2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Gaussian Processes for Informative Exploration in Reinforcement Learning

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

Abstract— **This paper presents the iGP-SARSA(λ) algorithm for temporal difference reinforcement learning (RL) with non-myopic information gain considerations. The proposed algorithm uses a Gaussian process (GP) model to approximate the state-action value function, Q, and incorporates the variance measure from the GP into the calculation of the discounted information gain value for all future state-actions rolled out from the current state-action. The algorithm was compared against a standard SARSA(λ) algorithm on two simulated examples: a battery charge/discharge problem, and a soaring glider problem. Results show that incorporating the information gain value into the action selection encouraged exploration early on, allowing the iGP-SARSA(λ) algorithm to converge to a more profitable reward cycle, while the ε-greedy exploration strategy in the SARSA(λ) algorithm failed to search beyond the local optimal solution.**

I. Introduction

RL techniques train for profitable behaviours by exploring the state and action space and observing the resulting rewards [1]. Incorporating information gain considerations can produce a more efficient and directed exploration of the state-action space. GP regression can be used to provide a probabilistic and continuous representation of the value function from which the information gain can be computed.

Bayesian probabilistic representations for policy improvement in RL have been investigated in prior work. In the Bayesian Q-learning approach taken by [2], probability distributions are maintained over the Q-values to compute the value of information of the next action. The value of information is defined as the the expected improvement in future decision quality due to the information gained from executing action a. This method is myopic by design since it only considers how the knowledge gained in the next action may affect the current preference over those available actions. The use of GPs in temporal difference learning tasks was introduced in [3] as a Bayesian approach to approximating the value function in problems with continuous state spaces. This was extended in [4] with their GPSARSA algorithm to include online action selection and policy improvement. The authors only use the GP variance to compute confidence intervals around the value estimate, however they also note that this measure could be used for various exploration strategies.

In a similar application, GPs have also been used for value function approximation in dynamic programming problems. In [5], a GP model is used to approximate the value function while a separate GP model is used to represent the uncertain system dynamics. The estimates of both GPs are used for policy iteration. In [6], both the V- and Q-functions are modelled by GPs, however, only the V-function retains the continuous space over the states since independent GP models are used to represent the Q-function at each state over the space of the available actions. A method of Bayesian active learning for action selection is described which weights the entropy at the next possible state, computed from the V-function GP variance, with the expected value of that state.

Currently, the only methods of incorporating information gain into the action selection have been myopic. However, given that in temporal difference learning techniques, the value function encapsulates the sum of all future expected rewards, it seems appropriate that this same approach should be applied to compute the value of information at any particular state-action.

This definition raises several issues that must be considered. Firstly, how should the information gain be measured? Following that, how can future information gain be combined in a disciplined manner? And finally, how can the information gain value be included with the value estimate to ultimately produce effective exploration-exploitation behaviour?

Prior work suggests using the entropy or expected variance of the sample point as a metric; given a GP model of the value function, these values can be readily computed, however they do not give a faithful representation of the uncertainty reduction over the entire state-action space, only at the queried points. Therefore, it is proposed that the reduction in the overall volume of the GP variance be used as the information gain metric. Moreover, with an integrable GP covariance function, a closed-form analytical solution can be found to compute the variance volume across the entire state-action space.

The proposed method for combining the future information gain rewards is inspired by the concept of RL backup using eligibility traces. In the SARSA(λ) method [1], the Q-function of previously visited states is updated with a discounted function (λ) of the current reward. This leaves a trace of 'eligibility' over a state history, such that a particular $Q(s, a)$ value reflects the rewards received by transitioning into that state as well as discounted rewards from states visited later in the trajectory. This concept motivated the current work which uses uncertainty in the value function to direct exploration by determining the value of collecting information along the range of possible future trajectories.

Our proposed iGP-SARSA(λ) algorithm for informative exploration in reinforcement learning was trialled on two

J.J. Chung, N.R.J. Lawrance and S. Sukkarieh are with the Australian Centre for Field Robotics, School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, NSW, 2006, Australia {j.chung, n.lawrance, salah}@acfr.usyd.edu.au

examples, a simple 1-dimensional battery charge/discharge problem and a soaring glider problem inspired by [7], [8]. Results show that in the charge/discharge problem, the iGP-SARSA(λ) algorithm encouraged exploration of the state-action space to discover the global optimal solution while the traditional SARSA(λ) algorithm using only ε-greedy search could not explore beyond the local optimal solution. In the higher dimensional soaring glider problem, the iGP-SARSA(λ) algorithm was able to explore the state-action space early on to converge to an energy gaining flight path loop significantly faster than the SARSA(λ) algorithm.

Section II describes in further detail the use of GPs for value function approximation. The proposed method of incorporating information gain into the action selection objective function is given in Section III. The setup of the charge/discharge problem and soaring glider problem is described in Section IV, with results reported in Section V. Concluding remarks are provided in Section VI.

