with 10% noise but rotates accurately. A successful solution will

lead the UAV to the longer route on the right hand side where the vehicle can localize itself at specific landmarks along the way, and avoid dangerous areas before heading for the goal areas.

Figure 4–4: 3D-Navigation Problem. This figure shows a 3D-indoor environment for navigation. "S" denotes possible starting positions, "G" denotes the goals, "D" denotes danger areas, and the three dark regions are landmarks for localizing the vehicle. (This figure is reproduced from [15] with the permission from the authors.)



4.1.2 Experiments Setup

We compared performance of IGRES to existing POMDP solvers RTDP-Bel [2], HSVI2 [30], FSVI [27], SARSOP [16] and MiGS [15] on all benchmark domains introduced above[1] .

We performed these experiments on a computer with a 2.50GHz Intel Core i5-2450M processor and 6GB of memory. We ran MiGS and IGRES 30 times each to compute policies on Homecare and 3D-Navigation, and 100 times each for the other domains. Then we ran 100 simulations for each policy computed. We ran each of the four other algorithms once for each domain until convergence or achieving good level of estimated values. Then we ran sufficient number of simulations to evaluate each policy computed by these algorithms. The average reward with the 95% confidence intervals and the corresponding computation times are reported.

4.1.3 Results

Table 4–1 reports the average reward returned and the computation time used by each of the algorithms.

For the first two domains, we observe that while RTDP-Bel, HSVI2, SARSOP and IGRES show good performance, MiGS fails on both Tiger and Noisy-tiger even

[1] For RTDP-Bel, we use the software package provided by the original authors of paper [27]. For HSVI2, we use the latest ZMDP version 1.1.7 (`http://longhorizon.org/trey/zmdp/`). For RTDP-Bel, we use the software package provided by the original authors of paper [2]. For SARSOP, we use the latest APPL version 0.95 (`http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.Download`). For MiGS, we use the software package provided by the original authors of paper [15].

Table 4–1: Results of benchmark problems.

	Return	Time(s)
Tiger		
$\lvert\mathcal{S}\rvert = 2, \lvert\mathcal{A}\rvert = 3, \lvert\Omega\rvert = 2$		
RTDP-Bel	$\mathbf{19.42 \pm 0.59}$	0.30
HSVI2	$\mathbf{19.31 \pm 0.09}$	<1
FSVI*	N/A	
SARSOP	$\mathbf{18.59 \pm 0.61}$	0.09
MiGS	-19.88 ± 0	100
IGRES (# subgoals: 1)	$\mathbf{19.41 \pm 0.59}$	1
Noisy-tiger		
$\lvert\mathcal{S}\rvert = 2, \lvert\mathcal{A}\rvert = 3, \lvert\Omega\rvert = 2$		
RTDP-Bel	$\mathbf{-13.67 \pm 0.28}$	1.22
HSVI2	$\mathbf{-13.69 \pm 0.04}$	<1
FSVI*	N/A	
SARSOP	$\mathbf{-13.66 \pm 0.18}$	0.18
MiGS	-19.88 ± 0	100
IGRES (# subgoals: 1)	$\mathbf{-13.67 \pm 0.18}$	1
RockSample(4,4)		
$\lvert\mathcal{S}\rvert = 257, \lvert\mathcal{A}\rvert = 9, \lvert\Omega\rvert = 2$		
RTDP-Bel	$\mathbf{17.94 \pm 0.12}$	10.7
HSVI2	$\mathbf{17.92 \pm 0.01}$	<1
FSVI	$\mathbf{17.85 \pm 0.18}$	1
SARSOP	$\mathbf{17.75 \pm 0.12}$	0.7
MiGS	8.57 ± 0	100
IGRES (# subgoals: 4)	17.30 ± 0.12	10
RockSample(7,8)		
$\lvert\mathcal{S}\rvert = 12545, \lvert\mathcal{A}\rvert = 13, \lvert\Omega\rvert = 2$		
RTDP-Bel	20.55 ± 0.13	103
HSVI2	$\mathbf{21.09 \pm 0.10}$	100
FSVI	20.08 ± 0.20	102
SARSOP	$\mathbf{21.35 \pm 0.13}$	100
MiGS	7.35 ± 0	100
IGRES (# subgoals: 8)	19.54 ± 0.12	100
Hallway2		
$\lvert\mathcal{S}\rvert = 92, \lvert\mathcal{A}\rvert = 5, \lvert\Omega\rvert = 17$		
RTDP-Bel	0.237 ± 0.006	1004
HSVI2	0.507 ± 0.001	250
FSVI	0.494 ± 0.007	280
SARSOP	$\mathbf{0.530 \pm 0.008}$	200
MiGs	0.522 ± 0.008	200
IGRES (# subgoals: 20)	$\mathbf{0.530 \pm 0.008}$	200

