Article

The International Journal of Robotics Research
2015, Vol. 34(2) 158–172

DOI: 10.1177/0278364914553683
ijr.sagepub.com

Learning to soar: Resource-constrained exploration in reinforcement learning

Jen Jen Chung, Nicholas R.J. Lawrance and Salah Sukkarieh

Abstract

This paper examines temporal difference reinforcement learning with adaptive and directed exploration for resource-limited missions. The scenario considered is that of an unpowered aerial glider learning to perform energy-gaining flight trajectories in a thermal updraft. The presented algorithm, eGP-SARSA(λ), uses a Gaussian process regression model to estimate the value function in a reinforcement learning framework. The Gaussian process also provides a variance on these estimates that is used to measure the contribution of future observations to the Gaussian process value function model in terms of information gain. To avoid myopic exploration we developed a resource-weighted objective function that combines an estimate of the future information gain using an action rollout with the estimated value function to generate directed explorative action sequences. A number of modifications and computational speed-ups to the algorithm are presented along with a standard GP-SARSA(λ) implementation with ε-greedy exploration to compare the respective learning performances. The results show that under this objective function, the learning agent is able to continue exploring for better state-action trajectories when platform energy is high and follow conservative energy-gaining trajectories when platform energy is low.

Keywords

Reinforcement learning, exploration, informative planning, aerial robotics, autonomous soaring

1. Introduction

The trade-off between exploration and exploitation heavily influences the rate of learning in reinforcement learning (RL) problems and is of particular importance in cases where the cost of taking observations can limit the operation of the learning agent, for example, an unpowered aerial glider learning to perform energy-gaining flight trajectories in a wind energy field. To handle such resource-constrained circumstances, a management scheme must effectively quantify the exploration and exploitation utilities and consolidate these in an exploration strategy that can adapt the learning behavior according to available resources and also direct exploration to areas of high uncertainty. This paper draws ideas from existing methods for informative exploration in RL and the wider robotic exploration literature, and leverages these in a framework that can address the additional resource constraints of this problem.

Information-based exploration and planning under uncertainty is a richly studied area in robotics research. Active and informative path planning algorithms developed by Singh et al. (2009) and Hollinger et al. (2013) have exploited the submodularity property of information gain to select the most informative set of sensing locations, while Stachniss et al. (2005), Amigoni and Caglioti (2010) and Levine et al. (2010) also consider trade-offs between the information gathered and the cost of travel. These exploration approaches have typically been used for informative mapping, but can be incorporated into RL problems where the task is to learn a resource-gathering policy in an unknown environment.

In the RL literature, there have also been a number of proposed strategies that feature implementations of informative exploration. Dearden et al. (1998) maintained probability distributions over the value function, while Engel et al. (2003, 2005) demonstrated the use of Gaussian process (GP) modeling for value function approximation and noted that the GP covariance could provide confidence bounds on the value estimate to direct exploration. More recently, Brochu et al. (2010) outlined methods drawn from

Australian Centre for Field Robotics, School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Australia

Corresponding author:
Jen Jen Chung, Australian Centre for Field Robotics, School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, NSW, 2006, Australia.
Email:j.chung@acfr.usyd.edu.au

Bayesian experimental design to combine the GP covariance and mean to form acquisition functions; and in the authors' prior work (Chung et al., 2013) an information gain rollout was applied to produce a nonmyopic information value to direct exploration.

However, exploratory actions can be expensive and the learning problem considered in this work draws a tight coupling between what is observed and the agent's ability to continue to make new observations. Specifically, we consider the problem of a glider (an under-powered aircraft) learning to soar in a wind field. The goal of the glider agent is to learn and perform resource- (in this case, energy) gathering flight trajectories in the state-action space. In order to do so it must spend that resource to explore the space to gather information. This raises the question of how to represent *exploration* value and *exploitation* value quantitatively in the RL framework and how to consolidate these two in an adaptive and nonmyopic way.

By approximating the value function with a GP, the GP covariance can be manipulated to obtain a measure of the information value of future actions. However, this cannot be naïvely combined with the RL value function since the ability to explore is now dependent on the available resources. An adaptive weighting factor is introduced to represent the current level of available resource and this is used to scale exploration against exploitation.

The proposed resource-constrained RL algorithm was tested on a 3D six-degree-of-freedom (6DOF) soaring glider simulation with a single thermal updraft wind profile. Two information measures were compared: the first considers the information gain of all reachable state-action locations within a specified horizon (eGP-SARSA(λ)); the second considers only the information gain of those state-actions to be visited within the specified horizon given the current policy (greedy rollout). These were also compared against the performance of an ε-greedy policy, a sparse-eGP approach (Csató and Opper, 2002) using the full information value (that is, the first information measure) and a timestep-weighted policy, iGP-SARSA(λ), adapted from Chung et al. (2013). The results show a clear performance improvement of eGP-SARSA(λ) over the other simulated cases in terms of exploration and energy gain. Furthermore, each of the tested algorithms incorporated different amounts of information, and a general decline in performance, in terms of successful soaring episodes, is observed as less information is retained in the value function approximation and the action selection policy.

In the following section, we briefly describe the SARSA(λ) algorithm upon which our proposed algorithm is built. Section 3 outlines the value function approximation method using GPs. Section 4 describes the computation of the information measure using the GP covariance and includes details of the rollout method for nonmyopic exploration. The resource-constrained exploration–exploitation objective function is introduced in Section 5 and the full eGP-SARSA(λ) algorithm is given in Section 6. Sections 7 and 8 present the soaring glider simulation setup and results, respectively. Concluding statements and suggestions for future work are given in Section 9.

2. SARSA(λ) RL

SARSA(λ) is an on-policy temporal difference (TD) control method first introduced in Rummery and Niranjan (1994) under the name *Modified Connectionist Q-learning* or MCQ-L. Its current name arises from the algorithm procedure where the learning agent begins in a state s, performs an action a, observes the transition reward r and next state s', and then selects the next action a' according to its learned policy. The λ refers to the use of eligibility traces to assign discounted credit along the trajectory history as new transition rewards are observed. The SARSA(λ) state-action value function $Q(s,a)$ is computed as the discounted sum of all future rewards leading out from each state-action transition. The state-action value update equation for SARSA(λ) is

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a) \tag{1}$$

where the TD error is given by the difference between the observed transition reward r_{t+1} and the expected reward,

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \tag{2}$$

and the eligibility trace with replacing traces is updated by

$$e_t(s,a) = \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases} \quad \forall s, a \tag{3}$$

The step size, $0 < \alpha < 1$, weights how far, in the direction of the TD error, to shift the current state-action value estimate. The discount rate, $0 \leq \gamma \leq 1$, determines the contribution of future rewards to the current return value. Finally, the eligibility trace discount factor $0 \leq \lambda \leq 1$ determines how much of the current update to assign back along the state-action history. Values of λ approaching 1 mean that the proportion of the current transition reward passed back along the trajectory decays more slowly. For $\lambda = 1$ all state-actions in the history are updated by the same (full) amount, whereas if $\lambda = 0$, only the most recent state-action is updated by the full amount, which is equivalent to one-step SARSA. In this way, eligibility traces allow more efficient reward tracking along entire trajectories, which can lead to faster convergence rates as shown in Sutton and Singh (1994).

The motivation for using a SARSA(λ) framework in this investigation is its on-policy learning characteristic, which allows the trace to develop over the full state-action trajectory. Credit assignment continuity is lost whenever the trace is reset, discarding relevant information and essentially treating one continuous trajectory as two (or more) separate trajectories. In off-policy methods such as Q-learning, presented in Watkins (1989), the trace is reset whenever an exploratory action is taken. Therefore, despite the physical connection, credit received after an exploratory

action cannot be assigned back to state-actions visited before that action. Since the number of exploratory actions depends on the sampling policy, in cases where they occur frequently, the effect of the eligibility trace is diminished. In another Q-learning method, presented in Peng and Williams (1996), the trace is not reset; however, regardless of the action taken, the most recent transition reward is taken to be the maximum reward achievable by the available action set. This algorithm uses a hybrid of on- and off-policy updates and as a result converges neither to the state-action values of the current policy, Q^{π}, nor to the optimal policy, Q^*. As opposed to these off-policy methods, SARSA(λ) takes full advantage of the eligibility trace, allowing it to form over the entire trajectory of an episode without needing to reset the trace whenever an exploratory action is taken. This is most relevant in problems where state-actions along a trajectory are highly correlated, that is, they cannot easily be reached from any other state-action, and learning is expected to occur over long trajectories.

