Information Gathering and Reward Exploitation of Subgoals for POMDPs

Hang Ma and Joelle Pineau
School of Computer Science
McGill University, Montreal, Canada

Abstract
Planning in partially observable Markov decision processes (POMDPs) is challenging especially when a long planning horizon is required. A few recent algorithms successfully tackle this case but at the expense of a weaker information-gathering capacity. In this paper, we propose 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" to 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. Additional experiments with an ecological adaptive management problem indicate that IGRES is a promising tool for POMDP planning in real-world settings.

Introduction
Partially observable Markov decision processes (POMDPs) have emerged as a rich framework for planning under uncertainty due to their ability 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 has been widely applied to various complex situations in practice thanks to its rich capabilities for planning in an uncertain environment (Cassandra 1998; Hsiao, Kaelbling, and Lozano-Pérez 2007; Pineau and Atrash 2007; Chades et al. 2012). Despite their mathematical expressivity, the difficulty of solving large POMDPs has limited their application in complex domains. For a POMDP problem modelled with \( n \) states, we must reason in an \((n - 1)\)-dimensional continuous belief space. Moreover, the complexity of POMDP planning is affected by the length of planning horizon (Littman 1996). 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; and how to scale up planning with long sequences of actions and delayed rewards.

Recent point-based planning algorithms have made impressive improvements by addressing one or the other of the above challenges: PBVI (Pineau, Gordon, and Thrun 2003), HSVI2 (Smith and Simmons 2004; 2005), FSVI (Shani, Brafman, and Shimony 2007) and SARSOP (Kurniawati, Hsu, and Lee 2008) 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; MiGS (Kurniawati et al., 2011) and some online solvers, e.g. PUMA (He, Brunskill, and Roy 2010), facilitate planning for problems that require long sequences of actions to reach a goal, by generating macro-actions and restricting the policy space. Other approaches, including a dynamic programming adaption to POMDPs called RTDP-Bel (Bonet and Geffner 2009) and a UCT-based online method called POMCP (Silver and Veness 2010) have demonstrated strong performance on some of the POMDP domains. However, we still lack methods that can tackle problems simultaneously requiring substantial information gathering and long planning horizons.

Here we propose a point-based 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 approaches: first, IGRES identifies potentially important states as subgoals; second, IGRES only gathers information and exploits rewards with macro-actions in the neighbourhood of those subgoals. 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 provides state-of-the-art performance on tasks that require significant information gathering and planning with long sequences of actions. We also show that IGRES can effectively tackle a new class of ecological adaptive management problems presented in the IJCAI 2013 data challenge track, thus providing evidence that IGRES is capable of solving tasks in useful real-world settings.

Technical Background
A partially observable Markov decision process (POMDP) is a tuple \( \langle \mathcal{S}, \mathcal{A}, \Omega, T, O, R, b_0, \gamma \rangle \) (Kaelbling, Littman, and Cassandra 1998), where \( \mathcal{S} \) is a finite set of states, \( \mathcal{A} \) is the set of actions, \( \Omega \) is the set of observations, \( T \) is the transition function, \( O \) is the observation function, \( R \) is the reward function, \( b_0 \) is the initial belief, and \( \gamma \) is the discount factor.
a finite set of actions, $\Omega$ is a finite set of observations, $T(s,a,s') = p(s'|s,a)$ is the transition function that maps each state and action to a probability distribution over states, $O(s',a,o) = p(o|s',a)$ is the observation function that maps a state and an action to a probability distribution over possible observations, $R(s,a)$ is the reward function, $b_0(s)$ is the initial belief state, and $\gamma$ is the discount factor.

A belief state $b \in B$ is a sufficient statistic for the history, and is updated after taking action $a$ and receiving observation $o$ as follows:

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

(1)

where $p(a,b) = \sum_{s \in S} b(s) \sum_{s' \in S} T(s,a,s') O(s',a,o)$ is a normalizing factor.