II. Value Function Approximation with Gaussian Processes

Reinforcement learning methods are traditionally applied to problems with discrete state and action spaces, this is because the value function typically deals with discrete and repeatable instances of state-action pairs. The policy determines the action to take when a particular state is encountered and is learnt by observing the reward received from each state-action transition. Credit assignment during each backup is applied only to the state-action pairs that have been visited, i.e. the value at unobserved state-action locations cannot be inferred from the observed set. In the case of continuous state or action spaces, any arbitrary state-action will almost surely never be visited more than once, therefore the observed reward and the subsequently computed value of that state-action does not provide any usable information for determining policy improvements.

To overcome this, value function approximation techniques have been developed to cater for problems where either the state or action spaces are continuous. In the following sets of experiments, we compare our proposed method with two existing approximation methods, SARSA(λ) with tile coding and GP-SARSA(λ).

Tile coding is a form of coarse coding which discretises the continuous state or action space into sets of overlapping tiles. Each tiling (set of tiles) is offset by a random amount less than the size of one tile, and each tile represents the receptive field for one binary feature of the value function approximation. The greater the number of tilings, the greater the resolution of the final approximation. The reader is referred to [1] §8.3.2 for an in-depth discussion on the use of tile coding as a value function approximation technique.

In contrast to the tile coding representation, the GP approximation provides a continuous representation of the estimated value function. The basic GP-SARSA(λ) algorithm with ε-greedy exploration is shown in Algorithm 1. The training inputs, $X = \{\mathbf{x}_i\}_{i=1}^N$, to the GP are the observed state-action pairs and the training targets $\mathbf{y} = \{y_i\}_{i=1}^N$ are the corresponding Q-values which are updated on the fly as the eligibility trace decays (line 22). The estimated mean value $\bar{Q}_*$ and covariance $\operatorname{cov}(Q_*)$ for a test point $\mathbf{x}_*$, covariance function k and covariance matrix $K = K(X, X)$ are

Algorithm 1 GP-SARSA(λ)

```
1:  Initialise $\theta$                                  ▷ GP hyperparameters
2:  $\mathbf{e} \leftarrow \mathbf{0}$                   ▷ Initialise trace
3:  $s, a \leftarrow$ initial state and action
4:  $\hat{Q}_a \leftarrow 0$                             ▷ Initial value estimate
5:  for each step do
6:      $e(s,a) \leftarrow 1$                            ▷ Replacing traces
7:      Take action $a$, observe reward, $r$, and next state, $s'$
8:      $\delta \leftarrow r - \hat{Q}_a$
9:      if $m \geq \varepsilon$ then
10:         for all $a'' \in \mathcal{A}(s')$ do
11:             $\hat{Q}_{a''} \sim \mathcal{GP}_Q$      ▷ GP approximation for $Q$
12:         end for
13:         $a' \leftarrow \arg\max_{a''} \hat{Q}_{a''}$
14:     else                  ▷ Random exploration with probability $\varepsilon$
15:         $a' \leftarrow$ a random action $\in \mathcal{A}(s)$
16:         $\hat{Q}_{a'} \sim \mathcal{GP}_Q$           ▷ GP approximation for $Q$
17:     end if
18:     $\delta \leftarrow \delta + \gamma \hat{Q}_{a'}$
19:     if $(s,a)$ is a new state then
20:         Append $(s,a)$ to $\mathcal{GP}_Q$ training inputs $X$
21:     end if
22:     $\mathbf{y} \leftarrow \mathbf{y} + \alpha\delta\mathbf{e}$      ▷ Update $\mathcal{GP}_Q$ training targets
23:     if retrain hyperparameters then
24:         $\theta \leftarrow \arg\min_\theta - \log p(\mathbf{y}|X,\theta)$   ▷ Minimise the
        negative log marginal likelihood
25:     end if
26:     $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e}$
27:     $s \leftarrow s'$
28:     $a \leftarrow a'$
29: end for
```

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

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

The underlying assumption captured by the GP approximation is that state-action pairs close to one another in the covariance function space will produce similar 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.

The squared exponential covariance function is used in the following experiments:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T M (\mathbf{x} - \mathbf{x}')\right), \quad (3)$$

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 lengths scales in each dimension of the training input vector. The hyperparameters of the covariance function (3) are the length scales, $\mathbf{l}$, and the process variance, σ_f^2. The covariance matrix, $K(X, X)$, is the covariance function evaluated between each pair of points in the observed set, X.

The hyperparameters used in the following experiments were trained offline on a set of simulation data gathered using an initial estimate of the hyperparameters. It is possible to train the hyperparameters online as shown in lines 23-25 of Algorithm 1, however the authors note that the initial stages of learning tend to generate large changes in the value function which may lead to instability in hyperparameter training early on, thus offline training was preferred.