3. Value function approximation

The value function $Q(s, a)$ encapsulates the expected sum of all future rewards leading out from each state-action transition. From the backup equation shown in equation (1) we see that SARSA(λ), and in fact RL in general, relies on repeat observations of the reward at each state-action location to learn the value function. This may be achievable in discrete (tabular) state-action learning cases, however in problems with either continuous states and/or continuous actions, the learning agent will almost surely never revisit any particular state-action and thus cannot take advantage of any experience gained along its trajectory. The algorithm is unable to extrapolate the value of state-action pairs that have not yet been visited, thus even when dealing with purely discrete spaces, a problem can be computationally infeasible if its state-action space is prohibitively large.

To extend RL to problems with continuous state-action spaces, methods for approximating the value function must be employed. While any function approximation technique could be used within the RL framework to model the value function, it is desirable to choose a method that is best able to generalize the available observations to the full state-action space. The interested reader is directed to Chapter 8 of Sutton and Barto (1998) for a survey of common RL function approximation techniques.

3.1. GP modeling

GP regression is a function approximation method that produces a continuous estimate of the function mean as well as a measure of the estimation uncertainty over the function space in the form of a variance: see Rasmussen and Williams (2005). This technique has previously been employed to approximate the value function in various RL frameworks: for example, Engel et al. (2003) applied GP regression to learn the continuous state-action value function inside a standard SARSA(λ) procedure. Deisenroth et al. (2009) combined GP regression with dynamic programming and demonstrated the ability of the regression technique to estimate the state-action value at unobserved locations.

The training inputs $X_N = \{\mathbf{x}_i\}_{i=1}^N$ to the GP are the observed state-action pairs

$$\mathbf{x}_i = [\mathbf{s}_i, \mathbf{a}_i] \tag{4}$$

and the training targets $\mathbf{y}_N = \{y_i\}_{i=1}^N$ are the corresponding Q-values which are updated on the fly as the eligibility trace decays,

$$y_i = Q(\mathbf{s}_i, \mathbf{a}_i) \tag{5}$$

For a test point $\mathbf{x}_*$, covariance function k, covariance matrix $K = K(X,X)$ and additive white noise drawn from $\mathcal{N}(0, \sigma_n^2)$, the estimated mean value $\bar{Q}_*$ and covariance cov (Q_*) are

$$\begin{aligned} \bar{Q}_* &= \mathbb{E}[Q(\mathbf{x}_*)|X, \mathbf{y}, \mathbf{x}_*] \\ &= K(\mathbf{x}_*, X)\left[K + \sigma_n^2 I\right]^{-1}\mathbf{y} \end{aligned} \tag{6}$$

$$\operatorname{cov}(Q_*) = K(\mathbf{x}_*, \mathbf{x}_*) - K(\mathbf{x}_*, X)\left[K + \sigma_n^2 I\right]^{-1}K(X, \mathbf{x}_*) \tag{7}$$

The underlying assumption captured by the GP approximation is that state-action pairs close to one another in the covariance function space will have similar associated Q-values. This assumption is not limited to problems where the state and/or action spaces are continuous; it can also be applied to problems where the states and actions are discrete but the transition function implies some sense of continuity, such as in grid search problems. To some extent this continuity is captured by the eligibility trace, however, the trace is only able to assign credit to state-action locations that have been visited whereas the GP approximation is able to estimate the value of locations that are yet to be visited. For this reason, the GP approximation has particular applications to RL problems where the state-action space is continuous or where the problem has a discrete state-action space that is too large to explore exhaustively.

Prior assumptions regarding the properties of the value function surface can be incorporated via judicious design of the GP covariance function. For example, in the following experiments we make the assumption of stationarity in the value function model, which assumes that the uncertainty and mean estimate at a location is dependent only on its relative (covariance space) distance to the training inputs and their observed values. We use the stationary squared exponential covariance function to compute the GP model:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^{\mathrm{T}} M (\mathbf{x} - \mathbf{x}')\right) \tag{8}$$

where M is a diagonal matrix with positive elements equal to $\mathbf{l}^{-2}$, and $\mathbf{l} = [l_1, l_2, \ldots, l_n]$ are the length scales in each

dimension of the training input vector. The hyperparameters of the covariance function shown in equation (8) are the length scales, **l**, and the process variance, σ_f^2. The noise variance, σ_n^2, due to the additive white noise shown in equation (7) is the only other hyperparameter of the GP model. The covariance matrix, K, is the covariance function evaluated between each pair of points in the observed set, X. It is worth noting that angular dimensions in the training inputs invoke wrap-around conditions that must be catered for in the squared distance computation of the exponential in equation (8).

The hyperparameters used in the following experiments were trained offline on a set of simulation data gathered using an initial estimate of the hyperparameters. While it is possible to train these hyperparameters online, offline training was preferred since the initial stages of learning tended to generate large changes in the value function which would lead to instability.

4. Explicit exploration

Traditional methods for invoking exploration actions, such as ε-greedy sampling and the softmax method described in Chapter 2 of Sutton and Barto (1998), are susceptible to local minima and fail to actively direct exploration to areas where information of the value function is sparse. With sufficient prior knowledge of the problem, it is possible to initialize the value function to bias exploration towards areas of high expected reward. However it is often the case that the agent begins the task with little or no knowledge of the field. A uniformly high expected value applied across the value function space will promote exploration to areas that have not yet been visited since the value function will gradually be driven down in locations where the agent has repeat observations of lower than expected rewards. This method can equivalently be thought of as applying an arbitrary information gain reward at each state-action at the start of exploration whose value decreases as observations are made at those locations. As discussed in Section 3 however, in a continuous state-action space, the agent will almost surely never revisit any single location, and so this method of promoting exploration would only serve to add a constant bias over the value function approximation.

Nevertheless, the task of exploring a value function space can be equated to an information-gathering task where observations at different locations in the space reduce the uncertainty of the value estimate at, and possibly around, the observation location depending on the estimation algorithm. In such a scenario, the information gain can be measured as the uncertainty reduction. Given a GP framework for modeling the value function, it is intuitive to use the GP variance measure to quantify and compare the information gain of possible state-action observations for directing exploration. Furthermore, since the ultimate goal of accumulating reward is inherently tied to accurately modeling the value function, using the GP variance to measure the information gain maintains a desirable consistency across the value function approximation and information reward.

Continuing along this line of thought, we draw inspiration from the definition of the value function as the discounted sum of all future rewards to define the *information value* as the discounted sum of all possible future information gain rewards extending from a state-action transition. This can be estimated using the rollout method proposed in the authors' prior work (Chung et al., 2013), and is described in Section 4.2. Using an information value introduces nonmyopic exploration, and by including this measure in the action selection criteria, it is possible to direct exploration for more effective coverage of the state-action space.

4.1. Information measure

The informative sampling literature outside of the RL community is rich with suggestions of how to measure and incorporate the notion of uncertainty reduction when choosing sampling locations. The alphabet optimality criteria are derived from maximizing properties of the information matrix such as maximizing the minimum eigenvalue in E-optimality, or minimizing various properties of the covariance matrix, such as the trace or determinant for A- and D-optimality, respectively, as applied in Kollar and Roy (2008) and Binney et al. (2013). Entropy and mutual information are also common information measures that have been investigated in many informative path planning applications such as in Singh et al. (2009) and Hollinger et al. (2013). These metrics are popular due to their submodularity property which provides performance guarantees on the associated greedy policy.

The idea of using information gain rewards to direct exploration in RL has been gaining momentum. Engel et al. (2005) suggested using confidence intervals derived from the GP covariance to expand the repertoire of exploration strategies used in RL. More recently, Still and Precup (2012) used the Kullback–Liebler divergence between successive estimates of the value function to quantify exploration utility, and in our prior work (Chung et al., 2013), we suggested using the change in the GP variance volume as a measure of information gain.

The GP variance over the state-action space represents a bounding volume around the estimated value function surface. Each consecutive observation results in a reduction in this volume and we define this reduction as the information gain of the corresponding state-action observation,

$$V_{bound_N} = \int_{x_{n_a}}^{x_{n_b}} \cdots \int_{x_{1_a}}^{x_{1_b}} \mathrm{cov}([x_1, \ldots, x_n]|X_N)\mathrm{d}x_1 \ldots \mathrm{d}x_n \quad (9)$$

$$I_{gain} = V_{bound_N} - V_{bound_{N+1}} \quad (10)$$

The training set X_N consists of n-dimensional state-action pairs, with dimensions $[x_1,\ldots, x_n] = [s_1,\ldots, s_u, a_1,\ldots, a_v]$;

furthermore, $X_{N+1} = X_N \bigcup \mathbf{x}_{N+1}$. The limits of integration in equation (9) can be chosen to incorporate the entire space or only a local portion of it, for example, a reachable set within a finite time horizon. In the following experiments, the integral is taken over the entire state-action space, that is, the limits are chosen as the *max* and *min* values for each dimension, $\{[x_{1a}, x_{1b}],\ldots, [x_{na}, x_{nb}]\} = \{[x_{1min}, x_{1max}],\ldots, [x_{nmin}, x_{nmax}]\}$.