Tag		
$\lvert\mathcal{S}\rvert = 870, \lvert\mathcal{A}\rvert = 5, \lvert\Omega\rvert = 30$		
RTDP-Bel	-6.32 ± 0.12	372
HSVI2	-6.46 ± 0.09	400
FSVI	$\mathbf{-6.11 \pm 0.11}$	35
SARSOP	$\mathbf{-6.08 \pm 0.12}$	30
MiGS	$\mathbf{-6.00 \pm 0.12}$	30
IGRES (# subgoals: 20)	$\mathbf{-6.12 \pm 0.12}$	30
Underwater Navigation		
$\lvert\mathcal{S}\rvert = 2653, \lvert\mathcal{A}\rvert = 6, \lvert\Omega\rvert = 103$		
RTDP-Bel	$\mathbf{750.07 \pm 0.28}$	338
HSVI2	718.37 ± 0.60	400
FSVI	725.88 ± 5.91	414
SARSOP	731.33 ± 1.14	150
MiGS	715.50 ± 1.37	400
IGRES (# subgoals: 20)	$\mathbf{747.25 \pm 0.50}$	10
Homecare		
$\lvert\mathcal{S}\rvert = 5408, \lvert\mathcal{A}\rvert = 9, \lvert\Omega\rvert = 928$		
RTDP-Bel**	N/A	
HSVI2	15.07 ± 0.37	2000
FSVI***	N/A	
SARSOP	$\mathbf{16.64 \pm 0.82}$	1000
MiGS	$\mathbf{16.70 \pm 0.85}$	1600
IGRES (# subgoals: 30)	$\mathbf{17.32 \pm 0.85}$	1000
3D-Navigation		
$\lvert\mathcal{S}\rvert = 16969, \lvert\mathcal{A}\rvert = 5, \lvert\Omega\rvert = 14$		
RTDP-Bel	-93.03 ± 0.01	2115
HSVI2	-91.98 ± 0	2000
FSVI**	N/A	
SARSOP	-99.97 ± 0	800
MiGS	$\mathbf{(2.977 \pm 0.512) \times 10^4}$	150
IGRES (# subgoals: 163)	$\mathbf{(3.272 \pm 0.193) \times 10^4}$	150

* ArrayIndexOutOfBoundsException is thrown.

** Solver is not able to compute a solution given large amount of computation time.

*** OutOfMemoryError is thrown when the input file is being parsed.

though the problems are extremely simple. The reason is that MiGS fails to produce policies that take actions changing the probability distribution within state dimensions. MiGS performs poorly on the RockSample problem for similar reasons as it does on the Tiger domain. IGRES easily solves the first four problems because it successfully samples important states as subgoals and takes helpful actions around the subgoals to gather information and exploit reward.

For the Hallway2 problem, SARSOP and IGRES achieve best solutions due to their strong ability to gather useful information.

For the Tag problem, RTDP-Bel and HSVI2 are not able to compute a good solution, while the other four algorithms achieve high rewards in a short computation time.

For Underwater Navigation task, while all solvers achieve a good solution, I-GRES is substantially faster than the other approaches.

For the Homecare problem, IGRES achieves higher rewards than the other solvers given similar computation time.

For 3D-Navigation task, RTDP-Bel, HSVI2 and SARSOP are unable to achieve a good solution because they fail to sample belief points that are far away from the existing set, and thus get trapped in the local area, whereas MiGS and IGRES easily overcome the effect of long planning horizon in this task.

We conclude from these results that IGRES is able to successfully tackle both problems requiring information gathering, and problems with long planning horizons, unlike previous solvers which specialize in one or the other of these classes of problems.

4.2 The Ecological Adaptive Management Problem

In this section, we apply IGRES to a class of ecological adaptive management tasks [20] that was presented as an IJCAI 2013 data challenge problem to the POMDP community. To the best of our knowledge, there has not been any published work on this challenge so far.