III. Incorporating Information Gain

The GP model of the value function provides a measure of uncertainty, in the form of a variance estimate, which can be used to direct exploration. While traditional RL techniques have induced exploration by either setting an initial biased estimate of the value function, selecting random actions with some small probability (ε-greedy) or a combination of both, these methods are not adaptive in terms of actively directing exploration to areas where information of the value function is sparse. Because of this, traditional methods are typically unable to explore beyond local optima and may fail to discover the globally optimal solution if it exists further away. There are other methods of promoting exploration that apply instrinsic motivation in the design of the reward function itself [9] [10]. However, since the ultimate goal of accumulating reward is inherently tied to accurately modelling the value function, using the GP variance to measure the information gain maintains a desirable consistency across the value function approximation and information reward. By including this measure in the action selection criteria, it is possible to direct exploration for more effective coverage of the state-action space.

A. Information Measure

The variance computed by the GP represents a bounding volume around the estimated function surface. The information gain of observing a particular state-action value can be computed as the change in this volume given the observation,

$$V_{bound_N} = \int_{x_{n_a}}^{x_{n_b}} \cdots \int_{x_{1_a}}^{x_{1_b}} \operatorname{cov}\left([x_1, \ldots, x_n] \,|X_N\right) \mathrm{d}x_1 \ldots \mathrm{d}x_n, \quad (4)$$

$$I_{gain} = V_{bound_N} - V_{bound_{N+1}}. \quad (5)$$

The training inputs X_N in (4) are n-dimensional state-action pairs, with dimensions $[x_1, \ldots, x_n]$; furthermore, $X_{N+1} = X_N \bigcup \mathbf{x}_{N+1}$.

Since the squared exponential covariance function given in (3) is an integrable function, an analytical solution to (4) can be found,

$$\begin{aligned} V_{bound_N} &= \int_{x_{n_a}}^{x_{n_b}} \cdots \int_{x_{1_a}}^{x_{1_b}} k\left([x_1, \ldots, x_n], [x_1, \ldots, x_n]\right) - \\ &\quad k\left([x_1, \ldots, x_n], X_N\right) K_{XX}^{-1} k\left(X_N, [x_1, \ldots, x_n]\right) \mathrm{d}x_1 \ldots \mathrm{d}x_n \\ &= \sigma_f^2 \prod_{m=1}^{n} \left(x_{m_b} - x_{m_a}\right) - \sigma_f^4 \left(\frac{\sqrt{\pi}}{2}\right)^n \prod_{m=1}^{n} l_m \\ &\quad \times \sum_{i,j} \left(K_{XX}^{-1}\right)_{i,j} \exp\left[-\sum_{m=1}^{n} \left(\frac{X_{m_i} - X_{m_j}}{2 l_m}\right)^2\right] \times \\ &\quad \prod_{m=1}^{n} \operatorname{erf}\left(\frac{x_{m_b} - \frac{X_{m_i} + X_{m_j}}{2}}{l_m}\right) - \operatorname{erf}\left(\frac{x_{m_a} - \frac{X_{m_i} + X_{m_j}}{2}}{l_m}\right), \end{aligned} \quad (6)$$

where k is the squared exponential covariance function, $K_{XX}^{-1} = \left[K(X, X) + \sigma_n^2 I\right]^{-1}$ is the inverse of the covariance matrix of all the observed locations including additive white noise drawn from $\mathcal{N}\left(0, \sigma_n^2\right)$, and σ_f^2 is the process variance hyperparameter of the covariance function.

The limits of integration can be chosen to incorporate the entire space or only a local portion of it, e.g. a reachable set within a finite time horizon. In the following experiments, the integral is taken over the entire state-action space.

B. Rollout

To facilitate nonmyopic exploration, the total information gain values for the action set include 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', summed with the discounted information gain from all possible future state-actions rolled out from s'' up to a threshold discount factor. Given a discount parameter γ_r, the total information gain of an action a is computed as

$$I_{a_{total}} = I_{gain_0} + \gamma_r I_{gain_1} + \gamma_r^2 I_{gain_2} + \ldots + \gamma_r^p I_{gain_p}, \quad (7)$$

where p is the highest integer for which γ_r^p is greater than the discount threshold γ_{thres}. The value of I_{gain} for each rollout level is the sum of the information gain of each state-action pair that can be reached at that level. See Fig. 1 for a graphical description.