Since the squared exponential covariance function given in equation (8) is an integrable function, an analytical solution to equation (9) can be found:

$$\begin{aligned} V_{bound_N} &= \int_{x_{n_a}}^{x_{n_b}} \cdots \int_{x_{1_a}}^{x_{1_b}} k([x_1, \ldots, x_n], [x_1, \ldots, x_n]) \\ &- k([x_1, \ldots, x_n], X_N) K_{XX}^{-1} k(X_N, [x_1, \ldots, x_n]) \mathrm{d}x_1 \ldots \mathrm{d}x_n \\ &= \sigma_f^2 \prod_{m=1}^{n} (x_{m_b} - x_{m_a}) - \sigma_f^4 \left(\frac{\sqrt{\pi}}{2}\right)^n \prod_{m=1}^{n} l_m \\ &\times \sum_{i,j} \left(K_{XX}^{-1}\right)_{i,j} \exp\left[-\sum_{m=1}^{n} \left(\frac{X_{m_i} - X_{m_j}}{2 l_m}\right)^2\right] \times \\ &\prod_{m=1}^{n} \left(\mathrm{erf}\left(\frac{x_{m_b} - \frac{X_{m_i} + X_{m_j}}{2}}{l_m}\right) - \mathrm{erf}\left(\frac{x_{m_a} - \frac{X_{m_i} + X_{m_j}}{2}}{l_m}\right)\right) \end{aligned} \tag{11}$$

where k is the squared exponential covariance function, and $K_{XX} = [K(X, X) + \sigma_n^2 I]$ is the covariance matrix of all the observed locations including additive white noise drawn from $\mathcal{N}(0, \sigma_n^2)$.

4.2. Rollout

The information *reward* of taking an observation at a particular state-action is simply computed as equation (10), however, approaching this from an RL perspective, it is more meaningful to consider the nonmyopic information *value*, that is, the total future information gain possible due to performing a particular action. There is one key difference to note between the RL state-action value and this information value: the RL reward for any state-action is constant whereas the information reward for reobserving a particular state-action decreases with the number of observations. The main consequence is that the information reward cannot be directly included in the computation of Q, and similarly, information gain credit cannot be meaningfully assigned back along the eligibility trace. Instead, we take the approach of looking at all the possible future state-action observations leading out from a particular action and use the discounted sum of those rewards to compute the information value.

The information gain rollout technique is illustrated in Figure 1. The information value sums the information gained from the next proposed transition $s',a' \rightarrow s''$, where $a' \in \mathcal{A}(s')$ and $\mathcal{A}(s')$ is the set of available actions from state s', with the discounted information gained from all possible future state-actions rolled out from s'' up to a

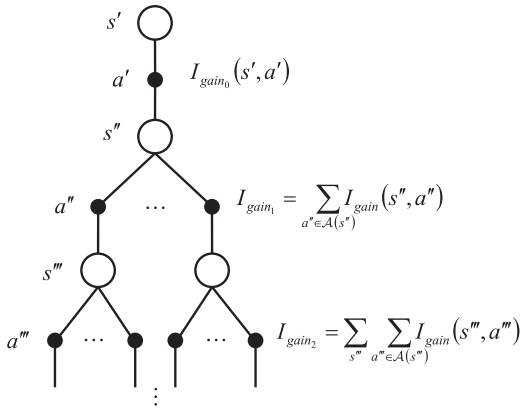


Fig. 1. The rollout method introduced in Chung et al. (2013). The information gain of each rollout level considers all the reachable state-actions at that level. The total information gain is a discounted sum of the information gain of each level.

threshold discount factor. Given a discount parameter γ_r, the total information gain of an action a is computed as

$$I_{a_{total}} = I_{gain_0} + \gamma_r I_{gain_1} + \gamma_r^2 I_{gain_2} + \ldots + \gamma_r^p I_{gain_p} \tag{12}$$

where p is the highest integer for which γ_r^p is greater than the discount threshold γ_{thres}.

5. Action selection

The goal of the action selection function in the RL problem is to consolidate the two competing objectives of *exploration* and *exploitation* to improve the overall value function estimate. In this regard, the task of designing an action selection objective function is similar to the problem of designing acquisition functions in the Bayesian optimization framework, as surveyed in Brochu et al. (2010), which similarly combines the GP variance and mean in various ways to promote exploration. The information value and state-action value are our two respective metrics and a judicious combination of the two into a single objective function can produce the desired behavior from the learning agent. There are a number of factors to consider: firstly, the scale of each value may be very different, resulting in one dominating the other if combined naïvely. Secondly, as noted in Section 4.2, having defined the information reward as the change in the GP variance volume, the value of information naturally decreases as more observations are taken. Finally, given a system subject to resource constraints, such as available platform energy, the desire to explore or exploit should adapt according to the current available resources. These issues will be addressed in the following subsections.

5.1. Reward trade-off

The ranking of the available actions according to the objective function ultimately determines which action is executed. Thus, it is useful to normalize the values when

combining them as this also removes the problem of their differences in scaling. Given the estimated values $\bar{Q}$, let the normalized values $\hat{Q}$ be

$$\hat{Q}_i = \frac{\bar{Q}_i}{\max|\bar{Q}_i|} \tag{13}$$

and similarly for the information values,

$$\hat{I}_i = \frac{I_{i_{total}}}{\max|I_{i_{total}}|} \tag{14}$$

where i is the current timestep. Note that the value estimate component lies within $[-1, 1]$ while the information gain component lies within $(0, 1]$.

5.1.1. Timestep-weighted information decay (iGP-SARSA(λ)). A side-effect of this normalization is that the natural decline in information value as more observations are taken is also eliminated. Generally it is desirable to maintain this decline in the final action selection objective function to encourage the learning agent to explore more at the start and less later on when the number of observations increases and confidence in the learned map increases. Chung et al. (2013) demonstrated that this decline in information value could be reintroduced as a time-dependent decreasing weighting factor on the normalized information value, and the overall effect of this was to scale the agent's behavior from exploratory to exploitative as learning progressed. If there is no set mission length or if no clear definition of a 'good enough' estimate of the value function is readily available, then it is difficult to decide how to represent the overall decline of information value if we do not allow the data to present this itself. For the purposes of comparison, we have adapted the timestep weighting presented in Chung et al. (2013) for use in the following experiments. The decay in the information weighting is introduced as

$$\omega_{t_i} = \frac{1}{1 + \frac{i}{\tau}} \tag{15}$$

giving the action selection function as

$$J_{t_i} = \hat{Q}_i + \omega_{t_i}\hat{I}_i \tag{16}$$

Without a set mission length, the variable τ was introduced to define the 'half-life' of the information value influence on the sampling policy. In the following trials, the value $\tau = 100$ was chosen by taking into account the expected mission length of each episode, particularly during the early stages of learning when it is desirable to perform exploration as well as begin to reinforce reward-gaining behavior.

5.1.2 Resource-limited information weighting (eGP-SARSA(λ)). RL research has typically dealt with problems where the single goal is to efficiently find an optimal state-action trajectory for an agent (or multiple agents) as the number of observations increases to infinity. Limitations on available resources such as platform energy, etc., have largely been neglected from such problems in existing research, and in cases where exploration costs are explicitly considered they are dealt with independently of the reward function (Kim et al., 2012). However, when the RL problem is a resource-seeking mission, the reward and the ability to continue collecting rewards become tightly coupled and such limitations can no longer be ignored. In the following experiments we consider a soaring glider learning to gather energy from a wind field. In order to do so it must expend energy to explore the space for profitable flight trajectories. The resource in this problem is the available platform energy, which determines the continuation or (critical) termination of a mission. For simplicity, in the following discussions we will focus solely on platform energy as the resource and reward of interest.

We propose a dynamic scaling of the exploration behavior according to the available platform energy by combining equation (13) and equation (14) to produce the resource-constrained exploration–exploitation objective function

$$J_{e_i} = \hat{Q}_i + \omega_{e_i}\hat{I}_i \tag{17}$$

The dynamic weighting factor is defined as

$$\omega_{e_i} = \frac{2}{\pi}\arctan(h_i) \times \left(\frac{E_i}{E_{max}}\right) \tag{18}$$

where E_i is the available platform energy at timestep i, E_{max} is the maximum amount of energy that can be stored on the platform (e.g. corresponding to a maximum speed at a maximum altitude), and

$$h_i = \left(\frac{z_i}{z_{max}}\right) \times 100\% \tag{19}$$

where $z_i \in [0, z_{max}]$ is the current altitude. The purpose of including h_i is to explicitly penalize states at low altitudes, regardless of available platform energy (which also includes the altitude in the potential energy component). The height penalty is squeezed into an arctan profile in $[0, 1)$ so that for the most part it does not greatly affect ω_e, which is dominated by the energy ratio term; however, at critically low altitudes, it drives ω_e to zero, ultimately penalizing trajectories that send the platform into the ground at high speeds.