A policy is a mapping from the current belief state to an action. A value function $V_n(b)$ specifies the expected reward gained starting from $b$ followed by policy $\pi$:

$$V_n(b) = \sum_{s \in S} b(s) R(s,\pi(b)) + \gamma \sum_{a \in \Omega} p(a|b,\pi(b)) V_n(b^{\pi(b),a}).$$

The goal of POMDP planning is to find an optimal policy that maximizes its value function denoted by $V^*$. In general, $V$ can be approximated arbitrarily closely by a piecewise-linear and convex function (Sondik 1978): $V(b) = \max_{a \in \mathcal{A}} (b(s)\alpha(s))$, represented by a finite set of hyper-planes, $\Gamma$.

### Point-based Solvers
Information gathering in large state spaces and planning with long sequences of actions are two major challenges for planning in large POMDPs. 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). To reduce computation, the choice of the candidate belief points at which to apply backups becomes crucial for this class of algorithms. PBVI (Pineau, Gordon, and Thrun 2003), one of the first point-based algorithms, samples only belief points in a reachable space $R(b_0)$ from an initial belief point $b_0$ as the representative set for backup operations rather than sampling from the high dimensional continuous $B$. HSVI2 keeps lower and upper bounds on the value function and use heuristics to sample belief states that reduce the gap between bounds. FSVI (Shani, Brafman, and Shimony 2007) uses only upper bound to guide the belief point sampling, and works well for domains that require only simple information gathering actions. SARSOP (Kurniawati, Hsu, and Lee 2008) focuses sampling around the optimal reachable space, $R'(b_0) \subseteq R(b_0)$, sampled from optimal sequences of actions. MiGS (Kurniawati et al. 2011) adopts the notion of generating macro-actions, which are sequences of actions, rather than single actions to expand the belief point set, which is one of the successful strategies to tackle problems with long planning horizons and delayed rewards.

### Algorithm 1 Backup($\Gamma,b$)

$\alpha_{a,o} \leftarrow \arg \max_{\alpha \in \mathcal{A}} \sum_{s \in S} \alpha(s) \gamma \alpha(o,s) \forall a \in \mathcal{A}, o \in \Omega$;

$\alpha_a(s) \leftarrow R(s,a) + \gamma \sum_{s',o} T(s,a,s') O(s',a,o) \alpha_{a,o}(s')$, $\forall a \in \mathcal{A}, s \in S$;

$\alpha' \leftarrow \arg \max_{\alpha' \in \mathcal{A}} \sum_{s \in S} \alpha'(s)b(s)$;

$\Gamma \leftarrow \Gamma \cup \{\alpha'\}$

### Gathering Information & Exploiting Rewards

Our work 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 tries to identify potentially important states as subgoals using a strategy similar to MiGS (Kurniawati et al. 2011). Then for each belief associated with a subgoal, IGRES generates another macro-action to gather information and exploit rewards.

### 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 (Cassandra, Kaelbling, and Kurien 1996; He, Brunskill, and Roy 2010). This suggests that states associated with high rewards or informative observations may also be important. Our algorithm leverages this structure by attempting to identify these potentially important states and use them as subgoals.

Specifically, we define two heuristic functions $h_r(s)$ and $h_i(s)$ of a state $s \in S$, to describe its importance in terms of rewards exploitation and information gathering respectively. We calculate the importance of rewards exploitation for some state $s$:

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

(2)

which captures the highest immediate reward we can get from state $s$ over all actions. We also calculate its information gain as:

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

(3)

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

Then a state $s \in S$ is sampled as subgoal using the softmax function:

$$p(s) \propto e^{\eta \left( \frac{h_r(s)}{\lambda} + \frac{h_i(s)}{\tau} \right)},$$

(4)

where the pre-defined positive constant $\eta$ serves as a normalizer and a controller for the smoothness of our random sampling, and $\lambda$ balances between the two heuristic functions.
Leveraging State Structure