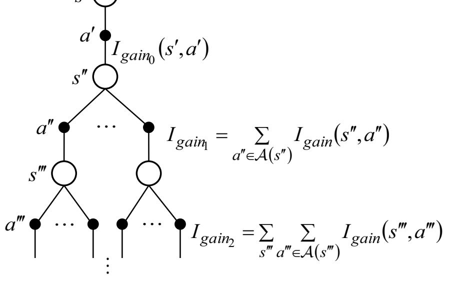

s'
a'
$I_{gain_0}(s',a')$
s''
a''
$I_{gain_1} = \sum_{a'' \in \mathcal{A}(s'')} I_{gain}(s'',a'')$
s'''
a'''
$I_{gain_2} = \sum_{s'''} \sum_{a''' \in \mathcal{A}(s''')} I_{gain}(s''',a''')$

Fig. 1: Rollout diagram showing how the information gain at each rollout level is incorporated into the total information gain calculation. The information gain of each level is computed as the sum of the information from all reachable state-actions at that level.

C. Action Selection

In traditional SARSA(λ) and basic GP-SARSA(λ), the action with the maximum value Q_a is selected for execution. Incorporating the information gain into the action selection raises a number of questions; firstly, should information gain be included in the value function approximation, and if not, how can the value estimate and the information gain components be consolidated to produce a meaningful objective function?

The value function expects repeatable rewards for revisiting a particular state-action pair, but an information gain value cannot be treated in the same way since repeat observations usually result in reducing the informative value of the observation. Furthermore, each observation changes the variance across the entire state-action space according to the covariance function. Therefore, the information gain value computed by (7) cannot be included into the value function approximation.

In the following experiments, we have chosen to combine the normalised estimated value and the normalised information gain reward through a time-dependent weighting,

$$\tilde{Q}_i = \frac{\hat{Q}_i}{|\max \hat{Q}_i|} + \left(1 - \frac{i}{N_{steps}}\right) \frac{I_{total_i}}{|\max I_{total_i}|}, \quad (8)$$

where i is the current timestep and N_{steps} is the total number of timesteps. Note that the value estimate component lies within $[-1, 1]$ while the information gain component lies within $[0, 1]$. It is desirable to normalise the two components before combining them since their magnitudes are typically of different scales, which would cause one to dominate the other if normalisation was not carried out. The weighting of the information gain component is included to promote exploration early on while decreasing its contribution in the later stages. This is motivated by prior examples of decreasing ε values over the course of training in other reinforcement learning examples [11], [12]. A decrease naturally occurs in the original I_{total} value due to the increasing number of observations included in the training set but its effect is removed by the normalisation step; we have reintroduced it as a linear decrease, however other profiles could also be considered to produce different exploration behaviours.

The action selection objective function can be written as

$$a = \underset{a \in \mathcal{A}(s)}{\arg\max}\ \tilde{Q}_i(s, a), \quad (9)$$

which replaces the ε-greedy action selection of Algorithm 1 to give the iGP-SARSA(λ) algorithm shown in Algorithm 2.

IV. Experiment Setup

The iGP-SARSA(λ) algorithm was compared against the SARSA(λ) algorithm with tile coding and the GP-SARSA(λ) algorithm shown in Algorithm 1 in two simulation experiments. The first is a battery charging/discharging problem and the second is a higher dimensional soaring glider problem.

Algorithm 2 iGP-SARSA(λ)

```
1: Initialise $\theta$                                   ▷ GP hyperparameters
2: $\mathbf{e} \leftarrow \mathbf{0}$                    ▷ Initialise trace
3: $s, a \leftarrow$ initial state and action
4: $\hat{Q}_a \leftarrow 0$                              ▷ Initial value estimate
5: for each step $i$ do
6:     $e(s, a) \leftarrow 1$                            ▷ Replacing traces
7:     Take action $a$, observe reward, $r$, and next state, $s'$
8:     $\delta \leftarrow r - \hat{Q}_a$
9:     for all $a'' \in \mathcal{A}(s')$ do
10:        $\hat{Q}_{a''} \sim \mathcal{GP}_Q$           ▷ GP approximation for $Q$
11:        $I_{a''_{total}} \leftarrow I_{gain_0} + \gamma_r I_{gain_1} + \ldots + \gamma_r^p I_{gain_p}$
12:    end for
13:    $\tilde{Q}_{a''} \leftarrow \frac{\hat{Q}_{a''}}{|\max \hat{Q}_{a''}|} + \left(1 - \frac{i}{N_{steps}}\right) \frac{I_{a''_{total}}}{|\max I_{a''_{total}}|}$
14:    $a' \leftarrow \arg\max_{a''} \tilde{Q}_{a''}$
15:    $\delta \leftarrow \delta + \gamma \hat{Q}_{a'}$
16:    if $(s, a)$ is a new state then
17:        Append $(s, a)$ to $\mathcal{GP}_Q$ training inputs $X$
18:    end if
19:    $\mathbf{y} \leftarrow \mathbf{y} + \alpha\delta\mathbf{e}$    ▷ Update $\mathcal{GP}_Q$ training targets
20:    if retrain hyperparameters then
21:        $\theta \leftarrow \arg\min_\theta - \log p(\mathbf{y}|X, \theta)$    ▷ Minimise the negative log marginal likelihood
22:    end if
23:    $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e}$
24:    $s \leftarrow s'$
25:    $a \leftarrow a'$
26: end for
```

A. Charge/Discharge Problem

Consider the task of providing energy above a particular rate from a battery with the charging and discharging profiles shown in Fig. 2. A reward of 0.01 is received when the discharge rate of the battery is greater than the reward threshold of $d_{thres} = 0.045$, while a cost of -0.01 is incurred when the battery is charging or is discharging below the reward threshold. The two profiles generate a non-convex reward surface where local optima exist between $40\% \leq energy \leq 100\%$.

