Journal of Machine Learning Research 20 (2019) 1-73 Submitted 6/18; Revised 7/19; Published 8/19

Unsupervised Basis Function Adaptation for Reinforcement Learning

Edward Barker EBARKER@STUDENT.UNIMELB.EDU.AU
School of Mathematics and Statistics
University of Melbourne
Melbourne, Victoria 3010, Australia

Charl Ras CJRAS@UNIMELB.EDU.AU
School of Mathematics and Statistics
University of Melbourne
Melbourne, Victoria 3010, Australia

Editor: George Konidaris

Abstract

When using reinforcement learning (RL) algorithms it is common, given a large state space, to introduce some form of approximation architecture for the value function (VF). The exact form of this architecture can have a significant effect on an agent's performance, however, and determining a suitable approximation architecture can often be a highly complex task. Consequently there is currently interest among researchers in the potential for allowing RL algorithms to adaptively generate (i.e. to learn) approximation architectures. One relatively unexplored method of adapting approximation architectures involves using feedback regarding the frequency with which an agent has visited certain states to guide which areas of the state space to approximate with greater detail. In this article we will: (a) informally discuss the potential advantages offered by such methods; (b) introduce a new algorithm based on such methods which adapts a state aggregation approximation architecture online and is designed for use in conjunction with SARSA; (c) provide theoretical results, in a policy evaluation setting, regarding this particular algorithm's complexity, convergence properties and potential to reduce VF error; and finally (d) test experimentally the extent to which this algorithm can improve performance given a number of different test problems. Taken together our results suggest that our algorithm (and potentially such methods more generally) can provide a versatile and computationally lightweight means of significantly boosting RL performance given suitable conditions which are commonly encountered in practice.

Keywords: reinforcement learning, unsupervised learning, basis function adaptation, state aggregation, SARSA

1. Introduction

Traditional reinforcement learning (RL) algorithms such as TD(λ) (Sutton, 1988) or Q-learning (Watkins and Dayan, 1992) can generate optimal policies, when dealing with small state and action spaces, by exactly representing the value function (VF). However, when environments are complex (with large or continuous state or action spaces), using such algorithms directly becomes too computationally demanding. As a result it is common to introduce some form of architecture with which to approximate the VF, for example a parametrised set of functions (Sutton and Barto, 2018; Bertsekas and Tsitsiklis, 1996). One issue when introducing VF approximation, however, is that the accuracy of the algorithm's VF estimate, and as a consequence its performance, is highly dependent upon the exact form of the architecture chosen (it may be, for example, that no element of the chosen set of parametrised functions closely fits the VF). Accordingly, a number of authors have explored the possibility of allowing the approximation architecture to be *learned* by the agent, rather than pre-set manually by the designer—see Busoniu et al. (2010) for an overview. It is hoped that, by doing this, we can design algorithms which will perform well within a more general class of environment whilst requiring less explicit input from designers.[1]

A simple and perhaps, as yet, under-explored method of adapting an approximation architecture involves using an estimate of the frequency with which an agent has visited certain states to determine which states should have their values approximated in greater detail. We might be interested in such methods since, intuitively, we would suspect that areas which are visited more regularly are, for a number of reasons, more "important" in relation to determining a policy. Such a method can be contrasted with the more commonly explored method of explicitly measuring VF error and using this error as feedback to adapt an approximation architecture. We will refer to methods which adapt approximation architectures using visit frequency estimates as being *unsupervised* in the sense that no direct reference is made to reward or to any estimate of the VF.

Our intention in this article is to provide—in the setting of problems with large or continuous state spaces, where reward and transition functions are unknown, and where our task is to maximise reward—an exploration of unsupervised methods along with a discussion of their potential merits and drawbacks. We will do this primarily by introducing an algorithm, PASA, which represents an attempt to implement an unsupervised method in a manner which is as simple and obvious as possible. The algorithm will form the principle focus of our theoretical and experimental analysis.

It will turn out that unsupervised techniques have a number of advantages which may not be offered by other more commonly used methods of adapting approximation architectures. In particular, we will argue that unsupervised methods have (a) low computational overheads and (b) a tendency to require less sampling in order to converge. We will also argue that the methods can, under suitable conditions, (c) decrease VF error, in some cases significantly, with minimal input from the designer, and, as a consequence, (d) boost performance. The methods will be most effective in environments which satisfy certain conditions, however these conditions are likely to be satisfied by many of the environments we encounter most commonly in practice. We will also see that the principle of using state visit frequency can, perhaps counter-intuitively, interact favourably with the process of exploration.[2] The

1. Introducing the ability to adapt an approximation architecture is in some ways similar to simply adding additional parameters to an approximation architecture. However separating parameters into two sets, those adjusted by the underlying RL algorithm, and those adjusted by the adaptation method, permits us scope to, amongst other things, specify two distinct update rules.
2. Whilst policy changes will result in changes to visit frequency, in practice this co-dependency does not typically result in instability, but rather assists the agent to narrow in on strong policies. We explore the nature of this interaction more fully in Section 1.5 and Section 4 (in particular Section 4.4).

fact that unsupervised methods are cheap and simple, yet still have significant potential to enhance performance, makes them appear a promising, if perhaps somewhat overlooked, means of adapting approximation architectures.

1.1. Article Overview

Our article is structured as follows. Following some short introductory sections we will offer an informal discussion of the potential merits of unsupervised methods in order to motivate and give a rationale for our exploration (Section 1.5). We will then propose (in Section 2) our new algorithm, PASA, short for "Probabilistic Adaptive State Aggregation". The algorithm is designed to be used in conjunction with SARSA, and adapts a state aggregation approximation architecture on-line.

Section 3 is devoted to a theoretical analysis of the properties of PASA. Sections 3.1 to 3.3 relate to finite state spaces. We will demonstrate in Section 3.1 that PASA has a time complexity (considered as a function of the state and action space sizes, S and A) of the same order as its SARSA counterpart, i.e. $O(1)$. It has space complexity of $O(X)$ as a function of S, where X is the number of cells in the state aggregation architecture, and $O(1)$ as a function of A. This is compared to $O(X)$ and $O(A)$ respectively for its SARSA counterpart. This means that PASA is computationally cheap: it does not carry significant computational cost beyond SARSA with fixed state aggregation.

In Section 3.2 we investigate PASA in the context of where an agent's policy is held fixed and prove that the algorithm converges. This implies that, unlike non-linear architectures in general, SARSA combined with PASA will have the same convergence properties as SARSA with a fixed linear approximation architecture (i.e. the VF estimate may, assuming the policy is updated, "chatter", or fail to converge, but will never diverge).

In Section 3.3 we will use PASA's convergence properties to obtain a theorem, again where the policy is held fixed, regarding the impact PASA will have on VF error. This theorem guarantees that VF error will be arbitrarily low as measured by routinely used scoring functions provided certain conditions are met, conditions which require primarily that the agent spends a large amount of the time in a small subset of the state space. This result permits us to argue informally that PASA will also, assuming the policy is updated, improve performance given similar conditions.

In Section 3.4 we extend the finite state space concepts to continuous state spaces. We will demonstrate that, assuming we employ an initial arbitrarily complex discrete approximation of the agent's continuous input, all of our discrete case results have a straightforward continuous state space analogue, such that PASA can be used to reduce VF error (at low computational cost) in a manner substantially equivalent to the discrete case.

In Section 3.5 we outline some examples to help illustrate the types of environment in which the conditions relevant to Sections 3.3 and 3.4 are likely to be satisfied. We will see that, even for apparently highly unstructured environments where prior knowledge of the transition function is largely absent, the necessary conditions potentially exist to guarantee that employing PASA will result in low VF error. In a key example, we will show that for environments with large state spaces and where there is no prior knowledge of the transition function, PASA will permit SARSA to generate a VF estimate with error which is arbitrarily low with arbitrarily high probability provided the transition function and policy are sufficiently close to deterministic and the algorithm has $X = X(S) \geq f(S)$ cells available in its adaptive state aggregation architecture, where f is $O(\sqrt{S} \ln S \log_2 S)$.

To corroborate our theoretical analysis, and to further address the more complex question of whether PASA will improve overall performance, we outline some experimental results in Section 4. We explore three different types of environment: a GARNET environment,[3] a "Gridworld" type environment, and an environment representative of a logistics problem.

Our experimental results suggest that PASA, and potentially, by extension, techniques based on similar principles, can significantly boost performance when compared to SARSA with fixed state aggregation. The addition of PASA improved performance in all but one of our experiments,[4] and regularly doubled or even tripled the average reward obtained. Indeed, in some of the environments we tested, PASA was also able to outperform SARSA with *no* state abstraction, the potential reasons for which we discuss in Section 4. This is despite minimal input from the designer with respect to tailoring the algorithm to each distinct environment type.[5] Furthermore, in each case the additional processing time and resources required by PASA are measured and shown to be minimal, as predicted.

1.2. Related Works

The concept of using visit frequencies in an unsupervised manner is not completely new however it remains relatively unexplored compared to methods which seek to measure the error in the VF estimate explicitly and to then use this error as feedback. We are aware of only three papers in the literature which investigate a method similar in concept to the one that we propose, though the algorithms analysed in these three papers differ from PASA in some key respects.

Moreover there has been little by way of theoretical analysis of unsupervised techniques. The results we derive in relation to the PASA algorithm are all original, and we are not aware of any published theoretical analysis which is closely comparable.

In the first of the three papers just mentioned, Menache et al. (2005) provide a brief evaluation of an unsupervised algorithm which uses the frequency with which an agent has visited certain states to fit the centroid and scale parameters of a set of Gaussian basis functions. Their study was limited to an experimental analysis, and to the setting of policy evaluation. The unsupervised algorithm was not the main focus of their paper, but rather was used to provide a comparison with two more complex adaptation algorithms which used information regarding the VF as feedback.[6]

In the second paper, Nouri and Littman (2009) examined using a regression tree approximation architecture to approximate the VF for continuous multi-dimensional state spaces. Each node in the regression tree represents a unique and disjoint subset of the state space.

3. An environment with a discrete state space where the transition function is deterministic and generated uniformly at random. For more details refer to Sections 3.5 and 4.1.
4. The single exception was likely to have been affected by experimental noise. See Section 4.1.
5. For each problem, with the exception of the number of cells available to the state aggregation architecture, the PASA parameters were left unchanged
6. Their paper actually found the unsupervised method performed unfavourably compared to the alternative approaches they proposed. However they tested performance in only one type of environment, a type of environment which we will argue is not well suited to the methods we are discussing here (see Section 3.5).

Once a particular node has been visited a fixed number of times, the subset it represents is split ("once-and-for-all") along one of its dimensions, thereby creating two new tree nodes. The manner in which the VF estimate is updated[7] is such that incentive is given to the agent to visit areas of the state space which are relatively unexplored. The most important differences between their algorithm and ours are that, in their algorithm, (a) cell-splits are permanent, i.e. once new cells are created, they are never re-merged and (b) a permanent record is kept of each state visited (this helps the algorithm to calculate the number of times newly created cells have already been visited). With reference to (a), the capacity of PASA to re-adapt is, in practice, one of its critical elements (see Section 3). With reference to (b), the fact that PASA does not retain such information has important implications for its space complexity. The paper also provides theoretical results in relation to the optimality of their algorithm. Their guarantees apply in the limit of arbitrarily precise VF representation, and are restricted to model-based settings. In these and other aspects their analysis differs significantly from our own.