In this section, we follow the algorithm of MiGS to exploit structure in the state space, that helps focus sampling in the belief space. We first consider the state graph $G_S$. We associate with each edge $(s, s')$ a weight that measures the difference between the expected total reward of state $s$ and destination $s'$ via action $a$: $w(s, s') = R(s, a) + \gamma \sum_{s'' \in S(s')} T(s, a, s'') \times \sum_{o \in O} O(s'', a) \alpha(s'') - \alpha(s'))$, where $\alpha(s) = R(s, a) + \gamma \sum_{o \in O} O(s', a) \alpha(s')$ approximates the expected total reward. Since $s'$ can be reached from $s$ with one single action, we expect the difference between $\alpha(s'')$ and $\alpha(s)$ is small. By replacing $\alpha(s'')$ with $\alpha(s)$, we get:

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

(5)

which captures the cost of selecting action $a$ transitioning from $s$ to $s'$. Then we define the distance as $d_S(s, s') = \min_{a \in A} \{T(s, a, s') > 0 \} |w(s, s')|$. We further extend the notion of distance $d_S$ such that distance from $s$ to $s'$ is just the shortest path by the distance measure above. Now we have reduced the directed multigraph $G_S$ to a weighted directed graph. If we assume strong connectivity of the state graph, we know the edge regions is also well structured by the edges of subgoal inducing that region, and the connectivity between subgoals is also well structured by the edges of $G_S$. We build a roadmap graph $G_M$ where each node corresponds to a sampled subgoal. Edge $mm'$ from subgoal $m$ to $m'$ is present if there exists a path from $m$ to $m'$ in graph $G_{K_m \cup K_{m'}}$. And this edge is labeled with a sequence of actions and a sequence of states according to the path. Then we know the edge $mm'$ is also associated with the weight of the shortest path. In this way, we partition the state space into regions where the states in each region lead towards a subgoal inducing that region, and the connectivity between regions is also well structured by the edges of $G_M$.

Sampling Belief States Using Macro-actions

Now we give a description of the belief sampling strategy. Specifically, given an estimate of current state $s$, a new belief state $b'$ is 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 with probability

$$p(o_i) \propto O(s_i, a_i, o_i) = p(o_i | s_i, a_i).$$

(7)

Using the updating rule of belief states (1), we immediately get a sequence of belief states $(b_0, b_1, b_2, \ldots, b_l)$ such that

$$b_1 = b_0^{a_0 \cdot a_1}; \quad b_i = b_{i-1}^{a_{i-1} \cdot a_i},$$

(8)

where $b_0 = b$ and $b_1 = b'$. As the core of the algorithm, two types of macro-actions, subgoal-oriented macro-actions and exploitation macro-actions, are generated, which split our planning strategy into subgoal transitioning phase and exploitation phase respectively.

Generating Subgoal-Oriented Macro-actions

With the help of roadmap $G_M$, we first extract a path transitioning to the closest $m \in M$ from current state $s$. To do so, we store a list of all outgoing edges for each subgoal according to $G_M$. If $s$ 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 7.8), we get a new belief state $b'$ associated with $m$. Since the subgoal $m \in M$ is potentially important, next we introduce an exploitation macro-action 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 $p(a)$ proportional to some heuristic function $\hat{a}(s)$. Many heuristics can be adopted here, such as choosing the action that maximizes long term reward, or just uniformly at random. We define $\hat{a}(s) = e^{\lambda T(s, a, s')}$ to favor the action that could gather information explicitly without changing the estimate of state, which works well empirically. And we update the estimate of state as $s'$ with probability $p(s') \propto T(s, a, s')$. We stop exploiting with probability $(1 - p_{exp})$ at each iteration, where $p_{exp}$ 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 2) for choosing subgoals.