The effect of the dynamic weighting factor in the objective function is to increase the influence of the information value when platform energy and altitude is high so that the agent tends to explore areas of the state-action space which have high uncertainty. When platform energy and/or altitude is low, the state-action value dominates the action selection objective function so that the agent tends to exploit areas of the state-action space which are believed to produce high energy reward, thereby replenishing the resource.

Algorithm 1 eGP-SARSA(λ)

$\bar{Q}_a \leftarrow 0$ ▷ Initial value estimate
Initialize θ ▷ GP hyperparameters
for each episode **do**
  $\mathbf{e} \leftarrow \mathbf{0}$ ▷Initialize trace
  $s,a \leftarrow$ initial state and action
  for each step i **do**
    $e(s, a) \leftarrow 1$ ▷Replacing traces
    Take action a, observe reward r, next state s', and current energy E_i
    $\delta \leftarrow r - \bar{Q}_a$
    for all $a'' \in \mathcal{A}(s')$ **do**
      $\bar{Q}_{a''} \sim \mathcal{GP}_Q$ ▷GP approximation for Q
      $I_{a''_{rotal}} \leftarrow I_{gain_0} + \gamma_r I_{gain_1} + \ldots + \gamma_r^p I_{gain_p}$
    end for
    $\omega_e \leftarrow \frac{2}{\pi}\arctan(h_i) \times \left(\frac{E_i}{E_{max}}\right)$
    $\hat{Q}_{a''} \leftarrow \frac{\bar{Q}_{a''}}{\max|\bar{Q}_{a''}|}$ ▷Normalized state-action value
    $\hat{I}_{a''} \leftarrow \frac{I_{a''_{total}}}{\max|I_{a''_{total}}|}$ ▷Normalized information value
    $J_{e_{a''}} \leftarrow \hat{Q}_{a''} + \omega_e \hat{I}_{a''}$
    $a' \leftarrow \text{argmax}_{a''} J_{e_{a''}}$
    $\delta \leftarrow \delta + \gamma \bar{Q}_{a'}$
    if sparsify **then**
      if $\beta_i > \beta_{tol}$ **then**
        Append (s, a) to $\mathcal{GP}_Q$ training inputs X
        if $|\mathcal{BV}| >$ budget **then**
          Delete training input with lowest score
        end if
      end if
    else
      if (s, a) is a new state-action **then**
        Append (s, a) to $\mathcal{GP}_Q$ training inputs X
      end if
    end if
    $\mathbf{y} \leftarrow \mathbf{y} + \alpha\delta\mathbf{e}$ ▷Update $\mathcal{GP}_Q$ training targets
    if retrain hyperparameters **then**
      $\theta \leftarrow \text{argmin}_\theta - \log p(\mathbf{y}|X,\theta)$ ▷Minimize the negative log marginal likelihood
    end if
    $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e}$
    $s, a \leftarrow s', a'$
  end for
end for

6. eGP-SARSA(λ)

The energy-weighted action selection objective function is applied directly into the SARSA(λ) algorithm with GP value function approximation. We present the full algorithm, eGP-SARSA(λ), in Algorithm 1.

For each action selection, Algorithm 1 requires simulation of the motion model and estimation of the resulting information gain across a tree with branching factor in the number of possible actions and search depth imposed by the limit γ_r. It is possible, at each depth, to compute the information gain components of each branch in parallel, reducing the exponential complexity of traversing the tree to a constant multiple of the search depth. However, practically speaking, it would be a challenge to mount the number of processors required to maintain this computational speed.

6.1. Greedy rollout

To prevent the exponential branching of the information gain rollout tree, a directed sampling method can be applied to restrict the rollout to only include the expected future state-actions. In other words, only consider the information gain from future state-actions where the actions are selected from an approximation of the current policy,

$$\tilde{J}_{e_a} = \hat{Q}_a + \omega_e \frac{I_{a_{gain_0}}}{\max|I_{a_{gain_0}}|} \tag{20}$$

when computing the information value. The rollout diagram for this *greedy rollout* method is similar to the backup diagram for Q-learning and is shown in Figure 2.

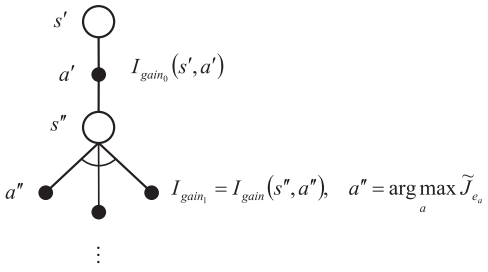


Fig. 2. Rollout diagram for the greedy rollout method. Only the information gain from the expected state-actions of the current policy are considered in the calculation of the information value.

Enforcing this restriction on the state-actions considered in the information gain rollout ensures that the computation of the information value is linear in the rollout depth. That is, the cost of computing the information value becomes $\mathcal{O}(N^2tM)$, where the N^2 term arises from the evaluation of the GP covariance, there are t layers in the search tree, and the motion model computation has complexity $\mathcal{O}(M)$. It is expected that when the current policy is in a state of flux, the greedy rollout information value will poorly represent the actual future information gains since the state-action value function, and consequently the policy, is likely to change. In contrast, this issue does not exist when the full information rollout is taken since all possible actions are considered and contribute to the information value.

6.2. Sparsification

One of the main drawbacks of using GP regression for value function approximation is the $\mathcal{O}(N^3)$ inversion of the covariance matrix at each update step in equation (6) and equation (7), where N is the number of training inputs to the GP. By updating a single observation at a time, we can take advantage of the matrix inversion lemma to reduce this to $\mathcal{O}(N^2)$, however, this operation is still prohibitive over the long training sequences which are a feature of RL. To bound the computation time, we place a budget on the number of inputs used to train the GP and apply the sparsification method described by Csató and Opper (2002). This sparsification technique provides a method for selecting observations to include in the GP training input set, or basis vector ($\mathcal{BV}$) set, that best reconstructs the underlying value function according to a linear independence test.

Briefly, the variance at the current observation point $\mathbf{x}_{i+1}$ gives a measure of its linear independence or 'novelty',

$$\beta_{\mathbf{x}_{i+1}} = k(\mathbf{x}_{i+1}, \mathbf{x}_{i+1}) - K(\mathbf{x}_{i+1}, X_N)K(X_N, X_N)^{-1} K(X_N, \mathbf{x}_{i+1}) \quad (21)$$

Intuitively, $\beta_{\mathbf{x}}$ computes the uncertainty of the estimate at $\mathbf{x}$ given the current training set X_N and assuming a noise-free observation model. As described in Csató (2002), points with β greater than the tolerance value β_{tol} are included in the training set. If the number of points in the $\mathcal{BV}$ set exceed the budget, then the observation with the lowest score as defined by

$$\varepsilon_i = \frac{K_{XX}^{-1} y_i}{K_{ii}^{-1} - K_{XX_{ii}}^{-1}} \quad (22)$$

where K_{ii}^{-1} is the ith diagonal element of K^{-1} and similarly for $K_{XX_{ii}}^{-1}$, is removed from the $\mathcal{BV}$ set. By keeping track of the relevant inverse matrices and applying the matrix inversion lemma at each update/downdate step, it is possible to bound the GP computation time to $\mathcal{O}(N_{max}^2)$, where N_{max} is the maximum number of training inputs as defined by the computation budget.

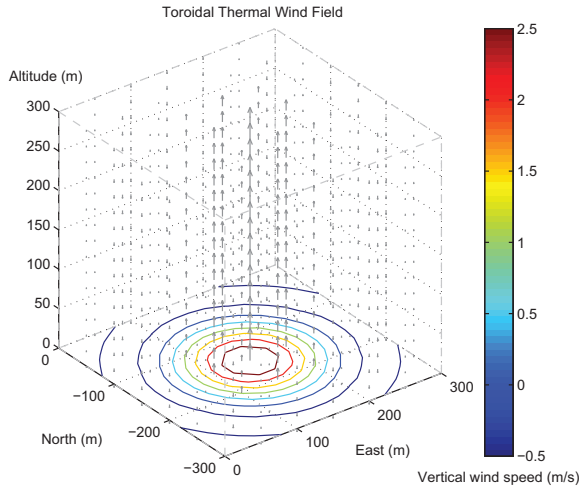


Fig. 3. Toroidal thermal model of a thermal updraft. The model is dominated by the central column of rising air that is offset by the thinner surrounding ring of sinking air. Small lateral wind components also exist outside the flight boundary at the upper and lower extremities of the thermal to maintain the toroidal flow of the air.