In the third paper, which is somewhat similar in approach and spirit to the second (and which also considers continuous state spaces), Bernstein and Shimkin (2010) examined an algorithm wherein a set of kernels are progressively split (again "once-and-for-all") based on the visit frequency for each kernel. Their algorithm also incorporates knowledge of uncertainty in the VF estimate, to encourage exploration. The same two differences to PASA (a) and (b) listed in the paragraph above also apply to this algorithm. Another key difference is that their algorithm maintains a distinct set of kernels for each action, which implies increased algorithm space complexity. The authors provide a theoretical analysis in which they establish a linear relationship between policy-mistake count[8] and maximum cell size in an approximation of the VF in a continuous state space.[9] The results they provide are akin to other PAC ("probably approximately correct") analyses undertaken by several authors under a range of varying assumptions—see, for example, Strehl et al. (2009) or, more recently, Jin et al. (2018). Their theoretical analysis differs from ours in many fundamental respects. Unlike our theoretical results in Section 3, their results have the advantage that they are not dependent upon characteristics of the environment and pertain to performance, not just VF error. However, similar to Nouri and Littman (2009) above, they carry the significant limitation that there is no guarantee of arbitrarily low policy-mistake count in the absence of an arbitrarily precise approximation architecture, which is equivalent in this context to arbitrarily large computational resources.[10]

There is a much larger body of work less directly related to this article, but which has central features in common, and is therefore worth mentioning briefly. Two important threads of research can be identified.

First, as noted above, a number of researchers have investigated the learning of an approximation architecture using feedback regarding the VF estimate. Early work in this area includes Singh et al. (1995), Moore and Atkeson (1995) and Reynolds (2000). Such approaches most commonly involve either (a) progressively adding features "once-and-for-

7. The paper proposes more than one algorithm. We refer here to the fitted Q-iteration algorithm.
8. Defined, in essence, as the number of time steps in which the algorithm executes a non-optimal policy.
9. See, in particular, their Theorems 4 and 5.
10. Our results, in contrast, provide guarantees relating to maximally reduced VF error under conditions where resources may be limited.

all"—see for example Munos and Moore (2002),[11] Keller et al. (2006) or Whiteson et al. (2007)—based on a criteria related to VF error, or (b) the progressive adjustment of a fixed set of basis functions using, most commonly, a form of gradient descent—see, for example, Yu and Bertsekas (2009), Di Castro and Mannor (2010) and Mahadevan et al. (2013).[12] Approaches which use VF feedback form an interesting array of alternatives for adaptively generating an approximation architecture, however such approaches can be considered as "taxonomically distinct" from the unsupervised methods we are investigating. The implications of using VF feedback compared to unsupervised adaptation, including some of the comparative advantages and disadvantages, are explored in more detail in our motivational discussion in Section 1.5. We will make the argument that unsupervised methods have certain advantages not available to techniques which use VF feedback in general.

Second, given that the PASA algorithm functions by updating a state aggregation architecture, it is worth noting that a number of principally theoretical works exist in relation to state aggregation methods. These works typically address the question of how states in a Markov decision process (MDP) can be aggregated, usually based on "closeness" of the transition and reward function for groups of states, such that the MDP can be solved efficiently. Examples of papers on this topic include Hutter (2016) and Abel et al. (2016) (the results of the former apply with generality beyond just MDPs). Notwithstanding being focussed on the question of how to create effective state aggregation approximation architectures, these works differ fundamentally from ours in terms of their assumptions and overall objective. Though there are exceptions—see, for example, Ortner (2013)[13]—the results typically assume knowledge of the MDP (i.e. the environment) whereas our work assumes no such knowledge. Moreover the techniques analysed often use the VF, or a VF estimate, to generate a state aggregation, which is contrary to the unsupervised nature of the approaches we are investigating.

It is worth, before concluding our discussion of related works, making brief mention of approaches which use gradient descent to optimise a fixed set of parameters of a complex and differentiable non-linear VF approximator.[14] Such approaches encompass (in part) what are known as deep RL methods, which have recently shown impressive results on an array of challenging problems (Mnih et al., 2015; Silver et al., 2018). When using this type of approach, additional techniques are often employed to encourage stability, since few convergence guarantees exist for non-linear VF approximation architectures in general. Our algorithm, PASA, is not incompatible with these types of approach fundamentally (the SARSA algorithm which PASA supports could, hypothetically, be replaced by a non-linearly parametrised RL algorithm). However the principles underlying unsupervised basis

11. The authors in this article investigate several distinct adaptation, or "splitting", criteria. However all depend in some way on an estimate of the value function.

12. Whilst less common, some approaches have been proposed, such as Bonarini et al. (2006), which fall somewhere in between (a) and (b). In the paper just cited the authors propose a method which involves employing a cell splitting rule offset by a cell merging (or pruning) rule.

13. This paper explores the possibility of aggregating states based on learned estimates of the transition and reward function, and as such the techniques it explores differ quite significantly from those we are investigating.

14. These approaches implicitly bear some similarity to basis function adaptation methods which use VF error as feedback.

function adaptation are very different from those underlying, for example, deep RL. The motivational discussion in Section 1.5 is helpful in drawing out the core differences between unsupervised methods and approaches, such as deep RL, which apply gradient descent in conjunction with a fixed, non-linear VF approximation architecture.

1.3. Formal Framework[15]

We assume that we have an agent which interacts with an environment over a sequence of iterations $t \in \mathbb{N}$. We will assume throughout this article (with the exception of Section 3.4) that we have a finite set $\mathcal{S}$ of *states* of size S (Section 3.4 relates to continuous state spaces and contains its own formal definitions where required). We also assume we have a discrete set $\mathcal{A}$ of *actions* of size A. Since S and A are finite, we can, using arbitrarily assigned indices, label each state s_i ($1 \leq i \leq S$) and each action a_j ($1 \leq j \leq A$).

For each t the agent will be in a particular state and will take a particular action. Each action is taken according to a *policy* π whereby the probability the agent takes action a_j in state s_i is denoted as $\pi(a_j|s_i)$.

The *transition function* $P\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ defines how the agent's state evolves over time. If the agent is in state s_i and takes an action a_j in iteration t, then the probability it will transition to the state $s_{i'}$ in iteration $t+1$ is given by $P(s_{i'}|s_i,a_j)$. The transition function must be constrained such that $\sum_{i'=1}^{S} P(s_{i'}|s_i,a_j) = 1$.

Denote as $\mathscr{R}$ the space of all probability distributions defined on the real line. The *reward function* $R\colon \mathcal{S} \times \mathcal{A} \to \mathscr{R}$ is a mapping from each state-action pair (s_i,a_j) to a real-valued random variable $R(s_i,a_j)$, where each $R(s_i,a_j)$ is defined by a cumulative distribution function $F_R(s_i,a_j) : \mathbb{R} \to [0,1]$, such that if the agent is in state s_i and takes action a_j in iteration t, then it will receive a real-valued *reward* in iteration t distributed according to $R(s_i,a_j)$. Some of our key results will require that $|R(s_i,a_j)| \leq c$, where c is a constant such that $0 \leq c < \infty$, for all i and j, in which case we use R_{m} to denote the maximum magnitude of the expected value of $R(s_i,a_j)$ over all i and j.

Prior to the point at which an agent begins interacting with an environment, both P and R are taken as being unknown. However we may assume in general that we are given a prior distribution for both. Our overarching objective is to design an algorithm to adjust π during the course of the agent's interaction with its environment so that total reward is maximised over some interval (for example, in the case of our experiments in Section 4, this will be a finite interval towards the end of each trial).

1.4. Scoring Functions

Whilst our overarching objective is to maximise performance, an important step towards achieving this objective involves reducing error in an algorithm's VF estimate. This is based on the assumption that more accurate VF estimates will lead to better directed policy updates, and therefore better performance. A large part of our theoretical analysis in Section 3 will be directed at assessing the extent to which VF error will be reduced under different circumstances.

15. The formal framework we assume in this article is a special case of a Markov decision process. For more general MDP definitions see, for example, Chapter 2 of Puterman (2014).

Error in a VF estimate for a fixed policy π is typically measured using a *scoring function.* It is possible to define many different types of scoring function, and in this section we will describe some of the most commonly used types.[16] We first need a definition of the VF itself. We formally define the *value function* Q^π_γ for a particular policy π, which maps each of the $S \times A$ state-action pairs to a real value, as follows:

$$Q^\pi_\gamma(s_i, a_j) \coloneqq \mathrm{E}\left(\sum_{t=1}^{\infty} \gamma^{t-1} R\big(s^{(t)}, a^{(t)}\big) \middle| s^{(1)} = s_i, a^{(1)} = a_j\right),$$

where the expectation is taken over the distributions of P, R and π (i.e. for particular instances of P and R, not over their prior distributions) and where $\gamma \in [0, 1)$ is known as a *discount factor*. We will generally omit the subscript γ. We have used superscript brackets to indicate dependency on the iteration t. Initially the VF is unknown.

Suppose that $\hat{Q}$ is an estimate of the VF. One commonly used scoring function is the squared error in the VF estimate for each state-action, weighted by some arbitrary function w which satisfies $0 \leq w(s_i, a_j) \leq 1$ for all i and j. We will refer to this as the *mean squared error* (MSE):

$$\mathrm{MSE}_\gamma \coloneqq \sum_{i=1}^{S} \sum_{j=1}^{A} w(s_i, a_j) \left(Q^\pi_\gamma(s_i, a_j) - \hat{Q}(s_i, a_j)\right)^2. \tag{1}$$

Note that the true VF Q^π_γ, which is unknown, appears in (1). Many approximation architecture adaptation algorithms use a scoring function as a form of feedback to help guide how the approximation architecture should be updated. In such cases it is important that the score is something which can be measured by the algorithm. In that spirit, another commonly used scoring function (which, unlike MSE, is not a function of Q^π_γ) uses T^π_γ, the *Bellman operator*, to obtain an approximation of the MSE. This scoring function we denote as L. It is a weighted sum of the *Bellman error* at each state-action:[17]

$$L_\gamma \coloneqq \sum_{i=1}^{S} \sum_{j=1}^{A} w(s_i, a_j) \left(T^\pi_\gamma \hat{Q}(s_i, a_j) - \hat{Q}(s_i, a_j)\right)^2,$$

where:

$$T^\pi_\gamma \hat{Q}(s_i, a_j) \coloneqq \mathrm{E}\left(R(s_i, a_j)\right) + \gamma \sum_{i'=1}^{S} P(s_{i'} | s_i, a_j) \sum_{j'=1}^{A} \pi(a_{j'} | s_{i'}) \hat{Q}(s_{i'}, a_{j'}).$$

The value L still relies on an expectation within the squared term, and hence there may still be challenges estimating L empirically. A third alternative scoring function $\tilde{L}$, which

16. Sutton and Barto (2018) provide a detailed discussion of different methods of scoring VF estimates. See, in particular, Chapters 9 and 11.
17. Note that this scoring function also depends on a discount factor γ, inherited from the Bellman error definition. It is effectively analogous to the constant γ used in the definition of MSE.

steps around this problem, can be defined as follows (F_R is defined in Section 1.3):

$$\tilde{L}_\gamma := \sum_{i=1}^{S}\sum_{j=1}^{A} w(s_i,a_j) \sum_{i'=1}^{S} P(s_{i'}|s_i,a_j) \sum_{j'=1}^{A} \pi(a_{j'}|s_{i'}) \\ \times \int_{\mathbb{R}} \Big(R(s_i,a_j) + \gamma\hat{Q}(s_{i'},a_{j'}) - \hat{Q}(s_i,a_j)\Big)^2 \, dF_R(s_i,a_j).$$

These three different scoring functions are arguably the most commonly used scoring functions, and we will state results in Section 3 in relation to all three. Scoring functions which involve a projection onto the space of possible VF estimates are also commonly used. We will not consider such scoring functions explicitly, however our results below will apply to these scoring functions, since, for the architectures we consider, scoring functions with and without a projection are equivalent.