Data Structures

IGRES maintains a belief tree $T_{\mathcal{B}}$ rooted at the initial belief node $u_0$, which consists of all belief states that are sampled. It also stores all belief states that are associated with subgoals in a set $R_\delta$. Similar to MiGS, at each iteration, the algorithm chooses a belief state in the tree with probability proportional to $|K_m(b)|$, where the $c$-neighbourhood $K_c(b)$ is defined as: $K_c(b) = \{ b \in \mathcal{R} | d_K(b, b') \leq c \}$, to favor belief states that have fewer neighbours. A similar belief expansion strategy is adopted by PBVI, except that instead of $L_1$ distance, the algorithm uses the relaxed $K$-distance $d_K$ defined based on $K$-partitioning (Eqn 6):

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

(9)

which measures the differences of probability for partitions rather than single states. We also store the estimate of current state in each node. Then a new belief state is generated by the subgoal oriented macro-action and the estimate of current state is updated to be the subgoal state. To control the total number of belief states inserted in the tree, we calculate the $K$-distance from the new belief state to the belief set $R_\delta$.

3322
to decide whether we will exploit this subgoal or not. If we
decide to exploit, we continue to do exploitation. The new
belief states are inserted in the tree, and an α-backup is done
for each of the belief states along the path back to the root.
The corresponding estimate of current state is also stored
in each belief node so that if we choose to sample new be-
iefs from that node later, we could start immediately from
the estimate of current state. This strategy enables sampling
deeper belief points without maintaining a belief tree that is
too large. After the value of the initial belief state is not im-
proved for some number of rounds, new states are sampled

\[ \text{Algorithm 2 IGRES} \]

Input: \( b_0, G_M, \delta \)
Output: \( V(b_0) \)

\[ T_R \leftarrow \{ u_{b_0} \}, R_\delta \leftarrow \{ b_0 \}; \]
while number of rounds \( V(b_0) \) not improving \( < \) limit do
Sample node \( u_s \in T_R \) probability \( \propto \frac{1}{|K(s)|} ; \)
if \( b = b_0 \) then
  Sample \( u_s, \text{state} \) according to \( b(s) ; \)
end if
Current estimate of state \( s \leftarrow u_s, \text{state} ; \)
Generate subgoal oriented macro-action \( b \) with state esti-
mate \( s \), which gives updated belief \( b' \) and \( s' \); if \( \min_{b \in R_s} d_{K_s}(b', b) > \delta \) then
  BackupAtLeaf(\( b', s', T_R, u_b \) ); \( \triangleright \) Algorithm 3.
  \( R_\delta \leftarrow R_\delta \cup \{ b \} ; \)
Generate exploitation macro-action \( b' \) with state esti-
mate \( s' \), which gives updated belief \( b'' \) and \( s'' \);
  BackupAtLeaf(\( b'', s'', T_R, u_b \) ); \( \triangleright \) Algorithm 3.
end if
end while

\[ \text{Algorithm 3 BackupAtLeaf}(b, s, T_R, u_{\text{parent}}) \]
Create a node \( u_b \);
\( u_b, \text{state} \leftarrow s' ; \)
Insert \( u_b \) to \( T_R \) as a child of node \( u_{\text{parent}} ; \)
Backup for each belief inducing each node along the path from
\( u_b \) back to root \( u_{b_0} ; \) \( \triangleright \) Algorithm 1.

Analysis of K-Distance Threshold The \( K \)-distance
threshold \( \delta \) is used to control the number of belief states
associated with subgoals for backup operations.

\[ \text{Theorem 1. If} \quad d_K(b, b') \leq \delta, \quad \text{then} \quad |V^*(b) - V^*(b')| \leq \frac{1}{1 - \epsilon} (|R_{\max} - 2|I_{\max} R_{\min}|), \forall b, b' \in B, \]
where \( I_{\max} \) is the maximal number of states traversed
along the path from any state to the subgoal in the same
partition, which is less than the number of states in that par-

tition. This theorem is derived from Theorem 1 in (Kurni-
awati et al. 2011) by relating the error to the size of each
partition, which implies that \( V^*(b') \) can be approximated by
\( V^*(b) \) with a small error controlled by the \( K \)-distancethresh-
old \( \delta \) and an extra error depends on the number of states in
each partition (details in tech report).