7. Experimental setup

The algorithm was tested on a 3D 6DOF soaring glider simulation under a static (thermal) soaring scenario. As explained in Section 5.1.2, we wish to investigate RL problems where the reward and the ability to seek reward are tightly coupled. In the soaring scenario, the reward is the energy gained by the glider agent flying particular trajectories in the wind field; to learn these trajectories, the agent must expend energy to explore the state-action space. The goal of these experiments was to show that given a wind energy field consisting of a single thermal, the proposed resource-constrained learning algorithm is able to effectively explore the state-action space to generate energy-positive trajectories. In the early stages of learning, it was expected that many of the trajectories would result in failure or with the glider exiting the field, thus the problem was formulated as an episodic learning task where each episode was terminated due to the glider either crashing or exiting the designated field.

7.1. Thermal wind field

The toroidal thermal model is a realistic 3D updraft model that is commonly used in the autonomous soaring literature (Bencatel et al., 2013). The toroidal thermal model developed in Lawrance and Sukkarieh (2011) and shown in Figure 3 was the 3D wind field used in the following simulations. The thermal model is conservative in its flow, meaning that there is no net horizontal or vertical flow. This is achieved via symmetry in the horizontal flow at the upper and lower extremities of the thermal, and balanced rising and sinking volumetric flow in the vertical plane. In this way, the thermal core of strong rising air is offset by a surrounding column of slow sinking air, while the lateral transitions from rising to sinking air at the top of the thermal are equal and opposite to the transitions from sinking to rising air at the bottom of the thermal.

The shape of the thermal model is determined by three main terms: the core vertical wind speed, w_{core} = 3m/s; the major radius, R_{therm} = 100m; and the elliptical factor, k_{therm} = 100. The core vertical wind speed is scaled against a sinusoidal function of the thermal radius to form a smooth vertical wind profile. The elliptical factor determines the contribution of the lateral wind components: by choosing a high value for k_{therm} the thermal model was dominated by the vertical wind flow.

7.2. Glider platform

The dynamic model for the glider is a non-linear aerodynamic point mass model based on the RnR SBXC 4.32m wingspan scale model glider (see the Appendix for platform specifications). The applied forces are the aerodynamic force (decomposed into lift and drag) and the weight force. Lift is limited by the maximum lift coefficient and load factor constraints. Body force due to sideslip is not considered. A flat Earth model is assumed due to the relatively small scale of the aircraft and flight paths, so the weight force is directed down. Since this is a glider model the effect of wind is an important consideration. This model includes effects for the changes in lift and drag due to the wind and locally linear spatial wind gradients but does not account for moments imparted by differential wind across the lifting surfaces. The model equations can be found in detail in Lawrance (2011). The simulation is performed using numerical integration of the non-linear equations of motion.

Control is modeled using commanded inputs of roll and pitch rates. At each control timestep there are three discrete commands available for both roll and pitch, such that a single action

$$\mathbf{a} = \left[a_{\dot{\phi}}, a_{\dot{\theta}}\right] \tag{23}$$

gives rise to a total of nine available actions in the action set. Intuitively, the roll rate commands result in banking further to the left or right, or maintaining the current bank angle, and similarly for the pitch rate commands which control the aircraft climb angle. The actions are limited such that they will not generate commands for bank angles over $\pm$ 45° or pitch angles which would result in a stall or dive steeper than 50° so that all actions in the action set should always be achievable.

In the state-action value function approximation, the pitch and roll commands were regarded as separate dimensions in the training input space since they generated responses in orthogonal axes of the aircraft's body frame. As such, no meaningful length scale or metric could be defined to measure the 'difference' between banking and pitching, and so it was left to the GP to model the interactions between these commands and the expected value.

7.3. Reward function

The reward function used was based on the specific energy gained by the platform during each state-action transition; penalties were also applied when the glider stalled and if it dropped below an altitude of 0m. While the former condition was recoverable, the latter indicated a critical failure; thus the reward function was designed to reflect the severity of these states. The penalty incurred for stalling was computed as 25% of the specific kinetic energy at stall velocity,

$$r_{stall} = -25\% \times \frac{1}{2} v_{stall}^2 \tag{24}$$

while the penalty for crashing (altitude $\leq$ 0 m) was defined as twice the maximum allowable specific energy in the flight envelope,

$$r_{crash} = -\frac{2E_{max}}{m} \tag{25}$$

where E_{max} corresponds to the platform energy when flying at the maximum allowable speed and altitude, m is the platform mass and v_{stall} is the stall velocity of the platform. The complete reward function was computed as

$$r_i = \frac{E_i}{m} + stall \times r_{stall} + crash \times r_{crash} \tag{26}$$

where *stall* and *crash* are Boolean indicators for the two respective conditions.

7.4. Relative states

To use the full glider state space would require learning over 13 state dimensions in addition to the two action dimensions, rendering the learning task infeasible given the prohibitively large number of points required to generate a reasonable GP model of the entire state-action space. The problem can be simplified if it is assumed that the thermal center is known. The states of interest, that is, primary contributors to the reward function by virtue of wind field geometry, can therefore be defined as the distance to the thermal center, the bearing to the thermal center and the airspeed:

$$\mathbf{s} = [r_{therm}, \psi_{therm}, v_a] \tag{27}$$

The value function then becomes a function of the states and actions presented in equation (27) and equation (23).

The full glider state is still maintained since it is required in the rollout (where the state transition is performed under the assumption of no wind) and the reduced relative states are not Markovian in this regard. In fact, without full information of the wind field, the full glider state is also not strictly Markovian, however, we rely on smoothness in the GP model to handle variability in the states introduced by the unknown thermal wind field.

It is noted that although the r_{crash} component of the reward is a function of the altitude, this variable is not included as a dimension in the relative learning state. In fact, the event of crashing is linked to the climb angle and airspeed of the aircraft, and these two variables are themselves linked through the glider dynamics shown in Lawrance (2011). In the results given in the next section, we show that the RL algorithm learns to avoid high airspeeds since there is a strong correlation between this and receiving a strong negative reward due to r_{crash}.

8. Results

The following five learning algorithms were tested:

- eGP-SARSA(λ) with full rollout information value;
- iGP-SARSA(λ) with full rollout information value, τ_r = 100 (outlined in Section 5.1.1);
- eGP-SARSA(λ) with greedy rollout information value (outlined in Section 6.1);
- sparse eGP-SARSA(λ) with full rollout information value, 1000 $\mathcal{BV}$, $\beta_{tol} = \sigma_n^2$ (outlined in Section 6.2);
- GP-SARSA(λ) with ε-greedy exploration, ε = 0.01.

Each algorithm was run over 10 trials, each with a learning period of 50 episodes. The learning agent began each episode in a random location in the field with a bearing of 0° to the thermal center: these starting locations were consistent across all the tested learning algorithms.

8.1. Learning rates

The progression of the average reward across the episodes for each tested algorithm is shown in Figure 4; the average of each set of 10 trials is plotted for each episode along with 95% confidence intervals at five-episode intervals. The confidence intervals plotted for each algorithm are offset by one episode each to improve clarity, however, the confidence intervals for all of the tested algorithms are shown for the final episode. The graph shows that both the greedy and full rollout eGP-SARSA(λ) algorithms promptly converge to a higher average reward and are followed closely by GP-SARSA(λ) with ε-greedy sampling; however, the latter has a larger confidence interval than either of the eGP-SARSA(λ) cases. The iGP-SARSA(λ) trials produce the lowest initial average rewards and also display a high level of variation between the rewards gained in each

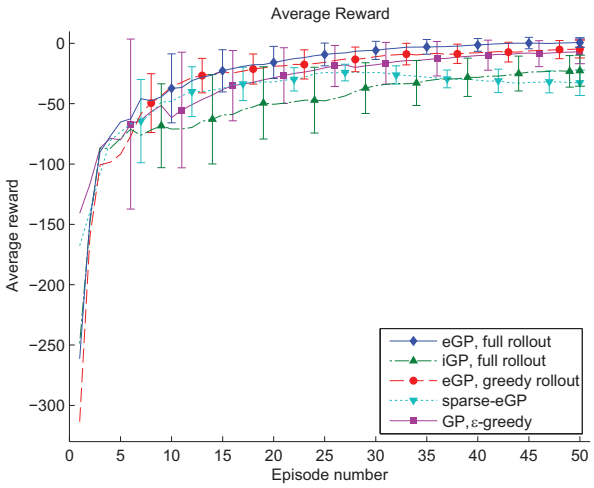


Fig. 4. Progression of the average reward across the episodes. The plots compare the performance of eGP-SARSA(λ) with full rollout, iGP-SARSA(λ) with full rollout, eGP-SARSA(λ) with greedy rollout, sparse eGP-SARSA(λ) with $\beta_{tol} = \sigma_n^2$, and GP-SARSA(λ) with ε-greedy sampling. The average of each set of 10 trials is plotted along with 95% confidence intervals at five-episode intervals, and the episodes plotted for each algorithm are offset by one episode each to improve clarity; however, the data at episode 50 are shown for all the tested algorithms.

simulation. The graph also shows the departure of the sparse-eGP approach in terms of reward-gaining performance as the training input set is reduced during the early episodes. Indeed, the average reward achieved begins to decrease during the later episodes and even drops below that of iGP-SARSA(λ) after episode 37. This suggests that the size of $\mathcal{BV}$ may be insufficient for representing the full state-action space or that the linear independence metrics used to reject and discard observations from $\mathcal{BV}$ are inadequate for this learning task.