We will need to consider some special cases of the weighting function w. Towards that end we define what we will term the *stable state probability* vector $\psi = \psi(\pi, s^{(1)})$, of dimension S, as follows:

$$\psi_i := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} I_{\{s^{(t)}=s_i\}},$$

where I is the indicator function for a logical statement such that $I_A = 1$ if A is true. The value of the vector ψ represents the proportion of the time the agent will spend in each state as $t \to \infty$ provided it follows the fixed policy π. In particular ψ_i indicates the proportion of the time the agent spends in the state s_i. As implied by its definition, ψ is a function of the policy π and the agent's starting state $s^{(1)}$ (i.e. its state at $t = 1$). In the case where a transition matrix obtained from π and P is irreducible and aperiodic, ψ will be the stationary distribution associated with π. None of the results in this paper relating to finite state spaces require that a transition matrix obtained from π and P be irreducible, however in order to avoid possible ambiguity, we will assume unless otherwise stated that ψ, whenever referred to, is the same for all $s^{(1)}$.

Perhaps the most natural, and also most commonly used, weighting coefficient $w(s_i, a_j)$ is $\psi(s_i)\pi(a_j|s_i)$, such that each error term is weighted in proportion to how frequently the particular state-action occurs (Menache et al., 2005; Yu and Bertsekas, 2009; Di Castro and Mannor, 2010). A slightly more general set of weightings is made up of those which satisfy $w(s_i, a_j) = \psi_i\tilde{w}(s_i, a_j)$, where $0 \leq \tilde{w}(s_i, a_j) \leq 1$ and $\sum_{j'=1}^{A} \tilde{w}(s_i, a_{j'}) \leq 1$ for all i and j. All of our theoretical results will require that $w(s_i, a_j) = \psi_i\tilde{w}(s_i, a_j)$, and some will also require that $\tilde{w}(s_i, a_j) = \pi(a_j|s_i)$.[18]

1.5. A Motivating Discussion

The principle we are exploring in this article is that frequently visited states should have their values approximated with greater precision. Why would we employ such a strategy? There is a natural intuition which says that states which the agent is visiting frequently are

18. It is worth noting that weighting by ψ and π is not necessarily the only valid choice for w. It would be possible, for example, to set $w(s_i, a_j) = 1$ for all i and j depending on the purpose for which the scoring function has been defined.

more important, either because they are intrinsically more prevalent in the environment, or because the agent is behaving in a way that makes them more prevalent, and should therefore be more accurately represented.

However it may be possible to pinpoint reasons related to efficient algorithm design which might make us particularly interested in such approaches. The thinking behind unsupervised approaches from this perspective can be summarised (informally) in the set of points which we now outline. Our arguments are based principally around the objective of minimising VF error (we will focus our arguments on MSE, though similar points could be made in relation to L or $\tilde{L}$). We will note at the end of this section, however, circumstances under which the arguments will potentially translate to benefits where policies are updated as well.

It will be critical to our arguments that the scoring function is weighted by ψ. Accordingly we begin by assuming that, in measuring VF error using MSE, we adopt $w(s_i, a_j) = \psi_i \tilde{w}(s_i, a_j)$, where $\tilde{w}(s_i, a_j)$ obeys the constraints noted in the previous subsection, is stored by the algorithm, and is not a function of the environment (for example, $\tilde{w}(s_i, a_j) = \pi(a_j|s_i)$ or $\tilde{w}(s_i, a_j) = 1/A$ for all i and j). Now consider:

1. Our goal is to find an architecture which will permit us to generate a VF estimate with low error. We can see, referring to equation (1), that we have a sum of terms of the form:
$$\psi_i \tilde{w}(s_i, a_j)\left(Q^{\pi}(s_i, a_j) - \hat{Q}(s_i, a_j)\right)^2 .$$
Suppose $\hat{Q}_{\text{MSE}}$ represents the value of $\hat{Q}$ for which MSE is minimised subject to the constraints of a particular architecture. Assuming we can obtain a VF estimate $\hat{Q} = \hat{Q}_{\text{MSE}}$ (e.g. using a standard RL algorithm), each term in equation (1) will be of the form:
$$\psi_i \tilde{w}(s_i, a_j)\left(Q^{\pi}(s_i, a_j) - \hat{Q}_{\text{MSE}}(s_i, a_j)\right)^2 .$$
In order to reduce MSE we will want to focus on ensuring that our architecture avoids the occurrence of large terms of this form. A term may be large either because ψ_i is large, because $\tilde{w}(s_i, a_j)$ is large, or because $Q^{\pi}(s_i, a_j) - \hat{Q}_{\text{MSE}}(s_i, a_j)$ has large magnitude. It is likely that any adaptation method we propose will involve directly or indirectly sampling one or more of these quantities in order to generate an estimate which can then be provided as feedback to update the architecture. Since $\tilde{w}(s_i, a_j)$ is assumed to be already stored by the algorithm, we focus our attention on the other two factors.

2. Whilst both ψ and $Q^{\pi} - \hat{Q}_{\text{MSE}}$ influence the size of each term, in a range of important circumstances generating an accurate estimate of ψ will be easier and cheaper than generating an accurate estimate of $Q^{\pi} - \hat{Q}_{\text{MSE}}$. We would argue this for three reasons:

   (a) An estimate of $Q^{\pi} - \hat{Q}_{\text{MSE}}$ can only be generated with accuracy once an accurate estimate of $\hat{Q}_{\text{MSE}}$ exists. The latter will typically be generated by the underlying RL algorithm, and may require a substantial amount of training time to generate, particularly if γ is close to one;[19]

19. Whilst the underlying RL algorithm will store an estimate $\hat{Q}$ of Q^{π}, having an estimate of Q^{π} is not the same as having an estimate of $Q^{\pi} - \hat{Q}$. If we want to estimate $Q^{\pi} - \hat{Q}$, we should consider it

(b) The value $Q^{\pi}(s_i, a_j) - \hat{Q}_{\text{MSE}}(s_i, a_j)$ may *also* depend on trajectories followed by the agent consisting of many states and actions (again particularly if γ is near one), and it may take many sample trajectories and therefore a long training time to obtain a good estimate, even once $\hat{Q}_{\text{MSE}}$ is known;

(c) For each single value ψ_i there are A terms containing distinct values for $Q^{\pi} - \hat{Q}_{\text{MSE}}$ in the MSE. This suggests that ψ can be more quickly estimated in cases where $\tilde{w}(s_i, a_j) > 0$ for more than one index j. Furthermore, the space required to store an estimate, if required, is reduced by a factor of A.

3. If we accept that it is easier and quicker to estimate ψ than $Q^{\pi} - \hat{Q}_{\text{MSE}}$, we need to ask whether measuring the former and not the latter will provide us with sufficient information in order to make helpful adjustments to the approximation architecture. If ψ_i is roughly the same value for all $1 \leq i \leq S$, then our approach may not work. However in practice there are many environments which (in some cases subject to the policy) are such that there will be a large amount of variance in the terms of ψ, with the implication that ψ can provide critical feedback with respect to reducing MSE. This will be illustrated most clearly through examples in Section 3.5.

4. Finally, from a practical, implementation-oriented perspective we note that, for fixed π, the value $Q^{\pi} - \hat{Q}_{\text{MSE}}$ is a function of the approximation architecture. This is not the case for ψ. If we determine our approximation architecture with reference to $Q^{\pi} - \hat{Q}_{\text{MSE}}$, we may find it more difficult to ensure our adaptation method converges.[20] This could force us, for example, to employ a form of gradient descent (thereby, amongst other things,[21] limiting us to architectures expressible via differential parameters, and forcing architecture changes to occur gradually) or to make "once-and-for-all" changes to the approximation architecture (removing any subsequent capacity for our architecture to adapt, which is critical if we expect, under more general settings, the policy π to change with time).[22]

To summarise, there is the possibility that in many important instances visit probability loses little compared to other metrics when assessing the importance of an area of the VF, and the simplicity of unsupervised methods allows for fast calculation and flexible implementation.

The above points focus on the problem of policy evaluation. All of our arguments will extend, however, to the policy *learning* setting, provided that our third point above consistently holds as each update is made. Whether this is the case will depend primarily on

in general as being estimated from scratch. The distinction is explored, for example, from a gradient descent perspective in Baird (1995). See also Chapter 11 in Sutton and Barto (2018).

20. This is because we are likely to adjust the approximation architecture so that the approximation architecture is capable of more precision for state-action pairs where $Q^{\pi}(s_i, a_j) - \hat{Q}_{\text{MSE}}(s_i, a_j)$ is large. But, in doing this, we will presumably remove precision from other state-action pairs, resulting in $Q^{\pi}(s_i, a_j) - \hat{Q}_{\text{MSE}}(s_i, a_j)$ increasing for these pairs, which could then result in us re-adjusting the architecture to give more precision to these pairs. This could create cyclical behaviour.

21. Gradient descent using the Bellman error is also known to be slow to converge and may require additional computational resources (Baird, 1995).

22. As we saw in Section 1.2, most methods which use VF feedback explored to date in the literature do indeed employ one of these two approaches.

the type of environment with which the agent is interacting. This will be explored further in Section 3.5 and Section 4.

Having now discussed, albeit informally, some of the potential advantages of unsupervised approaches to adapting approximation architectures, we would now like to implement the ideas in an algorithm. This will let us test the ideas theoretically and empirically in a more precise, rigorous setting.

2. The PASA Algorithm

Our Probabilistic Adaptive State Aggregation (PASA) algorithm is designed to work in conjunction with SARSA (though certainly there may be potential to use it alongside other, similar, RL algorithms). In effect PASA provides a means of allowing a state aggregation approximation architecture to be adapted on-line. In order to describe in detail how the algorithm functions it will be helpful to initially provide a brief review of SARSA, and introduce some terminology relating to state aggregation approximation architectures.

2.1. SARSA with Fixed State Aggregation

In its *tabular* form SARSA[23] stores an $S \times A$ array $\hat{Q}(s_i, a_j)$. It performs an update to this array in each iteration as follows:

$$\hat{Q}^{(t+1)}\big(s^{(t)}, a^{(t)}\big) = \hat{Q}^{(t)}\big(s^{(t)}, a^{(t)}\big) + \Delta\hat{Q}^{(t)}\big(s^{(t)}, a^{(t)}\big),$$

where:[24]

$$\Delta\hat{Q}^{(t)}\big(s^{(t)}, a^{(t)}\big) = \eta\left(R\big(s^{(t)}, a^{(t)}\big) + \gamma\hat{Q}^{(t)}\big(s^{(t+1)}, a^{(t+1)}\big) - \hat{Q}^{(t)}\big(s^{(t)}, a^{(t)}\big)\right) \tag{2}$$

and where η is a fixed step size parameter.[25] In the tabular case, SARSA has some well known and helpful convergence properties (Bertsekas and Tsitsiklis, 1996).

It is possible to use different types of approximation architecture in conjunction with SARSA. *Parametrised value function approximation* involves generating an approximation of the VF using a parametrised set of functions. The approximate VF is denoted as $\hat{Q}_\theta$, and, assuming we are approximating over the state space only and not the action space, this function is parametrised by a matrix of weights θ of dimension $X \times A$ (where, by assumption, $X \ll S$). Such an approximation architecture is *linear* if $\hat{Q}_\theta$ can be expressed in the form $\hat{Q}_\theta(s_i, a_j) = \varphi(s_i, a_j)^T\theta_j$, where θ_j is the jth column of θ and $\varphi(s_i, a_j)$ is a fixed vector of dimension X for each pair (s_i, a_j). The XA distinct vectors of dimension S given by $(\varphi(s_1, a_j)_k, \varphi(s_2, a_j)_k, \ldots, \varphi(s_S, a_j)_k)$ are called *basis functions* (where $1 \leq k \leq X$). It is common to assume that $\varphi(s_i, a_j) = \varphi(s_i)$ for all j, in which case we have only X distinct

23. The SARSA algorithm (short for "state-action-reward-state-action") was first proposed by Rummery and Niranjan (1994). It has a more general formulation SARSA(λ) which incorporates an eligibility trace. Any reference here to SARSA should be interpreted as a reference to SARSA(0).