At each step, two discrete actions are available: the battery can either charge up or discharge. The energy state of the battery is continuous between 0% and 100%, i.e. empty and full, respectively.

For the tile coding experiments, $m = 10$ tilings with 50 state partitions over 2 actions were used to discretise the state-action space. The step-size was chosen as, $\alpha = \frac{0.5}{m}$, with discount parameters, $\gamma = 0.9$, $\lambda = 0.7$, and random exploration probability, $\varepsilon = 0.01$.

In the GP- and iGP-SARSA(λ) experiments, the same discount parameters were used, while the step size was set to $\alpha = 0.5$. In the latter, the ε-greedy exploration probability was set to 0, thus any exploration relied solely on the information gain factor in (8). For each experiment, the battery began at a random energy state. The set of random starting states was consistent across the iGP-, GP-SARSA(λ) and SARSA(λ) experiments.

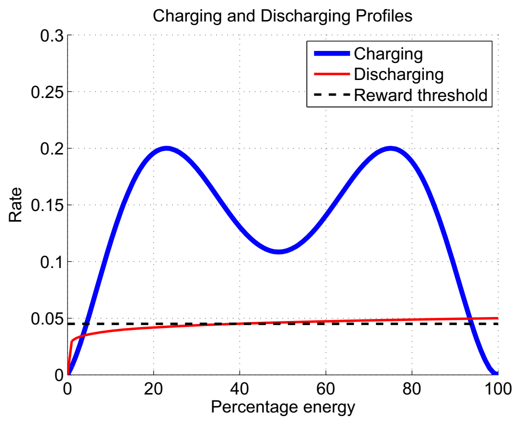


Fig. 2: Battery charging and discharging profiles as a function of the energy state. The discharge reward threshold is shown as the dashed line.

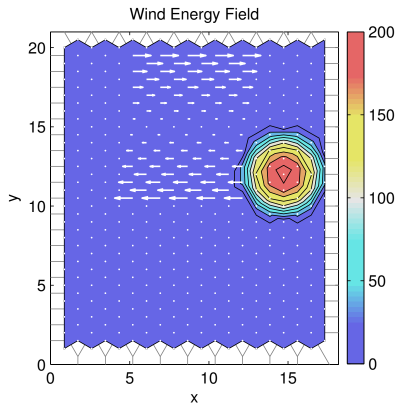


Fig. 3: Wind energy field for the soaring glider simulation. A thermal is centred at $(17, 12)$ and a wind shear field is present between the limits of $5 \leq x \leq 15$ and $10 \leq y \leq 20$.

B. Soaring Glider Problem

In the soaring glider problem, the search space is discretised into an hexagonal grid world with an underlying wind energy field as shown in Fig. 3. The glider state is defined by (x, y, ψ), the location and heading of the platform. The headings are discretised according to the six faces of the hexgrid cells, however, since this state is angular, the wraparound condition must be enforced when computing the squared distances for the GP covariance function. The glider is only able to travel within the defined search space and receives a penalty for choosing actions which attempt to move it beyond the boundaries. For each experiment, the glider began in the centre of the field facing north.

At each state, the glider has the choice of three actions: turn left, go straight or turn right. The state transition for each action is shown in Fig. 4. Headings are defined as $0°$ north, then $60°$, $120°$, etc., travelling in an anti-clockwise direction from north. Since the state-action space is discrete, tile coding was not necessary for these experiments and standard SARSA(λ) [1] was applied.