Table 1 presents the average number of observations and the average size of the training input set at the end of each simulation case. The sparse-eGP case encountered the fewest total observations, followed by the ε-greedy case. Furthermore, the sparse-eGP case on average rejected 435 observations from the $\mathcal{BV}$ set, which is over half the final average $\mathcal{BV}$ size. The linear independence test used to reject observations from $\mathcal{BV}$ uses the squared exponential covariance function, and therefore favors observations taken in isolated state-action locations. However, for the problem of learning to soar in a static thermal, energy-gaining flight can only occur in a local region around the thermal center where it is necessary to have a more refined policy. In regions further away from the thermal center, a reasonably coarse policy can be applied to direct the glider towards the energy-gaining regions. Therefore, a uniform density of training inputs over the entire state-action space may not provide as good a model for learning the policy as a distribution of inputs that concentrate observations in the regions of interest.

The progression of the average specific energy gain across the episodes is shown in Figure 5. The average

Table 1. Average observations and training inputs.

Algorithm	Average observations	$\mathcal{BV}$ size
eGP-SARSA(λ), full rollout	3448	3448
iGP-SARSA(λ), full rollout	2702	2702
eGP-SARSA(λ), greedy rollout	2118	2118
sparse-eGP, $\beta_{tol} = \sigma_n^2$	1264	829
GP-SARSA(λ), ε-greedy	2072	2072

specific energy profiles are similar to the average reward profiles, since a large portion of the reward is derived from the energy gain, however, the reward also includes the r_{stall} and r_{crash} components. Looking purely in terms of energy gain, it can be seen that the sparse-eGP case performs the worst of all the tested algorithms, with iGP-SARSA(λ) overtaking its energy-gaining performance much earlier in episode 5. The average specific energy gain per step for the sparse-eGP case dropped to −5.5J/kg by the final episode; to place this value in context, it is approximately the specific drag induced by the glider flying straight and level at a speed of 15.0m/s over one timestep.

Aside from sparse-eGP, all learning algorithms were able to achieve a positive average specific energy gain by episode 50. Figure 5 shows that GP-SARSA(λ) with ε-greedy sampling achieves a higher average specific energy gain than eGP-SARSA(λ) with greedy rollout despite the latter achieving a greater average reward in Figure 4. Both algorithms have much lower average energy gain profiles than eGP-SARSA(λ) with full rollout information value. Furthermore, eGP-SARSA(λ) with full rollout has a more consistent energy-gain performance as shown by the significantly smaller confidence intervals.

The reason for the disparities between Figures 4 and 5 comes down to the number of *crash* incidents each algorithm experiences; while *stall* events also lower the average reward, it is the heavy cost of crashing that has the greatest influence on the average reward. The termination statistics for all 10 trials of each algorithm are shown in Table 2: a successful termination is defined as one in which the glider gains enough energy to soar out through the upper flight boundary. Although eGP-SARSA(λ) with greedy rollout and GP-SARSA(λ) have similar termination statistics, the four fewer *crash* incidents of eGP-SARSA(λ) with greedy rollout boost its reward-gaining performance above that of ε-greedy, despite having slightly fewer successful terminations. The iGP-SARSA(λ) terminations also help to explain its poor reward-gaining performance; although iGP-SARSA(λ) has the second-highest number of successful terminations, 141, it also has the highest number of *crash* terminations at 77. These termination statistics show that eGP-SARSA(λ) with full rollout performs the best overall in terms of successful flights. Over half of all the episodes for full rollout eGP-SARSA(λ) terminated with the glider gaining enough energy to soar out through the upper flight boundary. Although it also experienced 33 crash

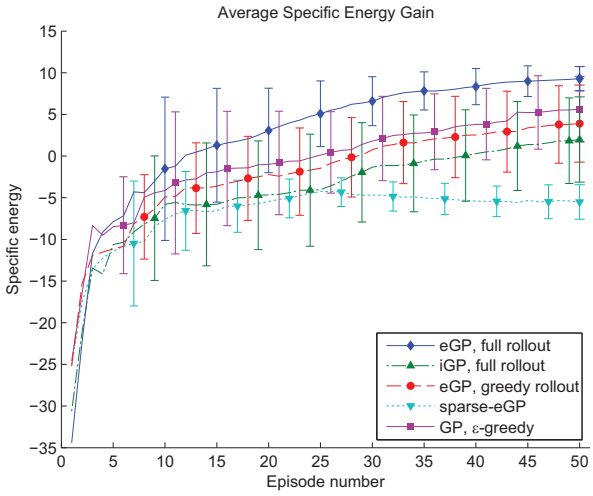


Fig. 5. Progression of the average specific energy gain across the episodes.

Table 2. Episode termination conditions.

Algorithm	Successes	Crash terminations	Lateral boundary termination
eGP-SARSA(λ), full rollout	256	33	211
iGP-SARSA(λ), full rollout	141	77	282
eGP-SARSA(λ), greedy rollout	78	18	404
sparse-eGP, $\beta_{tol} = \sigma_n^2$	8	58	434
GP-SARSA(λ), ε-greedy	82	22	396

terminations, this only represents 12.9% of the number of successful terminations, as compared to 54.6% for iGP-SARSA(λ), 23.1% for eGP-SARSA(λ) with greedy rollout, 725% for sparse-eGP, and 26.8% for ε-greedy.

Of the four different sampling policies, Figures 4 and 5 show that eGP-SARSA(λ) is more efficient at learning energy/reward-gaining trajectories than eGP with greedy rollout, ε-greedy and iGP with full rollout. This is due to the ability of the resource-constrained policy to manage platform energy such that the agent could fly consistently longer trajectories in each episode, shown in Figure 6, thereby visiting more state-action locations and learning a better model of the value function. The curves plotted in Figure 6 are consistent with the successful termination statistics given in Table 2, however, it is interesting to observe the progression of the sparse-eGP curve, which initially follows the iGP-SARSA(λ) plot but diverges rapidly after episode 25. Greedy rollout eGP-SARSA(λ) and GP-SARSA(λ) with ε-greedy sampling have very similar profiles that, from the very first episode, are slower than full rollout eGP-SARSA(λ) and iGP-SARSA(λ) to accumulate

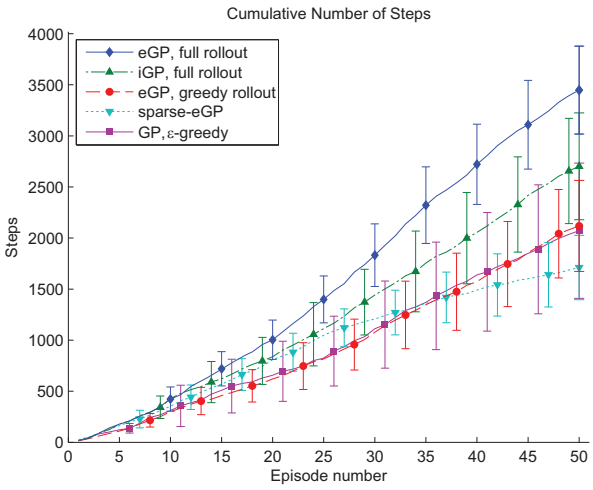


Fig. 6. Comparison of the cumulative number of steps across the episodes for each simulation case.

steps. The reason for the initial divergence between the full and greedy rollout can be attributed to the poorly modeled value function at the start of learning. While the full rollout accounts for the information gain of all the possible future state-actions, the greedy rollout only considers the information gain of those state-actions it expects to visit according to the current value function estimate. Since the value function is poorly modeled at the start of learning, the greedy rollout information value consequently computes a poor estimate of the information value.