Complexity Reduced by Macro-actions Consider con-
structing a belief tree \( T_R \) for a set of belief points \( R \) sampled
from the reachable space \( R(b_0) \), then we have \( \Theta(|A||\Omega|^d) \)
beliefs for a planning horizon \( h \). Let \( l \) be the minimum
length of a macro-action generated by IGRES, then the size
of \( T_R \) is reduced to \( O(|d|\Omega|^l|^h) = O(d^l|\Omega|^h) \). The maxi-
mum branching number \( d \) is the maximum degree of all sub-
goals in \( G_M \).

Completeness of Planning In the case where \( \delta = 0 \),
IGRES inserts every sampled belief state. Then consider the
fact that \( G_M \) is updated by adding without replacement new
subgoals sampled from a finite \( S \). We can show that even-
tually, all states would be sampled as subgoals. As new edges
are inserted to \( G_M \), all possible edges between each pair of
states would finally be included in \( G_M \). For \( p_{\max} > 0 \), self-
loop action sequences of any length could also be generated.
Thus IGRES is planning in a complete policy space. Local
exploration ensures completeness over the belief space.

Experiments
We first consider classic POMDP benchmark problems of
various sizes and types, and then present results for a real-
world challenge domain.

Benchmark Problems
POMDP as a model for planning differs largely from MDP
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 because they fail to take advantages of ob-
servation information to identify subgoals and sample more
focused beliefs. On the other hand, algorithms that adapt cer-
tain heuristics to gather information (e.g. SARSSOP) might
improve very slowly when they maintain a large set of sam-
bled 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 informa-
tive observations and rewarding actions, and gathers infor-
mation and exploits subgoals in a probabilistic manner with-
out making assumptions about the domain. Thus, IGRES has
the potential to provide good solutions to problems which re-
quire 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.

Tiger (Cassandra, Kaelbling, and Littman 1994) is the
most classic POMDP problem. Noisy-tiger (He, Brunskill,
and Roy 2010) is a modified version of Tiger problem where
the noise of the listen action is increased to make the
information gathering more difficult. RockSample (Smith and
Simmons 2004) is another classic information gathering
problem. Hallway(2 introduced in (Littman, Cassandra, and
Kaelbling 1995) is a robot navigation problem in a grid map
of maze. Tag is a domain introduced in (Pineau, Gordon,
and Thrun 2003) where a robot searches for a moving op-
ponent. Underwater Navigation (Kurniawati, Hsu, and Lee
2008) models an autonomous underwater vehicle (AUV)
navigating in a grid map. Homecare (Kurniawati, Hsu, and
Lee 2008) is a domain similar to Tag but with much more
complex dynamic, which models a robot tracking an elderly
person. 3D-Navigation (Kurniawati et al. 2011) models an unmanned aerial vehicle (UAV) navigating in an 3D-indoor environment, which requires the vehicle localizing at specific landmarks to avoid dangerous areas before heading for the goal states.

We compared performance of IGRES to existing POMDP solvers RTDP-Bel (Bonet and Geffner 2009), HSVI2 (Smith and Simmons 2005), FSVI (Shani, Brafman, and Shimony 2007), SAR SOP (Kurniawati, Hsu, and Lee 2008) and MiGS (Kurniawati et al. 2011) on all benchmark domains introduced above: RTDP-Bel is an adaptation of real time dynamic programming to POMDPs, and has demonstrated strong performance on several POMDP domains; and the rest are four point-based algorithms aiming at various POMDP domains respectively which we have mentioned in the second section of this paper. We performed these experiments on a computer with a 2.50GHz Intel Core i5-2450M processor and 6GB of memory. We ran the probabilistic solvers, MiGS and IGRES, 30 times each to compute policies on Homecare and 3D-Navigation, and 100 times each for the other domains. 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 four algorithms. The average reward with the 95% confidence intervals and the corresponding computation times are reported in Table 1.