The POMDPs in this domain represent networks for migratory routes by different shorebird species utilizing the East Asian-Australasian flyway, under uncertainty of the rate of sea level rise and its effect on shorebird populations. The goal is to select one node to perform protection action against a fixed amount of sea level rise in a weighted directed graph representing the flyway. The state space is a factored representation by the cross product of: one fully observable population variable, the fully observable protection variable for each node in the graph, and one variable for sea level rise which is not observable. Five different bird species are considered (characterized by different POMDP sizes and parameterizations).

4.2.1 Results for All Species of Birds

To solve this problem, we ran IGRES 30 times for each task to compute policies, and then ran 100 simulations to test each computed policy. Our results are generated on a 2.67GHz Intel Xeon W3520 computer with 8GB of memory. We also present results of the benchmark solutions computed by the original authors of the dataset using symbolic Perseus [24] on a more powerful computer [20].

We observe in Table 4–2 that IGRES outperforms the benchmark solutions by achieving higher rewards for all species. Even though IGRES is not specially designed for solving the problem and does not directly exploit the factored state structure,

Table 4–2: Results of Adaptive Management of Migratory Birds.

	Return	Time(s)
Lesser sand plover		
$\lvert\mathcal{S}\rvert = 108, \lvert\mathcal{A}\rvert = 3, \lvert\Omega\rvert = 36$		
symbolic Perseus*	4675	10
IGRES (# subgoals: 18)	5037.72 ± 8.82	10
Bar-tailed godwit b.		
$\lvert\mathcal{S}\rvert = 972, \lvert\mathcal{A}\rvert = 5, \lvert\Omega\rvert = 324$		
symbolic Perseus*	18217	48
IGRES (# subgoals: 36)	19572.41 ± 39.35	60
Terek sandpiper		
$\lvert\mathcal{S}\rvert = 2916, \lvert\mathcal{A}\rvert = 6, \lvert\Omega\rvert = 972$		
symbolic Perseus*	7263	48
IGRES (# subgoals: 72)	7867.95 ± 2.44	60
Bar-tailed godwit m.		
$\lvert\mathcal{S}\rvert = 2916, \lvert\mathcal{A}\rvert = 6, \lvert\Omega\rvert = 972$		
symbolic Perseus*	24583	58
IGRES (# subgoals: 72)	26654.06 ± 38.60	60
Grey-tailed tattler		
$\lvert\mathcal{S}\rvert = 2916, \lvert\mathcal{A}\rvert = 6, \lvert\Omega\rvert = 972$		
symbolic Perseus*	4520	378
IGRES (# subgoals: 72)	4860.91 ± 38.47	60
IGRES (# subgoals: 72)	4927.17 ± 38.14	300

* Results from [20].

it computes good solutions that yield high rewards in reasonable computation times for all these tasks.

4.2.2 Behaviors with Different Configurations

We further experiment with IGRES on one of the tasks, varying parameters to demonstrate its behaviour as the number of subgoals changes. In order to save experimental time, we only computed 20 policies for each combination of parameters.

Figure 4–5: Performance of IGRES on Grey-tailed Tattler.