8.2. Learned value function

The learned value function for one trial of eGP-SARSA(λ) at 10m/s velocity intervals up to the maximum allowable airspeed is shown in Figure 7. The plots show the estimated state-action value for the best set of turning actions, $[a_{\dot{\phi}}, a_{\dot{\theta}}]$, at each state. As expected, higher values are

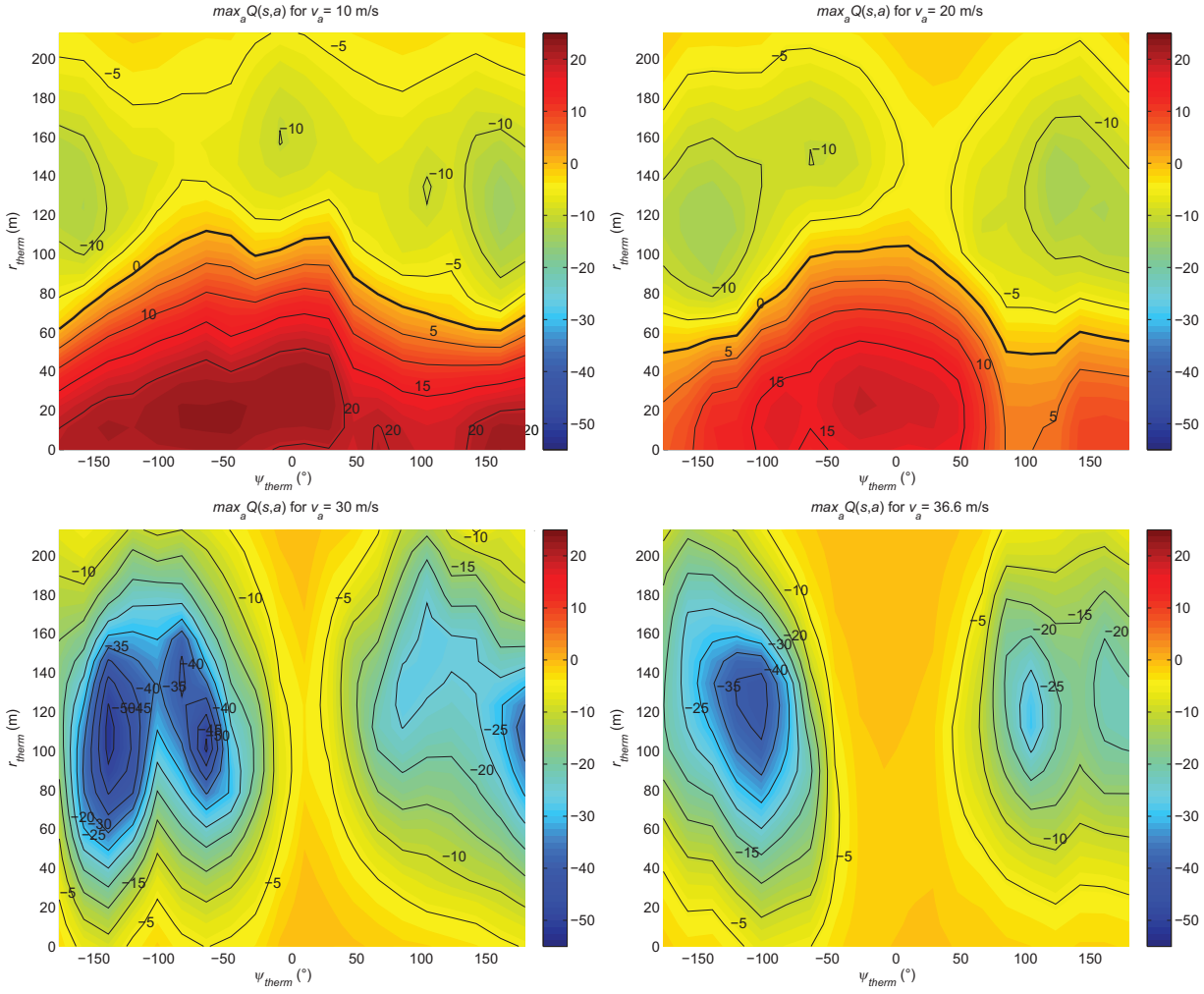
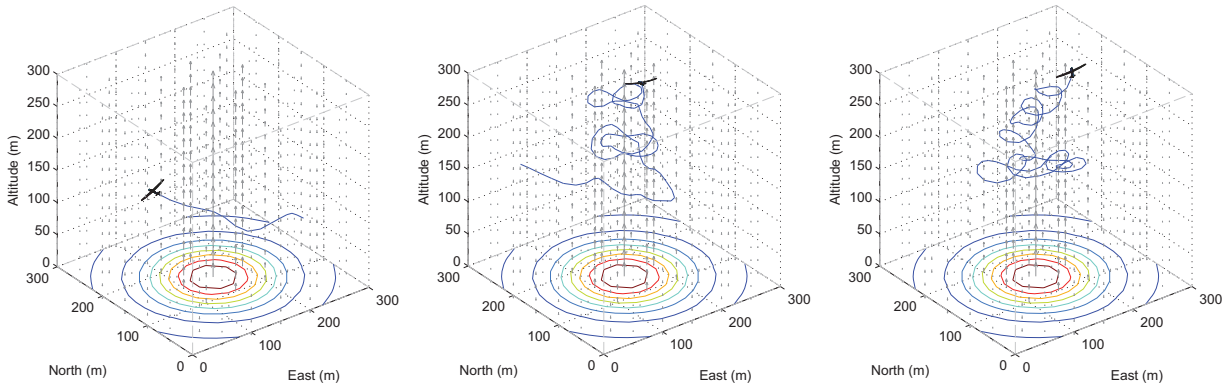


Fig. 7. The full eGP-SARSA(λ) value function for one trial evaluated at the best action for each state. There is a distinct change between the values at v = 20m/s and v = 30m/s : this is because the slowest airspeed for which the agent experienced a *crash* event was v = 30.9m/s. The strong negative reward signal observed at this state-action caused a steep reduction in the values around this airspeed. The values plotted for v = 36.6m/s appear to improve upon those at v = 30m/s, however, this is due mostly to the fewer observations made at and around the higher airspeed.



(a) Episode 1 (b) Episode 20 (c) Episode 50

Fig. 8. Evolution of the learned flight path for one trial of the eGP-SARSA(λ) case. The glider operates in a flight boundary defined by a 300 × 300 × 300m^3 volume. In these figures, the glider is enlarged by a factor of 10 so that it can be seen more easily. During the first episode, the glider performs largely random actions; in episode 20 the agent successfully gains enough energy to exit the field via the upper boundary; the flight path of episode 50 shows a consistent looping trajectory.

predicted for states close to the thermal center; at states further away, bearings closer to 0° produced higher values. The wraparound condition enforced by the GP distance measure can also be observed along the bearing axis.

There is a distinct change from positive values to negative values as the airspeed increases showing the agent learned that the event of crashing, which generates a strong negative reward from r_{crash}, tended to occur at high velocities. In this way, the two negative reward components, r_{crash} and r_{stall}, effectively placed an upper and lower bound on the glider airspeed.

It was expected that the value function would be symmetrical in the bearing axis since, if all else was equal, turning clockwise or anticlockwise should give equivalent energy-gain profiles. Figure 7 gives a hint of this symmetry, however, it appears that during the learning process, the agent favored making anticlockwise loops around the thermal center. This may be due to the fact that the agent traveled in an anticlockwise trajectory during the first episode in which it successfully gained enough energy to soar out through the upper boundary, thereby creating an initial bias for later trajectories.