24. Note that, in equation (2), γ is a parameter of the algorithm, distinct from γ as used in the scoring function definitions. However there exists a correspondence between the two parameters which will be made clearer below.

25. In the literature, η is generally permitted to change over time, i.e. $\eta = \eta^{(t)}$. However throughout this article we assume η is a fixed value.

basis functions, and $\hat{Q}_\theta(s_i, a_j) = \varphi(s_i)^T \theta_j$. If we assume that the approximation architecture being adapted is linear then the method of adapting an approximation architecture is known as *basis function adaptation*. Hence we refer to the adaptation of a linear approximation architecture using an unsupervised approach as *unsupervised basis function adaptation*.

Suppose that Ξ is a partition of $\mathcal{S}$, containing X elements, where we refer to each element as a *cell*. Indexing the cells using k, where $1 \leq k \leq X$, we will denote as $\mathcal{X}_k$ the set of states in the kth cell. A *state aggregation* approximation architecture—see, for example, Singh et al. (1995) and Whiteson et al. (2007)—is a simple linear parametrised approximation architecture which can be defined using any such partition Ξ. The parametrised VF approximation is expressed in the following form: $\hat{Q}_\theta(s_i, a_j) = \sum_{k=1}^{X} I_{\{s_i \in \mathcal{X}_k\}} \theta_{kj}$.

SARSA can be extended to operate in conjunction with a state aggregation approximation architecture if we update θ in each iteration as follows:[26]

$$\theta_{kj}^{(t+1)} = \theta_{kj}^{(t)} + \eta I_{\{s^{(t)} \in \mathcal{X}_k\}} I_{\{a^{(t)} = a_j\}} \left(R\big(s^{(t)}, a^{(t)}\big) + \gamma d^{(t)} - \theta_{kj}^{(t)} \right), \tag{3}$$

where:

$$d^{(t)} := \sum_{k'=1}^{X} \sum_{j'=1}^{A} I_{\{s^{(t+1)} \in \mathcal{X}_{k'}\}} I_{\{a^{(t+1)} = a_{j'}\}} \theta_{k'j'}^{(t)}. \tag{4}$$

We will say that a state aggregation architecture is *fixed* if Ξ (which in general can be a function of t) is the same for all t. For convenience we will refer to SARSA with fixed state aggregation as SARSA-F. We will assume (unless we explicitly state that π is held fixed) that SARSA updates its policy by adopting the ϵ-greedy policy at each iteration t.

Given a fixed state aggregation approximation architecture, if π is held fixed then the value $\hat{Q}_\theta$ generated by SARSA can be shown to converge—this can be shown, for example, using much more general results from the theory of stochastic approximation algorithms.[27] If, on the other hand, we allow π to be updated, then this convergence guarantee begins to erode. In particular, any policy update method based on periodically switching to an ϵ-greedy policy will not, in general, converge. However, whilst the values $\hat{Q}_\theta$ and π generated by SARSA with fixed state aggregation may oscillate, they will remain bounded (Gordon, 1996, 2001).

2.2. The Principles of How PASA Works

PASA is an attempt to implement the idea of unsupervised basis function adaptation in a manner which is as simple and obvious as possible without compromising computational efficiency. The underlying idea of the algorithm is to make the VF representation comparatively detailed for frequently visited regions of the state space whilst allowing the representation to be coarser over the remainder of the state space. It will do this by progressively

26. This algorithm is a special case of a more general variant of the SARSA algorithm, one which employs stochastic semi-gradient descent and which can be applied to any set of linear basis functions.

27. This is examined more formally in Section 3. Note that the same is true for SARSA when used in conjunction with any linear approximation architecture. Approximation architectures which are *non-linear*, by way of contrast, cannot be guaranteed to converge even when a policy is held fixed, and may in fact diverge. Often the employment of a non-linear architecture will demand additional measures be taken to ensure stability—see, for example, Mnih et al. (2015). Given that the underlying approximation architecture is linear, unsupervised basis function adaptation methods typically do not require any such additional measures.

updating $\Xi = \Xi^{(t)}$. Whilst the partition Ξ progressively changes it will always contain a fixed number of cells X, and every cell will always be non-empty. We will refer to SARSA combined with PASA as SARSA-P (to distinguish it from SARSA-F described above).

The algorithm is set out in Algorithms 1 and 2. Before describing the precise details of the algorithm, however, we will attempt to describe informally how it works. PASA begins with an initial partition $\Xi^{(1)}$. This initial partition is, to some extent, arbitrary. PASA will update Ξ only infrequently. We must choose the value of a parameter $\nu \in \mathbb{N}$, which in practice will be large (for our experiments in Section 4 we choose $\nu = 50{,}000$). In each iteration t such that $t \bmod \nu = 0$, PASA will update Ξ, otherwise Ξ remains fixed. As will become clearer below, the reason for updating Ξ infrequently is to allow certain weight vectors, which are used to update Ξ, to be progressively adapted in between updates to Ξ. Every time Ξ is updated (which involves a sequence of steps *performed in a single iteration*) it describes a new, complete partition of the state space with X cells.

PASA updates Ξ, at the regular intervals defined by ν, as follows. Each time it in effect completely rebuilds the partition Ξ using a much less granular fixed partition Ξ_0 as a starting point. Specifically, for each update it begins with a fixed set of X_0 *base cells*, with $X_0 < X$, which together form a partition Ξ_0 of $\mathcal{S}$ (the partition Ξ_0 is identical for every periodic update). Suppose we have an estimate of how frequently the agent visits each of these base cells based on its recent behaviour. We can define a new partition Ξ_1 by "splitting" the most frequently visited cell into two cells containing a roughly equal number of states (the notion of a cell "split" is described more precisely below). If we now have a similar visit frequency estimate for each of the cells in the newly created partition, we could again split the most frequently visited cell giving us yet another partition Ξ_2. If we repeat this process a total of $X - X_0$ times (which PASA is designed to do all in the space of a single iteration, whenever $t \bmod \nu = 0$), then we will have generated a partition Ξ of the state space with X cells. Moreover, provided our visit frequency estimates are accurate, those areas of the state space which are visited more frequently will have a more detailed representation of the VF. (To be clear: Ξ_1, for example, denotes the partition generated after the first step of the process just described, whereas $\Xi^{(1)}$, for example, denotes the actual partition used by the SARSA algorithm at $t = 1$.)

For this process to work effectively, PASA needs to have access to an accurate estimate of the visit frequency for each cell for each stage of the splitting process. We could, at a first glance, provide this by storing an estimate of the visit frequency of every individual state. We could then estimate cell visit frequencies by summing the estimates for individual states as required. However S is, by assumption, very large, and storing S distinct real values is implicitly difficult or impossible. Accordingly, PASA instead stores an estimate of the visit frequency of each base cell, and an estimate of the visit frequency of *one* of the two cells defined each time a cell is split. This allows PASA to calculate an estimate of the visit frequency of every cell in every stage of the process described in the paragraph above whilst storing only X distinct values. It does this by subtracting certain estimates from others (a process described in more detail below). In this way we can estimate cell visit frequencies efficiently, at the cost of only a small trade off which we describe in Section 2.4 below.

2.3. Some Additional Terminology Relating to State Aggregation

In this subsection we will introduce some formal concepts, including the concept of "splitting" a cell, which will allow us, in the next subsection, to formally describe the PASA algorithm.

Our formalism is such that S is finite.[28] This means that, for any problem, we can arbitrarily index each state from 1 to S. Suppose we have a partition $\Xi_0 = \{\mathcal{X}_{j,0} : 1 \leq j \leq X_0\}$ defined on $\mathcal{S}$ with X_0 elements. We will say that the partition Ξ_0 is *ordered* if every cell $\mathcal{X}_{j,0}$ can be expressed as an interval of the form:

$$\mathcal{X}_{j,0} := \{s_i : L_{j,0} \leq i \leq U_{j,0}\},$$

where $L_{j,0}$ and $U_{j,0}$ are integers and $1 \leq L_{j,0} \leq U_{j,0} \leq S$. Starting with an ordered partition Ξ_0, we can formalise the notion of *splitting* one of its cells $\mathcal{X}_{j,0}$, via which we can create a new partition Ξ_1. The new partition $\Xi_1 = \{\mathcal{X}_{j',1} : 1 \leq j' \leq X_1\}$ will be such that:

$$\begin{aligned}
&X_1 = X_0 + 1\\
&\mathcal{X}_{j,1} = \{s_i : L_{j,0} \leq i \leq L_{j,0} + \lfloor (U_{j,0} - L_{j,0} - 1)/2 \rfloor\}\\
&\mathcal{X}_{X_0+1,1} = \{s_i : L_{j,0} + \lfloor (U_{j,0} - L_{j,0} - 1)/2 \rfloor < i \leq U_{j,0}\}\\
&\mathcal{X}_{j',1} = \mathcal{X}_{j',0} \text{ for all } j' \in \{1, \ldots, j-1\} \cup \{j+1, \ldots, X_0\}
\end{aligned}$$

The effect is that we are splitting the interval associated with $\mathcal{X}_{j,0}$ as near to the "middle" of the cell as possible. This creates two new intervals, the "lower" interval replaces the existing cell, and the "upper" interval becomes a new cell (with index $X_0 + 1$). The new partition Ξ_1 is also an ordered partition. Note that the splitting procedure is only defined for cells with cardinality of two or more. For the remainder of this subsection our discussion will assume that this condition holds every time a particular cell is split. When we apply the procedure in practice we will take measures to ensure that, whenever a split occurs, this condition is always satisfied.

Starting with any initial ordered partition, we can recursively reapply this splitting procedure as many times as we like. Note that each time a split occurs, we specify the index of the cell we are splitting. This means, given an initial ordered partition Ξ_0 (with X_0 cells), we can specify a final partition Ξ_n (with $X_n = X_0 + n$ cells) by providing a vector ρ of integers, or *split vector*, which is of dimension n and is a list of the indices of cells to split. The split vector ρ must be such that, for each $1 \leq k \leq n$, the constraint $1 \leq \rho_k \leq X_0 + k - 1$ is satisfied (so that each element of ρ refers to a valid cell index). Assuming we want a partition composed of X cells exactly (i.e. so that $X_n = X$), then ρ must be of dimension $X - X_0$. Parts (a) and (b) of Figure 1 provide a partial illustration of how a pair of values ρ and Ξ_0 can be used to define a new, more granular partition.

Before proceeding we require one more definition. For each partition Ξ_k defined above, where $0 \leq k \leq n$, we introduce a collection of subsets of $\mathcal{S}$ denoted $\bar{\Xi}_k = \{\bar{\mathcal{X}}_j : 1 \leq j \leq X_0 + k\}$. Each element of $\bar{\Xi}_k$ is defined as follows:

$$\bar{\mathcal{X}}_j := \begin{cases} \{s_i : s_i \in \mathcal{X}_{j,0}\} & \text{if } 1 \leq j \leq X_0 \\ \{s_i : s_i \in \mathcal{X}_{j,j-X_0}\} & \text{if } X_0 < j \leq X \end{cases}$$

28. In the case of continuous state spaces we assume that we have a finite set of "atomic cells" which are analogous to the finite set of states discussed here. See Section 3.4.