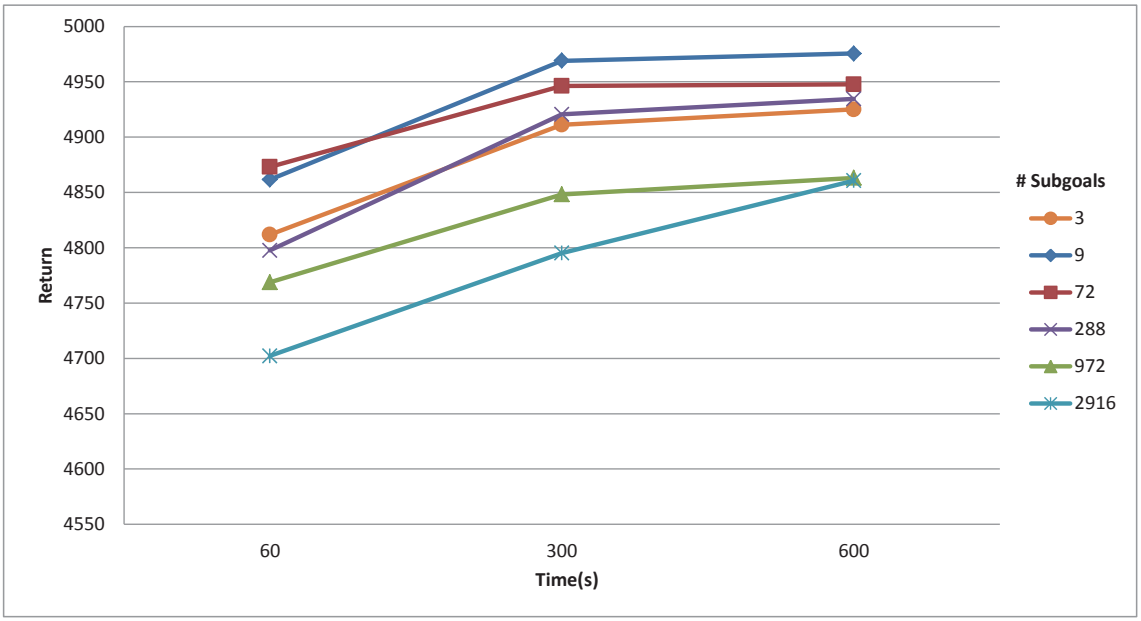


Figure 4–5 shows the behaviour of IGRES with different number of subgoals used given increasing computation time on Grey-tailed Tattler task. Given reasonable number of subgoals, IGRES is able to take advantage of subgoals that help sampling useful belief points. However, if the number of subgoals gets too large, a much larger set of beliefs would be sampled before the algorithm plans deep enough to compute a good solution, thus degrading the performance. On the other hand,

if the number of subgoals is too small, some important states might be omitted from the set of subgoals, which also degrades the performance. Recall that the baseline performance for symbolic Perseus (as shown in Table 4–5) with 378 seconds of planning yielded a return of just 4520, well under the results shown in Figure 4–5. Thus, the performance indicates that IGRES is robust against changes in the number of subgoals for this type of problems and consistently samples important subgoal states to compute good solutions.

CHAPTER 5
Conclusion

In this chapter, we conclude with a discussion of our approach as well as suggestions for the possible work in the future.

5.1 Discussion

Practically speaking, the challenges of effective information gathering in the large belief space and planning with long sequences arise in many planning tasks modeled with POMDP framework. The capacities of tackling the two objectives are important for applying POMDP planning algorithms in real-world settings.

Recent point-based algorithms have made impressive progress in effectively solving problems that require information gathering in large state spaces or aiming at problems with long planning horizon. However, combining these two complementary qualities into one single approach is advantageous but not easy.

In this thesis, we present a new multi-purpose POMDP planning algorithm, IGRES, which leverages state information to identify subgoals that are essential for computing good policies, and automatically generates macro-actions to improve computational efficiency while maintaining good performance. We provide some theoretical properties of the algorithm on how it can effectively tackle the challenges offered by real-world tasks. And to demonstrate and justify the theoretical results, we carry out experiments on a variety of domains in comparison with other approaches.

Despite the simplicity of macro-actions construction, IGRES computes better solution at up to 10 times faster speed for some problems, and successfully generalizes to a wider set of tasks than previous state-of-the-art POMDP solvers. We also present improved empirical performance for a set of real-world challenge tasks in ecological management, with significant potential impact. These promising results suggest that the notion of gathering information for exploitation of subgoals with macro-actions provides a new perspective to view POMDP problems, which advances application of POMDP planning for complex tasks in practice.

5.2 Future Work

Although IGRES provides promising experimental results that justify its theoretical properties, it still lacks strong theoretical guarantees on error bounds or convergence rate, which is often the case for the family of point-based approaches. Also, the hierarchical structure of the belief tree can be further explored to limit the branching number and provide bounds on the convergence of the algorithm. Any of such theoretical results will lead to better understandings of the complexity of approximate POMDP planning, and inspire further advances of applying POMDP framework in practice.

To scale POMDP planning from another perspective, many recent algorithms for various factored POMDPs have been proposed to explore the inner structure of the POMDP models, which provide promising results of scaling large POMDP problems. For example MOMDPs [21], which are factored POMDPs with mixed observation, assume part of the states are perfectly observable, and utilize this structure to speed

up the planning algorithm. It would be interesting to apply the notion of planning with macro-actions to different types of factored POMDPs.