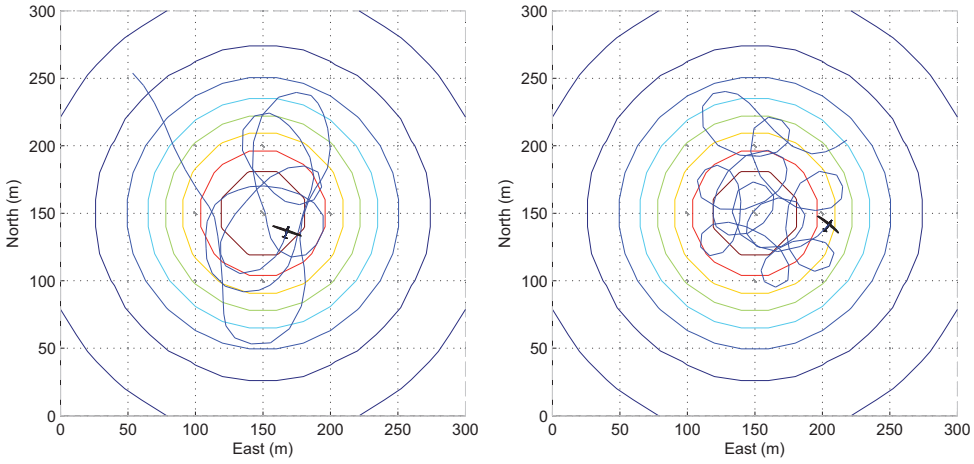
8.3. Flight trajectories

Successive flight trajectories from one trial of the eGP-SARSA(λ) simulation are shown in Figure 8. Unsurprisingly, in the first episode, the actions executed by the agent are unsuccessful in gaining energy and the episode ends with the glider exiting the lateral boundary of the field. Episode 20 is the first instance in which the agent gains enough energy from the thermal throughout the trajectory, allowing the glider to soar out through the upper boundary of the field. The trajectory in this episode is not a smooth rise, nor does it have a smooth circling motion, which would result in higher energy losses due to increased drag. This is in contrast to the trajectory executed in the final episode, which features both consistent altitude gain and a more regular turn radius. Incidentally, both episodes took 107 steps to completion; however, the agent began at a higher altitude in episode 20 but only achieved a lower energy gain rate of 12.6J/kg per step, as compared to the energy gain rate of 15.1J/kg per step achieved in episode 50.

The aerial view of the flight paths for episodes 20 and 50 is also shown in Figure 9. These paths illustrate the exploration behavior generated by eGP-SARSA(λ). In episode 20, the glider traverses a range of radial distances as it gains more platform energy and can afford to explore more of the state-action space, and in episode 50, the flight path resembles a superhelix, which is able to rapidly sample a range of radial distances and bearings.

9. Conclusions and future work

This paper presented the eGP-SARSA(λ) algorithm for directed, adaptive and nonmyopic exploration in resource-constrained RL problems. Two modifications to this algorithm were also presented to improve the computational performance. The eGP-SARSA(λ) algorithm with greedy rollout considers only the information gain from the expected state-action values according to the current estimate of the policy, and the sparse-eGP algorithm uses a sparsification method for reducing the number of observations used to train the GP. The results presented in this paper demonstrate the ability of the full rollout information value eGP-SARSA(λ) algorithm to learn complex reward-gaining trajectories under resource-constrained conditions. The normalized information value provided a quantitative measure of exploration utility such that by applying a dynamic weighting factor between the state-action value and information value, it was possible to scale exploration



(a) Episode 20 (b) Episode 50

Fig. 9. Aerial view of the flight paths shown in Figure 8: the glider has been enlarged by a factor of five so that it can be seen more easily. These paths illustrate the exploration behavior generated by eGP-SARSA(λ). In episode 20, the agent explores across the range of radial distances to the thermal center, while in episode 50 the aerial view shows that the agent flight path resembles a superhelix that samples over a range of bearings and radial distances.

and exploitation according to the available platform energy. The resulting behavior produced consistently successful energy-gaining trajectories while also exploring the highest number of state-action locations of all the compared algorithms. Over a simulated learning period of 50 episodes, the eGP-SARSA(λ) algorithm with full rollout information value outperformed the ε-greedy, sparse-eGP, eGP-SARSA(λ) with greedy rollout, and iGP-SARSA(λ) sampling policies in terms of average reward per step and average specific energy gain per step for an autonomous glider agent learning to soar in a thermal wind field.

The presented learning algorithm is able to incorporate resource constraint considerations into the action selection, however, it provides no strong guarantees for remaining within the resource budget when performing exploration, as evidenced by the early trajectories when the glider exited the field or crashed. Future research will involve generating robust policies with known safe routes to avoid breaching hard resource constraints.

This work aims towards developing practical solutions for guidance of a gliding aircraft in large unknown wind fields consisting of multiple types of wind features. Soaring flight in unknown wind fields is a high-dimensional problem, and the solution presented in the current work addressed this issue by identifying a lower-dimensional feature set to represent a known wind feature. The work presented here does not attempt to estimate the parameters of the wind field and has not addressed any robustness issues associated with imperfect knowledge of the distance and bearing to the thermal center. Future work will examine hierarchical methods to autonomously identify lower-dimensional feature spaces and consider how multiple policies could be learned and then integrated into an adaptive global policy for exploration of unknown spaces with resource constraints.

Funding

This work was supported by the Australian Centre for Field Robotics at the University of Sydney.

References

Amigoni F and Caglioti V (2010) An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems* 58(5): 684–699.

Bencatel R, Tasso de Sousa J and Girard A (2013) Atmospheric flow field models applicable for aircraft endurance extension. *Progress in Aerospace Sciences* 61: 1–25.

Binney J, Krause A and Sukhatme GS (2013) Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research* 32(8): 873–888.

Brochu E, Cora VM and de Freitas N (2010) *A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning*. Technical report, University of British Columbia, Canada.

Chung JJ, Lawrance NRJ and Sukkarieh S (2013) Gaussian processes for informative exploration in reinforcement learning. In: *2013 IEEE international conference on robotics and automation*, pp. 2633–2639.

Csató L (2002) *Gaussian processes: Iterative sparse approximations*. PhD Thesis, Aston University, UK.

Csató L and Opper M (2002) Sparse on-line Gaussian processes. *Neural Computation* 14(3): 641–668.

Dearden R, Friedman N and Russell S (1998) Bayesian Q-learning. In: *Proceedings of the national conference on artificial intelligence*, pp. 761–768.

Deisenroth MP, Rasmussen CE and Peters J (2009) Gaussian process dynamic programming. *Neurocomputing* 72(7–9): 1508–1524.

Engel Y, Mannor S and Meir R (2003) Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In: *Proceedings of the 20th international conference on machine learning*, pp. 154–161.

Engel Y, Mannor S and Meir R (2005) Reinforcement learning with Gaussian processes. In: *Proceedings of the 22nd international conference on machine learning*, pp. 201–208.

Hollinger GA, Englot B, Hover FS, et al. (2013) Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research* 32(1): 3–18.

Kim D, Kim KE and Poupart P (2012) Cost-sensitive exploration in Bayesian reinforcement learning. In: *Advances in neural information processing systems* 25, pp. 3077–3085.

Kollar T and Roy N (2008) Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research* 27(2): 175–196.

Lawrance NRJ (2011) *Autonomous soaring flight for unmanned aerial vehicles*. PhD Thesis, The University of Sydney, Australia.

Lawrance NRJ and Sukkarieh S (2011) Autonomous exploration of a wind field with a gliding aircraft. *Journal of Guidance, Control, and Dynamics* 34(3): 719–733.

Levine D, Luders B and How JP (2010) Information-rich path planning with general constraints using rapidly-exploring random trees. In: *AIAA Infotech@Aerospace conference*, Atlanta, GA.

Peng J and Williams RJ (1996) Incremental multi-step Q-learning. *Machine Learning* 22(1–3): 283–290.

Rasmussen CE and Williams CKI (2005) *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press.

Rummery GA and Niranjan M (1994) *On-line Q-learning using connectionist systems*. Technical report, Department of Engineering, University of Cambridge, UK.

Singh A, Krause A, Guestrin C, et al. (2009) Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research* 34(2): 707–755.

Stachniss C, Grisetti G and Burgard W (2005) Information gain-based exploration using Rao-Blackwellized particle filters. In: *Proceedings of robotics: Science and systems*, Cambridge, MA.

Still S and Precup D (2012) An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences* 131(3): 139–148.

Sutton RS and Barto AG (1998) *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press.

Sutton RS and Singh SP (1994) On step-size and bias in temporal-difference learning. In: *The proceedings of the eighth Yale workshop on adaptive and learning systems*, pp. 91–96.

Watkins CJCH (1989) *Learning from delayed rewards*. PhD Thesis, King's College London, UK.

Appendix: Glider platform specifications

The glider platform simulated in the experiments was modeled on the RnR SBXC glider. The aircraft parameters used in the simulations are given in Table 3.

Table 3. Simulated aircraft parameter values.

Aircraft parameter	Parameter value
Parasitic drag coefficient	0.012
Wing reference area	0.95677 m^2
Wing aspect ratio	19.54
Oswald's efficiency factor	0.85
Vehicle mass	5.44 kg
Maximum positive load factor	2.0
Maximum negative load factor	0
Maximum lift coefficient	1.2
Maximum roll rate	30°/s
Maximum roll angle	60°
Maximum air relative climb angle	50°
Approximate glide ratio	30
Stall speed	9.54 m/s
Maximum allowable airspeed	36.6 m/s