Each set $\bar{\mathcal{X}}_j$ is not a function of k. However, for $j > X_0$ the value of $\bar{\mathcal{X}}_j$ will only be available after $j - X_0$ steps in the sequence described above. The effect of the definition is that, for $0 \leq j \leq X_0$, we simply have $\bar{\mathcal{X}}_j = \mathcal{X}_{j,0}$ for all j, whilst for $X_0 < j \leq X$, $\bar{\mathcal{X}}_j$ will contain all of the states which are contained in $\bar{\mathcal{X}}_{j,j-X_0}$, which is the first cell created during the overall splitting process which had an index of j, as it was before any additional splitting of that cell. (In other words, all the base cells can be referred to by $\bar{\mathcal{X}}_j$ for $1 \leq j \leq X_0$, and, for $j > X_0$, each time a cell $\mathcal{X}_{i,j-X_0-1}$ is split, the newly created cell $\mathcal{X}_{j,j-X_0}$ is equal to $\bar{\mathcal{X}}_j$, while $\bar{\mathcal{X}}_i$ remains unchanged and still retains its value from the first time it was created.) Note that $\bar{\Xi}_k$ is *not* a partition, with the single exception of $\bar{\Xi}_0$ which is equal to Ξ_0. The notation just outlined will be important when we set out the manner in which PASA estimates the frequency with which different cells are visited.

2.4. Details of the Algorithm

We now add the necessary final details to formally define the PASA algorithm. We assume we have a fixed ordered partition Ξ_0 containing X_0 cells. The manner in which Ξ_0 is constructed does not need to be prescribed as part of the PASA algorithm, however we assume $|\mathcal{X}_{j,0}| \geq 1$ for all $1 \leq j \leq X_0$. In general, therefore, Ξ_0 is a parameter of PASA.[29] PASA stores a split vector ρ of dimension $X - X_0$. This vector in combination with Ξ_0 defines a partition Ξ, which will represent the state aggregation architecture used by the underlying SARSA algorithm. Recall that we used $\mathcal{X}_j$ to denote a cell in a state aggregation architecture in Section 2.1. In the context of SARSA-P, $\mathcal{X}_j$ will be a function of t, and we will use the natural convention that $\mathcal{X}_j = \mathcal{X}_{j,X-X_0}$. We also adopt the notation $\bar{\Xi} \coloneqq \bar{\Xi}_{X-X_0}$.

The vector ρ, and correspondingly the partition Ξ, will be updated every $\nu \in \mathbb{N}$ iterations, where ν (as noted above) is a fixed parameter. The interval defined by ν permits PASA to learn visit frequency estimates, which will be used when updating ρ. Subject to the constraint noted in the previous subsection, ρ can be initialised arbitrarily, however we assume that each ρ_k for $1 \leq k \leq X - X_0$ is initialised so that no attempt will be made to split a cell containing only one state (a *singleton* cell).

To assist in updating ρ, the algorithm will store a vector $\bar{u}$ of real values of dimension X (initialised as a vector of zeroes). We update $\bar{u}$ in each iteration as follows (i.e. using a simple stochastic approximation algorithm):

$$\bar{u}_j^{(t+1)} = \bar{u}_j^{(t)} + \varsigma \left(I_{\{s^{(t)} \in \bar{\mathcal{X}}_j\}} - \bar{u}_j^{(t)} \right), \tag{5}$$

where $\varsigma \in (0, 1]$ is a constant step size parameter. In this way, $\bar{u}$ will record the approximate frequency with which each of the sets in $\bar{\Xi}$ have been visited by the agent.[30] We also store an X dimensional boolean vector Σ. As will be made clearer below, Σ keeps track of whether a particular cell has only one state, as we don't want the algorithm to try to split singleton cells.

29. The reason we do not simply take $X_0 = 1$ is that taking $X_0 > 1$ can help to ensure that the values stored by PASA tend to remain more stable. In practice, it often makes sense to choose a suitable value for X_0, then simply take Ξ_0 to be the ordered partition consisting of X_0 cells which are as close as possible to equal size. See Section 4.

30. Hence, when estimating how frequently the agent visits certain sets of states, the PASA algorithm implicitly weights recent visits more heavily using a series of coefficients which decay geometrically. The rate of this decay depends on ς.

To update ρ the PASA algorithm, whenever $t \bmod \nu = 0$, performs a sequence of $X - X_0$ operations. A temporary copy of $\bar{u}$ is made, which we call u. The vector u is intended to estimate the approximate frequency with which each of the cells in Ξ have been visited by the agent. The elements of u will be updated as part of the sequence of operations which we will presently describe. We set the entries of Σ to $I_{\{|\mathcal{X}_{k,0}|=1\}}$ for $1 \leq k \leq X_0$ at the start of the sequence (the remaining entries can be set to zero). At each stage $k \in \{1, 2, \ldots, X - X_0\}$ of the sequence we update ρ as follows:

$$\rho_k = \begin{cases} j & \text{if } (1 - \Sigma_{\rho_k})u_{\rho_k} < \max\{u_i : i \leq X_0 + k - 1, \Sigma_i = 0\} - \vartheta \\ \rho_k & \text{otherwise} \end{cases} \tag{6}$$

where:

$$j = \arg\max_i \{u_i : i \leq X_0 + k - 1, \Sigma_i = 0\}$$

(if multiple indices satisfy the arg max function, we take the lowest index) and where $\vartheta > 0$ is a constant designed to ensure that a (typically small) threshold must be exceeded before ρ is adjusted. In this way, in each step k in the sequence the non-singleton cell $\mathcal{X}_{j,k-1}$ with the highest value u_j (over the range $1 \leq j \leq k-1$, and subject to the threshold ϑ) will be identified, via the update to ρ, as the next cell to split. In each step of the sequence we also update u and Σ:

$$\begin{aligned} u_{\rho_k} &= u_{\rho_k} - u_{X_0+k} \\ \Sigma_j &= I_{\{|\mathcal{X}_{j,k}| \leq 1\}} \text{ for } 1 \leq j \leq X_0 + k - 1. \end{aligned}$$

The reason we update u as shown above is because each time the operation is applied we thereby obtain an estimate of the visit frequency of $\mathcal{X}_{\rho_k,k}$, which is the freshly updated value of u_{ρ_k}, and an estimate of the visit frequency of the cell $\mathcal{X}_{X_0+k,k}$, which is $u_{X_0+k} = \bar{u}_{X_0+k}$ (since $u_{X_0+k} = \bar{u}_{X_0+k}$ at step k). We note that, for each of the cell visit frequency estimates u_{ρ_k} and u_{X_0+k} to be accurate, it is critical that both the original estimates u_{ρ_k} and u_{X_0+k} are accurate. Crucial to the operation of the algorithm is the fact that this dependence only flows in one direction. As estimates for larger cells tend to become more accurate, estimates for smaller cells which are a function of the estimates for larger cells also become more accurate. As a result we can depend upon accurate estimates for larger cells flowing through to accurate estimates for smaller cells. We might ask why we do not, for example, estimate the visit frequency of a newly created cell by simply dividing the visit frequency for the parent cell in two. This makes sense the first time a split is generated (and indeed doing this the first time a cell is split represents a sensible extension of the algorithm, see Section A.5), however, adopting the method we have described permits us to obtain estimates which will be exact in the limit as the algorithm is given time to converge.

Once ρ has been generated, we implicitly have a new partition Ξ. The PASA algorithm is outlined in Algorithm 1 and a diagram illustrating the main steps is at Figure 1. Note that the algorithm calls a procedure called Split, which is outlined in Algorithm 2.[31] Algorithm

31. Note that the pseudo code describes the underlying principles but is not an efficient implementation. The step to determine cell end points, for example, can be implemented in a far more efficient way than explicitly calculating a minimum or maximum. A full description of such details is beyond our present scope. See Section 3.1 for related details.

1 operates such that the cell splitting process (to generate Ξ) occurs concurrently with the update to ρ, such that, as each element ρ_k of ρ is updated, a corresponding temporary partition Ξ_k is constructed. Also note that the algorithm makes reference to objects Ξ' and $\bar{\Xi}'$. To avoid storing each Ξ_k and $\bar{\Xi}_k$ for $1 \leq k \leq X - X_0$, we instead initialise Ξ' and $\bar{\Xi}'$ as Ξ_0 and then recursively update Ξ' and $\bar{\Xi}'$ such that $\Xi' = \Xi_k$ and $\bar{\Xi}' = \bar{\Xi}_k$ at the kth stage of the splitting process.

In Section 2.2 we noted that estimating the visit probability of individual cells by subtracting estimates from one another (as PASA is designed to do) allows us to avoid storing visit probabilities for S individual states. There is a trade-off involved when estimating visit frequencies in such a way. Suppose that $t = n\nu$ for some $n \in \mathbb{N}$ and the partition $\Xi^{(t)}$ is updated and replaced by the partition $\Xi^{(t+1)}$. The visit frequency estimate for a cell in $\Xi^{(t+1)}$ is only likely to be accurate if the same cell was an element of $\Xi^{(t)}$, or if the cell is a union of cells which were elements of $\Xi^{(t)}$. Cells in $\Xi^{(t+1)}$ which do not fall into one of these categories will need time for an accurate estimate of visit frequency to be obtained (this will be shown more clearly in Section 3.2). The consequence is that it may take longer for the algorithm to converge (assuming fixed π) than would be the case if an estimate of the visit frequency of every state were available. The negative impact of this trade-off in practice does not appear to be significant (see Section 4), however the trade-off is essential to ensure we have an efficient algorithm from the perspective of space complexity.

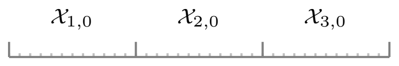
3. Theoretical Analysis

Having proposed the PASA algorithm we will now investigate some of its theoretical properties. With the exception of our discussion around complexity, these results will be constrained to problems of policy evaluation, where π is held fixed and the objective is to minimise VF error. A summary of all of the theoretical results can be found in Table 1.

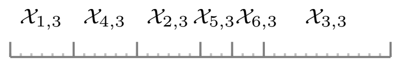
3.1. Complexity of PASA

PASA requires only a very modest increase in computational resources compared to fixed state aggregation. In relation to time complexity, $\bar{u}$ can be updated in parallel with the SARSA algorithm's update of θ (and the update of $\bar{u}$ would not be expected to have any greater time complexity than the update to θ by SARSA, or indeed another standard RL algorithm such as Q-learning). Indeed, once a state has been mapped to a cell, the SARSA and PASA operations each have $O(1)$ time complexity in relation to S.

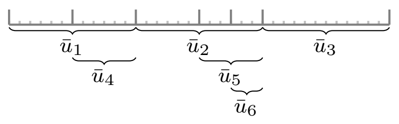
Assuming that we implement the mapping generated by Ξ using a tree-like structure—for details of such an approach, which can be applied equally to continuous and discrete state spaces, see, for example, Nouri and Littman (2009) or Munos and Moore (2002)—then the mapping from state to cell has, for discrete state spaces, a very low order of time complexity: $O(\log_2 S)$. In general the mapping from state to cell for a fixed partition will be of the same complexity, the only exception being when cells in the fixed partition are guaranteed to be above a certain size—for example we would have a minimum of $O(\log_2 X)$ for X fixed, equally-sized cells. The split vector ρ can be updated at intervals ν (and this update can also be run in parallel). In practice ν can be large because this allows time for $\bar{u}$ to converge. Hence, for all practical purposes, the increase in time complexity per iteration involved in the introduction of PASA is negligible.