IGRES adapts a naive strategy in generating exploitation macro-actions, which empirically has been shown to work well in general. This is partly due to the lack of knowledge about each domain. In fact, as is mentioned previously in the thesis, many domain heuristics can be adopted in constructing the macro-actions for effectively reasoning about subgoal states. We would like to see how different heuristics work out.

There are other issues and limitation about the experiments. This thesis only compares IGRES with five other algorithms on a limited number of domains for the purpose of demonstrating how the experimental results support the motivation and theoretical claims of effectively tackling the two practical challenges. However, it would be interesting to include other point-based algorithms or other types of approaches for comparison. It would also be interesting to see how IGRES performs on various other POMDP domains.

References

[1] K. J. Aström. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174C–205, 1965.

[2] B. Bonet and H. Geffner. Solving POMDPs: RTDP-bel vs. point-based algorithms. In *International Jont Conference on Artifical Intelligence*, pages 1641–1646, 2009.

[3] A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Uncertainty in Artificial Intelligence*, 1997.

[4] A. R. Cassandra. A survey of POMDP applications. In *AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, 1998.

[5] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *International Conference on Intelligent Robots and Systems*, 1996.

[6] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *National Conference on Artificial Intelligence*, 1994.

[7] I. Chades, J. Carwardine, T. G. Martin, S. Nicol, R. Sabbadin, and O. Buffet. MOMDPs: A solution for modelling adaptive management problems. In *National Conference on Artificial Intelligence*, 2012.

[8] H. Cheng. *Algorithms for partially observable Markov decision processes*. PhD thesis, University of British Columbia, University of British Columbia, 1988.

[9] M. Erwig. The graph Voronoi diagram with applications. *Networks*, 36(3):156–163, 2000.

[10] M. Hauskrecht and H. S. F. Fraser. Planning treatment of ischemic heart disease with partially observable Markov decision processes. *Artificial Intelligence in Medicine*, 18(3):221–244, 2000.

[11] R. He, E. Brunskill, and N. Roy. Puma: Planning under uncertainty with macro-actions. In *National Conference on Artificial Intelligence*, 2010.

[12] J. Hoey, P. Poupart, A. von Bertoldi, T. Craig, C. Boutilier, and A. Mihailidis. Automated handwashing assistance for persons with dementia using video and a partially observable Markov decision process. *Computer Vision and Image Understanding*, 114(5):503–519, 2010.

[13] K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez. Grasping POMDPs. In *IEEE International Conference on Robotics and Automation*, 2007.

[14] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[15] H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee. Motion planning under uncertainty for robotic tasks with long time horizons. *International Journal of Robotics Research*, 30(3):308–323, 2011.

[16] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[17] M. L. Littman. *Algorithms for sequential decision-making*. PhD thesis, Brown University, 1996.

[18] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning*, 1995.

[19] G. E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[20] S. Nicol, O. Buffet, T. Iwamura, and I. Chadès. Adaptive management of migratory birds under sea level rise. In *International Joint Conference on Artificial Intelligence*, 2013.

[21] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee. POMDPs for robotic tasks with mixed observability. In *Robotics: Science and Systems*, 2009.

[22] J. Pineau and A. Atrash. Smartwheeler: A robotic wheelchair test-bed for investigating new models of human-robot interaction. In *AAAI Spring Symposium: Multidisciplinary Collaboration for Socially Assistive Robotics*, 2007.

[23] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, 2003.

[24] P. Poupart. *Exploiting structure to efficiently solve large scale partially observable Markov decision processes.* PhD thesis, University of Toronto, 2005.

[25] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32(1):663–704, July 2008.

[26] N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Association for Computational Linguistics*, 2000.

[27] G. Shani, R. I. Brafman, and S. E. Shimony. Forward search value iteration for POMDPs. In *International Joint Conference on Artifical Intelligence*, 2007.

[28] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[29] T. Smith and R. G. Simmons. Heuristic search value iteration for POMDPs. In *Uncertainty in Artificial Intelligence*, 2004.

[30] T. Smith and R. G. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *Uncertainty in Artificial Intelligence*, 2005.

[31] E. J. Sondik. *The optimal control of partially observable Markov processes.* PhD thesis, Stanford University, 1971.

[32] E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26:282–304, 1978.

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

[34] R. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[35] G. Theocharous and L. P. Kaelbling. Approximate planning in POMDPs with macro-actions. In *Advances in Neural Information Processing Systems*, 2004.

[36] N. L. Zhang and W. Zhang. Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51, 2001.