The reward functions for the hexgrid example are based on a simple glider aircraft model. During flight in no wind the aircraft is assumed to lose energy at a constant rate while travelling forwards with an additional penalty when turning. This is equivalent to a glider travelling at a constant airspeed which loses energy in proportion to the lift to drag ratio $\frac{L}{D}$, the vehicle mass m and the acceleration due to gravity g. For an electric vehicle these can be considered constant values during flight. An additional penalty is applied to turns due to the drag from increased wing loading. With the fixed turn angles of $60°$ in the hex grid example, the additional turn penalty is approximated as 1.3 times the steady-level energy loss over the same distance. The resulting cost function for a movement action between cells spaced distance d apart is

$$r_{move} = -\frac{mgd}{\frac{L}{D}} \times \begin{cases} 1 & \text{if action = go straight} \\ 1.3 & \text{otherwise.} \end{cases} \qquad (10)$$

The energy sources in the field are either rising air (static soaring) or wind gradient (dynamic soaring) sources. A rising air energy source is placed at $(17, 12)$ with a defined core wind speed w_t and radius l_t. Energy is gained from rising air simply by flying through it (independent of heading) and is proportional to the strength of the wind. At a distance d_t from a thermal the reward is

$$r_{therm} = \begin{cases} 1 - \left(\frac{d_t}{l_t}\right)^2 & \text{if } d_t < l_t \\ 0 & \text{if } d_t \geq l_t. \end{cases} \qquad (11)$$

Dynamic soaring is the process of collecting energy by moving through a spatial wind gradient which effectively increases the airspeed of the aircraft. By flying cyclic patterns through wind shear it is possible to continuously increase airspeed. Typically, dynamic soaring is performed in a vertical wind gradient (with respect to altitude) rather than the planar wind gradient used here. Whilst an in-depth discussion of dynamic soaring is beyond the scope of this paper, suitable descriptions can be found in [8], [13], [14]. The dynamic soaring sources are modelled as a planar linear shear gradient, as noted by the wind vectors in Fig. 3. Energy capture from a wind gradient is due to the increased air-relative kinetic energy from the increase in airspeed from the wind gradient. For a linear wind gradient $\frac{\partial W_x}{\partial y}$, the kinetic energy and power are

$$E_{kinetic} = \frac{1}{2} m V^2, \qquad (12)$$

$$\frac{dE_{kinetic}}{dt} = \frac{1}{2} m \left(2V\frac{dV}{dt}\right) \qquad (13)$$

$$= \frac{1}{2} m \left(\frac{\partial W_x}{\partial y} V^2 \cos\psi \sin\psi\right). \qquad (14)$$

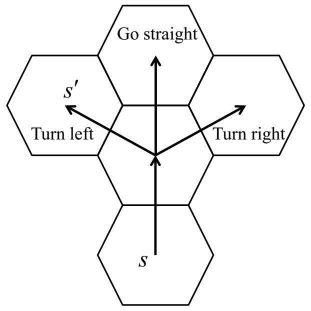


Fig. 4: State transition for each available action. Each action involves an initial forward step, the following step is either into the cell on a bearing of $60°$ to the left or right, or into the cell directly ahead.

This is effectively the projection of the wind gradient in the airspeed direction, for this simulation the resulting reward function for travelling distance d through a linear gradient $\frac{\partial W_x}{\partial y}$ at heading ψ is

$$r_{shear} = \frac{1}{2} m d \sin\psi \cos\psi \frac{\partial W_x}{\partial y} \left(2V + d \sin\psi \cos\psi \frac{\partial W_x}{\partial y} \right). \tag{15}$$

Finally, the reward function includes a penalty term for flight outside the specified area. The edge penalty is twice the magnitude of the strongest wind energy source,

$$r_{edge} = \begin{cases} -2 \max W & \text{if outside flight area} \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

The resulting reward function is the sum of these four components,

$$r = r_{move} + r_{therm} + r_{shear} + r_{edge}. \tag{17}$$

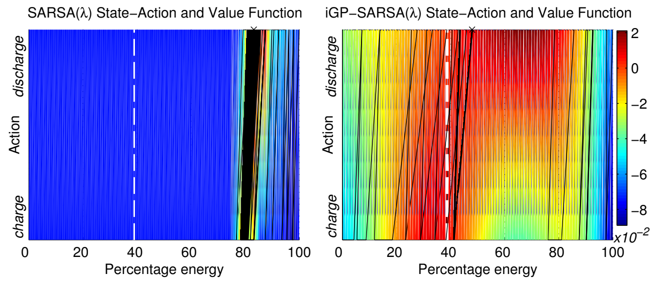
V. Results

A. Charge/Discharge Problem

The experiment was repeated over 100 trials of $500s$ simulations to compare iGP-SARSA(λ) from Algorithm 2 against GP-SARSA(λ) from Algorithm 1 and the baseline SARSA(λ) algorithm with tile coding. In each of the trials, the SARSA(λ) and GP-SARSA(λ) algorithm with replacing traces and ε-greedy search tended to converge to the locally optimal solution. The set of simulation results shown in Fig. 5 highlight this behaviour. The ε-greedy search was not able to encourage exploration beyond the local optimum around $energy = 80\%$, which produced an average reward of $r = 0.0033$, whereas the information gain component in the iGP-SARSA(λ) action selection objective function (8) promoted early exploration over the state-action space to discover the globally optimal charge/discharge cycle between $40\% \leq energy \leq 50\%$, which produced an average reward of $r = 0.005$. Furthermore, since more of the state-action space was explored in the iGP-SARSA(λ) trials, the approximated value function in these cases gave a better representation of the desired reward-gaining policy. The dashed white lines in Fig. 5 indicate the energy state above which discharging the battery will produce a reward. The learnt policy from iGP-SARSA(λ) always favours the discharging action above the charging action in these states, and prefers charging to discharging when in states to the left of this line.