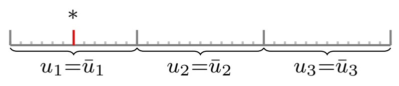
(a) We will have $X_0 = 3$ “base” cells in an ordered partition Ξ_0. We will use an arbitrarily chosen initial vector ρ to split these cells and obtain $X = 6$ cells.



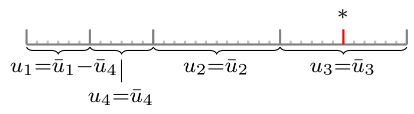
(b) Suppose ρ is initialised (before the first iteration) as $\rho = (1, 2, 5)$. This defines a sequence of splits to arrive at X cells depicted above. This partition represents the initial value of Ξ, $\Xi^{(1)}$.



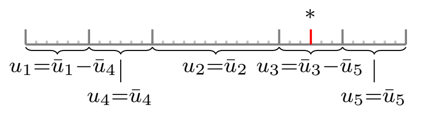
(c) Over the interval $t \in (1, \ldots, \nu)$, update the vector $\bar{u}$, an estimate of visit probabilities, then at iteration ν make a copy of $\bar{u}$, u.



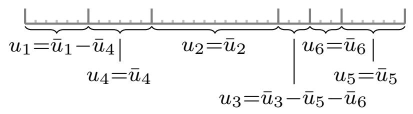
(d) At iteration ν we also generate a new value for ρ and Ξ. Start by splitting the cell with the highest value of u_i $(1 \leq i \leq 3)$. Assume this is u_1. The split, shown in red and with an asterisk, replaces a cell containing 12 states with two cells containing 6 states.



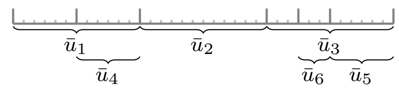
(e) Recalculate u and then split the cell with the next highest value of u_i (for $1 \leq i \leq 4$). Assume this is u_3.



(f) Repeat (for $1 \leq i \leq 5$). In general this step will be repeated $X - X_0$ times.


$u_1 = \bar{u}_1 - \bar{u}_4$
$u_2 = \bar{u}_2$
$u_6 = \bar{u}_6$
$u_4 = \bar{u}_4$
$u_5 = \bar{u}_5$
$u_3 = \bar{u}_3 - \bar{u}_5 - \bar{u}_6$

(g) This defines a new split vector ρ and partition Ξ. In this case $\rho = (1, 3, 3)$.



(h) Generate a new estimate $\bar{u}$ over the interval $t \in (\nu{+}1, \ldots, 2\nu)$, and continue the process indefinitely.

Figure 1: A simple example of the operation of the PASA algorithm, with $X = 6$, $X_0 = 3$ and $S = 36$. The horizontal line represents the set of all states, arranged in some arbitrary order, with the space between each short light grey tick representing a state (s_1 is the left-most state, with states arranged in order of their index up to s_{36}, the right-most state). The slightly longer vertical ticks represent the way in which states are distributed amongst cells: the space between two such longer ticks represents all of the states in a particular cell.

Algorithm 1 The PASA algorithm, called at each iteration t. We assume $\bar{u}$, ρ, Ξ_0, Ξ' and $\bar{\Xi}'$ are stored in memory. By definition $\bar{\Xi}_0 = \Xi_0$. The value of the partition Ξ' at the conclusion of the function call will be used at $t+1$ to determine the state aggregation approximation architecture employed by SARSA. The values ς, ϑ, and ν are constant parameters. Initialise c (a counter) at zero. Initialise each element of the vector $\bar{u}$ at zero. Denote $s = s^{(t)}$. Return is void.

```
1: function PASA($s$)
2:     for $k \in \{1, \dots, X\}$ do
3:         $\bar{u}_k \leftarrow \bar{u}_k + \varsigma(I_{\{s \in \bar{\mathcal{X}}_k\}} - \bar{u}_k)$        ▷ Update estimates of visit frequencies
4:     end for
5:     $c \leftarrow c + 1$
6:     if $c = \nu$ then        ▷ Periodic updates to $\rho$ and $\Xi$
7:         $c \leftarrow 0$
8:         $u \leftarrow \bar{u}$
9:         $\Xi' \leftarrow \Xi_0$
10:        $\bar{\Xi}' \leftarrow \Xi_0$
11:        for $k \in \{1, \dots, X_0\}$ do
12:            $\Sigma_k \leftarrow I_{\{|\mathcal{X}_{k,0}|=1\}}$        ▷ Reset flags used to identify singular cells
13:        end for
14:        ▷ Iterate through sequence of cell splits
15:        for $k \in \{1, \dots, X - X_0\}$ do
16:            ▷ Identify the cell with the highest visit probability estimate
17:            $u_{\max} \leftarrow \max\{u_i : i \leq X_0 + k - 1, \Sigma_i = 0\}$
18:            $i_{\max} \leftarrow \min\{i : i \leq X_0 + k - 1, u_i = u_{\max}, \Sigma_i = 0\}$
19:            ▷ If threshold $\vartheta$ is exceeded, update partition
20:            if $(1 - \Sigma_{\rho_k}) u_{\rho_k} < u_{i_{\max}} - \vartheta$ then
21:                $\rho_k \leftarrow i_{\max}$        ▷ Reassign value for $\rho$
22:            end if
23:            $u_{\rho_k} \leftarrow u_{\rho_k} - u_{X_0+k}$        ▷ Update value of $u$
24:            $(\Xi', \bar{\Xi}', \Sigma) \leftarrow$ Split$(k, \rho_k, \Xi', \bar{\Xi}', \Sigma)$        ▷ Call function to split cell
25:        end for
26:    end if
27: end function
```

PASA does involve additional space complexity with respect to storing the vector $\bar{u}$: we must store X real values, such that PASA will have $O(X)$ space complexity when considered as a function of S. The values of Ξ and $\bar{\Xi}$ must also be stored, however Ξ must be stored for fixed state aggregation as well and, again when stored as a tree-like structure, each of Ξ and $\bar{\Xi}$ will only have $O(X)$ space complexity. Clearly PASA has space complexity of $O(1)$ when considered as a function of A (the number of actions available has no impact on the complexity or design of PASA). The SARSA component has space complexity of $O(X)$ when considered as a function of S, and space complexity of $O(A)$ when considered as a function of A (this reflects the $X \times A$ cell-action pairs). The implication is that the

Algorithm 2 Function to split selected cell in step k of sequence of cell splits. Called by PASA. The values Ξ'' and $\bar{\Xi}''$ are uninitialised and have $X_0 + k$ elements each. Use $\mathcal{X}'_j$, $\mathcal{X}''_j$, $\bar{\mathcal{X}}'_j$ and $\bar{\mathcal{X}}''_j$ to denote the jth element of Ξ', Ξ'', $\bar{\Xi}'$ and $\bar{\Xi}''$ respectively.

```
1: function Split($k$,$\rho_k$,$\Xi'$,$\bar{\Xi}'$,$\Sigma$)
2:     ▷ Set all but last element of $\Xi''$ and $\bar{\Xi}''$
3:     for $j \in \{1, \ldots, X_0 + k - 1\}$ do
4:         $\mathcal{X}''_j \leftarrow \mathcal{X}'_j$
5:         $\bar{\mathcal{X}}''_j \leftarrow \bar{\mathcal{X}}'_j$
6:     end for
7:     $L \leftarrow \min\{i : s_i \in \mathcal{X}_{\rho_k}\}$                ▷ Determine cell end points
8:     $U \leftarrow \max\{i : s_i \in \mathcal{X}_{\rho_k}\}$
9:     ▷ Update $\Xi''$
10:    $\mathcal{X}''_{\rho_k} \leftarrow \{s_i : L \le i \le L + \lfloor (U - L - 1)/2 \rfloor\}$
11:    $\mathcal{X}''_{X_0+k} \leftarrow \{s_i : L + \lfloor (U - L - 1)/2 \rfloor < i \le U\}$
12:    ▷ Set last element of $\bar{\Xi}''$
13:    $\bar{\mathcal{X}}''_{X_0+k} \leftarrow \{s_j : L + \lfloor (U - L - 1)/2 \rfloor < i \le U\}$     ▷ $\bar{\mathcal{X}}''_{\rho_k}$ does not change
14:    ▷ Identify singular cells (only need to update values for newly split cells)
15:    $\Sigma_{\rho_k} = I_{\{|\mathcal{X}''_{\rho_k}|=1\}}$
16:    $\Sigma_k = I_{\{|\mathcal{X}''_{X_0+k}|=1\}}$
17:    return $(\Xi'', \bar{\Xi}'', \Sigma)$                ▷ Return new partitions
18: end function
```

introduction of PASA as a pre-processing algorithm will not materially impact overall space requirements (in particular if A is large).

3.2. Convergence Properties of PASA

We would now like to consider some of the convergence properties of PASA. We will assume, for our next result, that π is held fixed for all t. There may be some potential to reformulate the result without relying on the assumption of fixed π, however our principle interest will be in this special case.

Our outline of PASA assumed a single fixed step size parameter ς. For our proof below it will be easier to suppose that we have a distinct fixed step size parameter ς_k for each element $\bar{u}_k$ of $\bar{u}$ ($1 \le k \le X$), each of which we can set to a different value (fixed as a function of t). For the remainder of this section ς should be understood as referring to this vector of step size parameters. We use $[1{:}k]$ to denote the set of indices from 1 to k, so that, for example, we can use $x_{[1:k]}$ to indicate a vector comprised of the first k elements of an arbitrary vector x. We will require some definitions.

Definition 1 *We will say that some function of t, $x = x^{(t)}$,* becomes ε-fixed over τ after T *provided T is such that, for all $T' > T$, the value x will remain the same for all t' satisfying $T' \le t' \le T' + \tau$ with probability at least $1 - \varepsilon$. We will similarly say, given a set of values $\mathcal{R}$, that x* becomes ε-fixed at $\mathcal{R}$ over τ after T *provided T is such that, for all $T' > T$, the* *value x is equal to a single element of $\mathcal{R}$ for all t' satisfying $T' \leq t' \leq T' + \tau$ with probability at least $1 - \varepsilon$.*

Definition 1 provides the sense in which we will prove the convergence of PASA in Proposition 4. It is weaker than most conventional definitions of convergence, however it will give us conditions under which we can ensure that ρ will remain stable (i.e. unchanging) for arbitrarily long intervals with arbitrarily high probability. This stability will allow us to call on well established results relating to the convergence of SARSA with fixed state aggregation. Such a convergence definition also permits a fixed step size parameter ς. This means, amongst other things, that the algorithm will "re-converge" if P or (more importantly) π change.

Definition 2 *We define $\mu_{j,k} := \sum_{i:s_i \in \mathcal{X}_{j,k}} \psi_i$. We also define $\tilde{\rho} = \tilde{\rho}(\pi)$ as the set of all split vectors which satisfy $\tilde{\rho}_k = j \Rightarrow \mu_{j,k-1} \geq \mu_{j',k-1}$ for all $1 \leq j' \leq X_0 + k - 1$ and all $1 \leq k \leq X - X_0$.*

The value $\mu_{j,k} = \mu_{j,k}(\pi)$ is the stable state probability of the agent visiting states in the cell $\mathcal{X}_{j,k}$ (assuming some policy π). This definition will be important as PASA progressively generates an estimate of this value, using the vectors u and $\bar{u}$. The set $\tilde{\rho}$ is the set of all split vectors which make the "correct" decision for each cell split (i.e. for some $\rho \in \tilde{\rho}$, the cell in Ξ_{k-1} with index ρ_k has the equal highest stable-state visit probability of all the cells in Ξ_{k-1} for $1 \leq k \leq X - X_0$). The set $\tilde{\rho}_{[1:k]}$ is defined such that $\rho_{[1:k]} \in \tilde{\rho}_{[1:k]}$ if and only if there exists $\rho' \in \tilde{\rho}$ such that $\rho'_{[1:k]} = \rho_{[1:k]}$. We require one final definition.