The number of subgoals for IGRES is randomly picked roughly according to the size of each domain. We will address the effect of varying the size of subgoals in the later set of experiments. For the Tigers domains, we observe that while RTDP-Bel, HSVI2, SAR SOP and IGRES show good performance, MiGS fails even though both 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 RockSample for the similar reasons. We notice that for both RockSample problems there exist small gaps between the results of IGRES and the best solutions in the same amount of time. This class of problems only require performing single-step information-gathering actions at specific locations with fully observable movement actions, which the algorithms adapting certain deterministic heuristics might be able to take advantage of. On the other hand, IGRES generates macro-actions in a probabilistic manner. So the parameters of generating macro-actions should be tuned to match this pattern in order to reach the optimal performance. For example on RockSample(4,4) with 20 subgoals, IGRES returns 17.91 ± 0.12 in 10 seconds of computation time. For the Hallway2 domain, SAR SOP 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 short computation time. For Underwater Navigation task, IGRES is substantially faster than RTDP-Bel to compute the best solution. For the Homecare problem, IGRES is able to achieve the best rewards given same amount of computation time. For 3D-Navigation task, RTDP-Bel, HSVI2 and SAR SOP are unable to achieve

| Task            | | | |
|-----------------|-------------------|-------------------|
| **Table 1:** Results of benchmark problems. |

\[ \text{Table 1: Results of benchmark problems.} \]
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.

The Ecological Adaptive Management Problem

Next, we apply IGRES to a class of ecological adaptive management problems (Nicol et al. 2013) 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 about 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).

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 (Poupart 2005) on a more powerful computer (Nicol et al. 2013).

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

We further experiment with IGRES on one of the tasks, varying parameters to demonstrate its behaviour as the number of subgoals changes. Figure 1 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 1) with 378 seconds of planning yielded a return of just 4520, well under the results shown in Figure 1. 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.

Table 2: Results of adaptive management of migratory birds.

<table>
<thead>
<tr>
<th>Bird Species</th>
<th>Return</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lesser sand plover</td>
<td>IGRES</td>
<td>4675</td>
</tr>
<tr>
<td>Bar-tailed godwit b.</td>
<td>IGRES</td>
<td>19572</td>
</tr>
<tr>
<td>Terek sandpiper</td>
<td>IGRES</td>
<td>7263</td>
</tr>
<tr>
<td>Grey-tailed tattler</td>
<td>IGRES</td>
<td>26654</td>
</tr>
<tr>
<td>Bar-tailed godwit m.</td>
<td>IGRES</td>
<td>24583</td>
</tr>
<tr>
<td>Grey-tailed tattler</td>
<td>IGRES</td>
<td>4927.17</td>
</tr>
<tr>
<td>Bar-tailed godwit m.</td>
<td>IGRES</td>
<td>4860.91</td>
</tr>
<tr>
<td>Bar-tailed godwit m.</td>
<td>IGRES</td>
<td>4520</td>
</tr>
</tbody>
</table>

* Results from (Nicol et al. 2013).

Figure 1: Performance of IGRES on Grey-tailed Tattler.

Conclusion

In this paper, we present a new multi-purpose POMDP planning algorithm, IGRES, which exploits 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. Despite the simplicity of macro-actions generation, IGRES computes better solution at up to 10 times faster speed for some problems, and successfully generalizes to a wide set of POMDP tasks. 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. Our code will be publicly released.
to help future efforts.\footnote{The software package will be available at: http://cs.mcgill.ca/%7Ehma41/IGRES/}

Acknowledgements
This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the Discovery Grants Program and the NSERC Canadian Field Robotics Network (NCFRN), as well as by the Fonds de Recherche du Quebec Nature et Technologies (FQRNT).

References
Shani, G.; Brafman, R. I.; and Shimony, S. E. 2007. Forward search value iteration for POMDPs. In International Joint Conference on Artificial Intelligence.