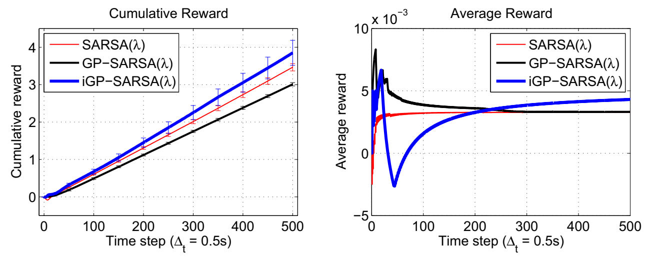
The mean cumulative reward over all trials and the average rewards for one trial are shown in Fig. 6. The steeper gradient of the cumulative reward for the iGP-SARSA(λ) case shows that overall, more profitable reward cycles were discovered. The average rewards plotted in Fig. 6 (b) show an initial dip in the reward as the iGP-SARSA(λ) algorithm explores but it is able to discover the global optimal loop after approximately $45s$.

The collated results over all 100 trials are shown in Table I. Taking the mode of the average reward per step once the SARSA(λ) algorithm converged to a solution as the benchmark, $r = 0.0033$, the iGP-SARSA(λ) method was



(a) SARSA(λ)   (b) iGP-SARSA(λ)

Fig. 5: Battery charge/discharge paths for one set of trials overlaid on top of the approximated value function. The dashed white line shows the energy at which the discharge rate is above the reward threshold. A cost is incurred if the battery attempts to discharge anywhere to the left of the dashed line. The SARSA(λ) trial coverges early on to the nearest locally optimal solution, around $energy = 80\%$, while the iGP-SARSA(λ) algorithm is able to find the globally optimal cycle between $40\% \leq energy \leq 50\%$. The final state-action location is indicated by the cross.



(a) Battery cumulative reward   (b) Battery average reward

Fig. 6: Comparison of the mean cumulative reward over all 100 trials for SARSA(λ) and GP-SARSA(λ) against iGP-SARSA(λ). The average reward received for the particular trial shown in Fig. 5 is given in (b).

able to converge to a better solution in 86% of the trials, with a 40% chance of converging to the optimal solution of $r = 0.005$. In 14% of the cases, the iGP-SARSA(λ) algorithm converged to a poorer solution than the benchmark. In comparison, the SARSA(λ) algorithm was able to find the optimal solution in only 12% of the trials with a total of only 33% of solutions greater than the mode, GP-SARSA(λ) was unable to discover the optimal solution and was only able to perform better than the benchmark in 4% of the trials. It is hypothesised that poor GP initialisation resulted in the cases where iGP-SARSA(λ) and GP-SARSA(λ) failed to find a better solution than the SARSA(λ) algorithm. Providing a sparse spread of state-action observations at the start may avoid over-reliance on the trace for awarding credit in the opening stages of learning; this is an avenue of future investigation and should be compared against the SARSA(λ) and GP-SARSA(λ) methods with the same initial observation set.

TABLE I
Results From 100 Simulation Trial Sets

Average reward $(\times 10^{-3})$	$r > 3.3$	$r = 3.3$	$r < 3.3$
% SARSA(λ) trials	33	59	8
% GP-SARSA(λ) trials	4	54	42
% iGP-SARSA(λ) trials	86	0	14

B. Soaring Glider Problem

In the higher dimensional soaring glider problem, the iGP-SARSA(λ) algorithm encourages early exploration of the state-action space but is able to balance exploration and exploitation to converge to an energy gaining flight plan loop much earlier than other two algorithms. As shown in Fig. 7, the three algorithms converge to different loops. Investigation of the cumulative reward of each loop, plotted in Fig. 8, shows that the reward gradients are similar despite the different flight paths. In this experiment, the iGP-SARSA(λ) algorithm converges to a solution after approximately 200 timesteps while the ε-greedy exploration strategy of GP-SARSA(λ) and SARSA(λ) do not discover and settle on an energy gaining flight loop until approximately 1700 and 3600 timesteps, respectively. Inspection of the complete flight logs shows that 156 state-actions were observed in the iGP-SARSA(λ) experiment, while 1616 and 3372 state-actions were visited in the GP-SARSA(λ) and SARSA(λ) experiments. This shows that directed informative exploration is much more effective than random exploration in higher dimensional problems, especially in problems where reward gain is critical, such as the soaring glider example where the platform must gain energy to stay aloft.