Definition 3 *If, for each $1 \leq i \leq X_0 + k$, and for all $\varepsilon > 0$, $h > 0$ and $\tau \in \mathbb{N}$, there exists $\varsigma_{[1:(X_0+k)]}$, H_i (a closed interval on the real line of length h which satisfies $\mu_{i,0} \in H_i$ for $i \leq X_0$ and $\mu_{i,i-X_0} \in H_i$ for $X_0 < i \leq X$) and T_i such that each*

$$I_i := I_{\{\bar{u}_i \in H_i\}}$$

is ε-fixed over τ after T_i we will say that $\bar{u}$ can be stabilised up to k. *Similarly if for all ε and τ there exists $\varsigma_{[1:(X_0+k-1)]}$, $\vartheta > 0$ and T such that $\rho_{[1:k]}$ is ε-fixed at $\tilde{\rho}_{[1:k]}$ over τ after T we will say that* ρ can be uniquely stabilised up to k.

The implications of this definition are as follows. If $\bar{u}$ can be stabilised up to k, it means we can find values $\varsigma_{[1:(X_0+k)]}$ (likely to be small) such that we can ensure that the value of each $\bar{u}_i$ for $1 \leq i \leq X_0 + k$ will, with high probability, eventually come to rest within a small interval H_i of the real line. Similarly, if ρ can be uniquely stabilised up to k, it implies that we can find values $\varsigma_{[1:(X_0+k-1)]}$ and ϑ (the value for ϑ is also likely to be small) so that $\rho_{[1:k]}$ will eventually remain at a fixed value with the properties of $\tilde{\rho}$ (i.e. such that each split is "correct") with high probability for a long period. The reason for introducing the index k into our definition is because our arguments below, wherein we establish that ρ will converge in the sense described in Definition 1, will depend on induction, and an interdependence between the elements of $\bar{u}$ and ρ.

Proposition 4 *For any particular instance of the pair (P, π), for every $\tau \in \mathbb{N}$ and $\varepsilon > 0$, there exists ς, $\vartheta > 0$ and T such that the vector $\rho^{(t)}$ generated by a PASA algorithm parametrised by ς and ϑ will become ε-fixed at $\tilde{\rho}$ over τ after T.*

Proof We will argue by induction. We will want to establish each of the following partial results, from which the result will follow: (1) $\bar{u}$ can be stabilised up to 0; (2) For $1 \leq k \leq X - X_0$, if $\bar{u}$ can be stabilised up to $k-1$, then ρ can be uniquely stabilised up to k; and, (3) For $1 \leq k \leq X - X_0$, if $\bar{u}$ can be stabilised up to $k-1$ then $\bar{u}$ can be stabilised up to k.

We begin with (1). We can argue this by relying on results regarding stochastic approximation algorithms with fixed step sizes. We rely on the following (much more general) result: Theorem 2.2 in Chapter 8 of Kushner and Yin (2003). We will apply the result to each $\bar{u}_i$ (for $1 \leq i \leq X_0$). The result requires that a number of assumptions hold (see Appendix A.1 where we state the assumptions and verify that they hold in this case) and states, in effect for our current purposes, the following.[32] For all $\delta > 0$, the fraction of iterations the value of $\bar{u}_i$ will stay within a δ-neighbourhood of the limit set of the ordinary differential equation (ODE) of the update algorithm for $\bar{u}_i$ over the interval $\{0, \dots, T\}$ goes to one in probability as $\varsigma_i \to 0$ and $T \to \infty$. Recalling that, for all $1 \leq i \leq X_0$, $\bar{u}_i$ is initialised at zero, in our case the ODE is $\mu_{i,0}(1 - e^{-t})$ and the limit set is the point $\mu_{i,0}$ (this statement can be easily generalised to any initialisation of $\bar{u}_i$). The result therefore means that for any τ, h and ε we can, for each $\bar{u}_i$, choose T_i, ς_i and find $H_i \ni \mu_{i,0}$ such that I_i will be ε-fixed over τ after T_i, such that (1) holds.

We now look at (2). To see this holds, we elect arbitrary values τ' and ε' for which we need to find suitable values T', ϑ and $\varsigma'_{[1:(X_0+k-1)]}$. Suppose we set $0 < 2\vartheta \leq \min\{|\mu_{j,k'} - \mu_{j',k'}| : 0 \leq k' \leq k-1, 1 \leq j \leq X_0 + k', 1 \leq j' \leq X_0 + k', \mu_{j,k'} \neq \mu_{j',k'}\}$ (if $\mu_{j,k'}$ is the same for all j for all k', any value $\vartheta > 0$ can be chosen). Furthermore, using our assumption regarding $\bar{u}$, we select $h < \vartheta/2k$, noting that, if each $\bar{u}_i$ for $1 \leq i \leq k-1$ remains within an interval of size $\vartheta/2k$ for the set of iterations $T' \leq t \leq T' + \tau'$, then each u_i for $1 \leq i \leq X_0 + k'$ (for each generated value of u in step k' of the sequence of updates of u for $k' \leq k-1$) will remain in an interval of length $\vartheta/2$ over the same set of iterations (since each u_i will be a set of additions of these values).

For any $k' \leq k-1$, define $i_{\max} := \arg\max_i\{\mu_{i,k'} : 1 \leq i \leq X_0 + k'\}$ (taking the lowest index if this is satisfied by more than one index). If, for any $1 \leq i' \leq X_0 + k'$, $\mu_{i',k'} \neq \mu_{i_{\max},k'}$, then, provided each $\bar{u}_i$ for $1 \leq i \leq X_0 + k'$ remains in an interval of length h over the iterations $T' \leq t \leq T' + \tau'$:

$$\begin{aligned} u^{(t)}_{i_{\max}} - u^{(t)}_{i'} &\geq \mu_{i_{\max},k'} - h(k'+1) - \big(\mu_{i',k'} + h(k'+1)\big) \\ &\geq 2\vartheta - 2h(k'+1) > \vartheta, \end{aligned}$$

for the value of u at step k' and for all t satisfying $T' \leq t \leq T' + \tau'$ so that $\rho_{k'} \in \tilde{\rho}_{k'}$ for $T' \leq t \leq T' + \tau'$ for all $1 \leq k' \leq k$, where $\tilde{\rho}_{k'+1}$ is the set of integers $\{m : \rho_{k'+1} = m, \rho \in \tilde{\rho}\}$.

Again for any $k' \leq k-1$, if, for some i', $\mu_{i_{\max},k'} = \mu_{i',k'}$, let's assume (without loss of generality) that $\rho^{(T')}_{k'+1} = i_{\max}$. We will have, again provided each $\bar{u}_i$ for $1 \leq i \leq X_0 + k'$ remains in an interval of length h over the iterations $T' \leq t \leq T' + \tau'$:

$$\begin{aligned} u^{(t)}_{i'} - u^{(t)}_{i_{\max}} &\leq \mu_{i',k'} + h(k'+1) - \big(\mu_{i_{\max},k'} - h(k'+1)\big) \\ &< \mu_{i',k'} + \frac{\vartheta}{2} - \left(\mu_{i_{\max},k'} - \frac{\vartheta}{2}\right) = \vartheta, \end{aligned}$$

32. The result is in fact stated with reference to a time-shifted interpolated trajectory of $\bar{u}_i$, where the trajectory of $\bar{u}_i$ is uniformly interpolated between each discrete point in the trajectory and then scaled by ς_i. The result as we state it in the body of the proof follows as a consequence.

for all t satisfying $T' \leq t \leq T'+\tau'$ which implies that $\rho_{k'+1}$ will not change for $T' \leq t \leq T'+\tau'$ for all $1 \leq k' \leq k$. As a result if we choose, from our assumed condition regarding $\bar{u}$, ε so that $(1-\varepsilon)^{X_0+k-1} \geq 1-\varepsilon'$, and we choose $\tau = \tau' + \nu$ (to allow for the interval delay before ρ is updated), then (2) is satisfied, since we can choose $T' = \max_i T_i$ and $\varsigma'_{[1:(X_0+k-1)]} = \varsigma_{[1:(X_0+k-1)]}$ where $\varsigma_{[1:(X_0+k-1)]}$ and each T_i are chosen to satisfy our choices for ε and τ.

Finally, we examine (3). Suppose we choose the values τ, h and ε and must find suitable values ς_i, T_i and H_i (for $1 \leq i \leq X_0+k$). By assumption, for each $\bar{u}_i$ for $1 \leq i \leq X_0+k-1$, we can find suitable values in order to satisfy the condition. However we also know, from the arguments in (2), that by selecting suitable values ε_1, h_1 and τ_1 to which our assumption regarding $\bar{u}_i$ for $1 \leq i \leq X_0+k-1$ applies, we can ensure, for any values of ε_2 and τ_2, that $\rho_{[1:k]}$ will be ε_2-fixed over τ_2 after T for some T. Note that if, for some i, I_i is ε-fixed over τ after T for some H_i of length h then I_i will also be ε'-fixed over τ' after T for some H'_i of length h' for any $\varepsilon' > \varepsilon$, $\tau' < \tau$ and $h' > h$. This last observation means that we can choose ε_0, h_0 and τ_0 so that $\varepsilon_0 < \varepsilon_1$, $\varepsilon_0 < \varepsilon$, $h_0 < h_1$, $h_0 < h$, $\tau_0 > \tau_1$ and $\tau_0 > \tau$, so that, for any ε_2, τ_2, ε, h and τ, we can find suitable values ς_i, T_i and H_i (for $1 \leq i \leq X_0+k-1$) so that all conditions are satisfied.

Again relying on the result from Kushner and Yin (2003), for any ε_3, h and τ there exists ς_{X_0+k}, H_{X_0+k} of length h and T'' which will ensure that $\bar{u}_{X_0+k}$ will remain in H_{X_0+k} with probability at least $1-\varepsilon_3$ for all t such that $T'' \leq t \leq T''+\tau$, provided the value of $\rho_{k'}$ for $1 \leq k' \leq k$ is held fixed for all $t \leq T''+\tau$, for any starting value of $\bar{u}_k$ bounded by the interval $[-1, 1]$ (the limit set of the ODE is the same for all such starting values, and since the interval is compact we can choose the minimum value ς_{X_0+k} required to satisfy the condition for all starting values). Now, we can choose ε_2 and ε_3 such that $(1-\varepsilon) > (1-\varepsilon_2)(1-\varepsilon_3)$ and choose τ_2 such that $\tau_2 \geq T''+\tau$. In this way, given the value of ς_{X_0+k} shown to exist above and $T_{X_0+k} \geq \max_i T_i + T''$ we will have the required values to ensure the necessary property also holds for $\bar{u}_{X_0+k}$. ■

Since ρ completely determines Ξ the result implies the convergence of Ξ to a partition Ξ_{lim}. This fact means that SARSA-P will converge if π is held fixed (this is discussed in more detail below). Moreover a straightforward extension of the arguments in Gordon (2001) further implies that SARSA-P will not *diverge* even when π is updated. The result also implies that Ξ_{lim} will have the property that more frequently visited states will, in general, occupy smaller cells.

Note again that we have taken care to allow the vector ς to remain fixed as a function of t. This will, in principle, allow PASA to adapt to changes in π (assuming we allow π to change). We will use fixed step sizes in our experiments below. Whilst in our experiments in Section 4 we use only a single step size parameter (as opposed to a vector), the details of the proof above point to why there may be merit in using a vector of step size parameters as part of a more sophisticated implementation of the ideas underlying PASA (i.e. allowing ς_k to take on larger values for larger values of the index k, for $k > X_0$, may allow the algorithm to converge more rapidly).