VI. Conclusions

The results presented in this paper show that the informative exploration component of the action selection in the proposed iGP-SARSA(λ) algorithm encourages effective exploration of the state-action space early on in the learning process to converge quickly to high reward gaining loops. iGP-SARSA(λ) was shown to outperform SARSA(λ) and GP-SARSA(λ) which explore via random actions. An extension to this can be to include a continuous and dynamic weighting factor between $\hat{Q}_a$ and $I_{a_{total}}$, such as that proposed in [15], which adaptively reflects the current desire for exploration or exploitation depending, for example, on the accumulated reward. This would have particular applications to tasks such as the soaring glider problem.

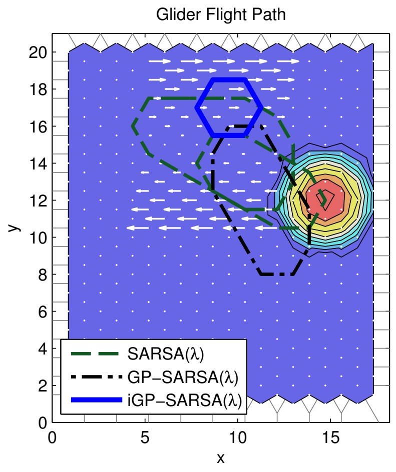
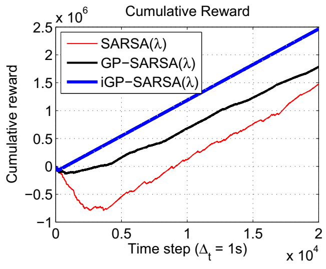
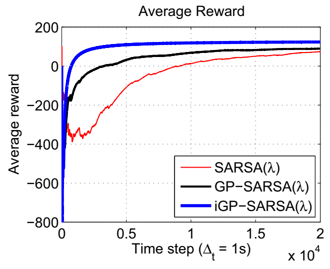


Fig. 7: The final flight plan loops learnt by the iGP-SARSA(λ) algorithm, shown by the thick blue line, GP-SARSA(λ), shown by the black broken dashed line, and the SARSA(λ) algorithm, shown by the dashed green line.



(a) Glider cumulative reward



(b) Glider average reward

Fig. 8: (a) The cumulative rewards for the iGP-SARSA(λ), GP-SARSA(λ) and SARSA(λ) simulations and (b) the average reward during the simulation for the three experiments.

References

[1] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. The MIT press, 1998.

[2] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *Proceedings of the National Conference on Artificial Intelligence*, 1998, pp. 761–768.

[3] Y. Engel, S. Mannor, and R. Meir, "Bayes meets Bellman: The Gaussian process approach to temporal difference learning," in *Proceedings of the 20th International Conference on Machine Learning*, vol. 20, no. 1, 2003, pp. 154–161.

[4] ——, "Reinforcement learning with Gaussian processes," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 201–208.

[5] C. Rasmussen and M. Kuss, "Gaussian processes in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 16, pp. 751–759, 2004.

[6] M. Deisenroth, C. Rasmussen, and J. Peters, "Gaussian process dynamic programming," *Neurocomputing*, vol. 72, no. 7-9, pp. 1508–1524, 2009.

[7] J. Wharington, "Autonomous control of a soaring aircraft by reinforcement learning," Ph.D. dissertation, Department of Aerospace Engineering, Royal Melbourne Institute of Technology, 1998.

[8] N. R. J. Lawrance, "Autonomous soaring flight for unmanned aerial vehicles," Ph.D. dissertation, The University of Sydney, 2011.

[9] S. Singh, A. G. Barto, and N. Chentanez, "Intrinsically motivated reinforcement learning," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 1281–1288.

[10] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg, "Intrinsically motivated reinforcement learning: An evolutionary perspective," *Autonomous Mental Development, IEEE Transactions on*, vol. 2, no. 2, pp. 70–82, 2010.

[11] K. Scheffler and S. Young, "Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning," in *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., 2002, pp. 12–19.

[12] M. English and P. Heeman, "Learning mixed initiative dialog strategies by using reinforcement learning on both conversants," in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2005, pp. 1011–1018.

[13] C. J. Wood, "The flight of albatrosses (A computer simulation)," *Ibis*, vol. 115, no. 2, pp. 244–256, 1972.

[14] H. Weimerskirch, T. Guionnet, J. Martin, S. A. Shaffer, and D. P. Costa, "Fast and fuel efficient? Optimal use of wind by flying albatrosses," *Proceedings of the Royal Society of London - Biological Sciences*, vol. 267, no. 1455, pp. 1869 – 1874, 2000.

[15] J. J. Chung, M. A. Trujillo, and S. Sukkarieh, "A new utility function for smooth transition between exploration and exploitation of a wind energy field," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems Conference Proceedings*, 2012, pp. 4999–5005.