The following related property of PASA will also be important for our subsequent analysis. The result gives conditions under which a guarantee can be provided that certain states will occupy a singleton cell in Ξ_{lim}. It will be helpful to define (i) as the index j which satisfies the condition $|\{k : \psi_k > \psi_j \text{ or } (\psi_k = \psi_j, k < j)\}| = i - 1$ (i.e. it is the index of the ith most frequently visited state, where we revert to the initial index ordering in case of equal values). Adopting this notation we therefore interpret, for example, $s_{(i)}$ as the ith most frequently visited state according to ψ. We continue to treat ς as a vector.

Proposition 5 *Suppose, given a particular instance of the pair* (P, π) *and a PASA algorithm parametrised by* X*, that, for some* i *satisfying* $1 \leq i \leq S$, $X \geq i\lceil \log_2 S \rceil$ *and* $\psi_{(i)} > \sum_{j=i+1}^{S} \psi_{(j)}$. *Then the mapping* Ξ_{lim} *obtained by applying Proposition 4 will be such that the state* $s_{(i)}$ *occupies a singleton cell.*

Proof Suppose that the conditions are satisfied for some index i but that the result does not hold. This implies that for at least one split in the sequence over $1 \leq l \leq X - X_0$ via which Ξ_{lim} is defined, PASA had the option to split a cell containing at least one state $s_{(i')}$ for $i' \leq i$, we will call this cell $\mathcal{X}_{k,l}$, and instead split an alternative cell $\mathcal{X}_{k',l}$ which contained no such state. This follows from the fact that $X \geq i\lceil \log_2 S \rceil$. As a result of our second assumption the cell $\mathcal{X}_{k,l}$ must be such that $\mu_{k,l} > \mu_{k',l}$. However this creates a contradiction to Proposition 4. ■

Finally, we turn to a further proposition, which applies to *sets* of pairs (P, π), and which will be important for our discussion in Section 3.5. A limitation of Proposition 4 is that, whilst we may be able to find parameters ϑ and ς to ensure that Ξ converges for a single pair (P, π), if, instead of a single pair (P, π), we are given an infinite set of pairs (we may, for example, only be given a prior distribution for P, which could be supported over a continuous range of values), then for certain pairs our chosen parameter values may not guarantee convergence in the sense we've defined.

We can address this by adjusting our arguments slightly, at the expense of some additional assumptions, and a slight (though relatively minor) weakening of the statement of the result. Appendix A.5 outlines some minor variants to the PASA algorithm (which are principally relevant to our experiments in Section 4). Crucially, for our next result, we need to assume that the PASA algorithm is altered in the sense indicated in points 3 and 4 of Appendix A.5. The effect of these changes will be that the order in which splits occur will not, for our current purposes, affect the final ordering of the cells in Ξ. This will permit us to guarantee, with the help of our next result, that Ξ will remain fixed, in a way which would be difficult otherwise. Note that these changes do not impact any of the discussion in Section 3.1, and are computationally very cheap to implement. Furthermore, all of the results stated in this article continue to hold (albeit with slightly adjusted proofs) in the presence of these changes. We need one more definition:

Definition 6 *Suppose that* $\mathcal{Q}$ *is a set and* $\mathcal{R}$ *is a set of subsets of* $\mathcal{Q}$. *Suppose also that* $x = x^{(t)}$ *and that* $x^{(t)} \in \mathcal{Q}$ *for all* t. *We will say that* x becomes ε-constrained at $\mathcal{R}$ over τ after T *provided* T *is such that, for all* $T' > T$, *with probability at least* $1 - \varepsilon$ *there is at least one element* $r \in \mathcal{R}$ *such that* $x \in r$ *for all* t' *satisfying* $T' \leq t' \leq T' + \tau$.

The definition allows us to loosen the concept of convergence slightly, such that we can define a set of subsets, and provide that a variable x will eventually remain in one of these subsets for a long time with high probability. We don't know which subset the value will converge to in advance, and, once the value has converged to a subset, it may continue to change, however it will remain, with high probability, within that subset.

The proposition we now outline provides a guarantee that certain (high probability) states will occupy a singleton cell for all pairs (P, π) in a set, given some single pair of parameters ς and ϑ. Unlike Proposition 4, however, it does not guarantee that ρ will eventually be fixed at an element of $\tilde{\rho}$ per se. It instead guarantees that ρ becomes constrained to take a (possibly changing) value within a larger set of possible split vectors (containing $\tilde{\rho}$), characterised by certain states occupying singleton cells.

The result requires some further additional terminology which we set out within the result itself and its proof. The terminology is not used subsequently in this article.

Proposition 7 *Suppose that φ, ξ and κ are constants such that $\varphi > 0$, $\xi > 2\varphi$ and κ is a positive integer. Suppose also that $\mathcal{Z}$ is a set of pairs (P, π) each element of which is defined on the same set of states and actions. Suppose, finally, that for all pairs in $\mathcal{Z}$, there exists a subset of states $\mathcal{I}(P, \pi) \subset \mathcal{S}$ of size $|\mathcal{I}| \leq \kappa$ such that $\sum_{i:s_i \notin \mathcal{I}} \psi_i \leq \varphi$ and $s_i \in \mathcal{I} \Rightarrow \psi_i \geq \xi$. Then for all $\varepsilon > 0$ and $\tau \in \mathbb{N}$ there exists ς, ϑ and T such that, given a PASA algorithm (adjusted according to points 3 and 3 of Appendix A.5) with parameters ς, ϑ and $X \geq \kappa\lceil \log_2 S \rceil$, for all elements of $\mathcal{Z}$, ρ will become ε-constrained at $\hat{\rho}$ over τ after T, where $\hat{\rho} = \hat{\rho}(P, \pi)$ is a set of subsets of all possible split vectors, and where each element r of $\hat{\rho}$ is a set of split vectors where:*

1. *All split vectors in r correspond to the same final partition $\Xi(r)$; and,*
2. *The partition $\Xi(r)$ is such that each element of $\mathcal{I}$ occupies a singleton cell.*

Proof Begin by defining ρ^* as the set of split vectors which satisfy the constraint that, for $1 \leq l \leq X - X_0$, $\rho_l \in \{m : \mathcal{I} \cap \mathcal{X}_{m,l-1} \neq \emptyset, |\mathcal{X}_{m,l-1}| > 1\}$ whenever this last-mentioned set is non-empty (if it is empty, ρ_l has no constraints beyond those applicable to all split vectors). We then define $\hat{\rho}$ as the set of subsets defined if we partition ρ^* such that all elements of ρ^* which correspond to the same partition Ξ fall into the same element of $\hat{\rho}$. We can see that $\hat{\rho}$ will have the properties required by the result. Note that every split vector ρ can be mapped to a split vector $\rho_{[1:k]}$ by removing its last $X - X_0 - k$ values. If we map every split vector ρ to $\rho_{[1:k]}$ and preserve its membership of each subset of $\hat{\rho}$, we will have a new set of subsets of the set of all split vectors $\rho_{[1:k]}$, which we denote $\hat{\rho}_{[1:k]} = \hat{\rho}_{[1:k]}(P, \pi)$.

To obtain the result itself we proceed in much the same manner as Proposition 4. We first, however, introduce two new pieces of terminology. If $\bar{u}$ can be stabilised up to k for all elements of $\mathcal{Z}$ for the same values of $\varsigma_{[1:(X_0+k)]}$ and T_i for $1 \leq i \leq X_0 + k$, we will say that $\bar{u}$ *can be stabilised up to k for all $\mathcal{Z}$*. Furthermore, if for all ε and τ there exists T, ϑ and $\varsigma_{[1:(X_0+k-1)]}$ such that, for all elements of $\mathcal{Z}$, $\rho_{[1:k]}$ becomes ε-constrained at $\hat{\rho}_{[1:k]}$ over τ after T, we will say that ρ *can be effectively stabilised up to k for all $\mathcal{Z}$*.

We will argue by induction, relying on the following three claims: (1) $\bar{u}$ can be stabilised up to 0 for all $\mathcal{Z}$; (2) For $1 \leq k \leq X - X_0$, if $\bar{u}$ can be stabilised up to $k - 1$ for all $\mathcal{Z}$, then ρ can be effectively stabilised up to k for all $\mathcal{Z}$; and, (3) If $\bar{u}$ can be stabilised up to $k - 1$ for all $\mathcal{Z}$ then $\bar{u}$ can be stabilised up to k for all $\mathcal{Z}$. This will be enough to establish the result.

For (1), since the set of all possible values (P, π) is compact, and using an identical argument to that used in relation to statement (1) from the proof of Proposition 4, we can find $\varsigma_{[1:X_0]}$ and T_i for $1 \leq i \leq X_0$ such that $\bar{u}$ will be stabilised up to 0 for all elements of $\mathcal{Z}$.

For (2), suppose, for any $k' \leq k-1$, that $\mathcal{X}_{l,k'}$ is a cell which contains an element of $\mathcal{I}$, and $\mathcal{X}_{l',k'}$ and $\mathcal{X}_{l'',k'}$ are cells which contain no such element. We make two observations. First, for all $\mathcal{Z}$ over the iterations $T' \leq t \leq T' + \tau'$:

$$\begin{aligned} u_l^{(t)} - u_{l'}^{(t)} &\geq \sum_{i:s_i \in \mathcal{X}_{l,k'}} \psi_i - h(k'+1) - \left(\sum_{i:s_i \in \mathcal{X}_{l',k'}} \psi_i + h(k'+1) \right) \\ &\geq \mu_{l,k'} - hk - \big(\mu_{l',k'} + hk\big) \geq \xi - \varphi - 2hk. \end{aligned}$$

Second, again for all $\mathcal{Z}$ and over the iterations $T' \leq t \leq T' + \tau'$, suppose that $\Xi^{(T'-1)}$ contains at least one cell which is a strict subset of $\mathcal{X}_{l'',k'}$ (such that the split vector $\rho^{(T'-1)}$ splits $\mathcal{X}_{l'',k'}$ at some point):

$$\begin{aligned} u_{l'}^{(t)} - u_{l''}^{(t)} &\leq \mu_{l',k'} + h(k'+1) - \big(\mu_{l'',k'} - h(k'+1)\big) \\ &\leq \mu_{l',k'} + hk - \big(\mu_{l'',k'} - hk\big) \leq \varphi + 2hk. \end{aligned}$$

Now, since we can choose any value of h in relation to $\bar{u}_i$ for $1 \leq i \leq X_0 + k - 1$, and any value of ϑ, we choose values (which must exist) to satisfy the following inequality (recalling that ξ and φ are, by assumption, fixed for all elements of $\mathcal{Z}$, and that $\xi > 2\varphi$):

$$\varphi + 2hk \leq \vartheta < \xi - \varphi - 2hk.$$

Once this is done, we can see that, whenever at least one non-singleton cell is available containing a state in $\mathcal{I}$, it will be split before any cell *not* containing such a state is split. Furthermore, once all states in $\mathcal{I}$ occupy a singleton cell, the remaining splits which occur will remain unchanged each time Ξ is updated (though the order in which the splits occur may change). As a result, for any ε' and τ', we can choose h, ϑ, ε and τ in relation to our assumption regarding $\bar{u}$ so that h and ϑ satisfy our above assumption, so that $(1-\varepsilon)^{X_0+k-1} \geq 1 - \varepsilon'$, and so that $\tau = \tau' + \nu$. If the values $\varsigma_{[1:(X_0+k-1)]}$ and T_i for $1 \leq i \leq X_0 + k - 1$ are required to obtain h, ε and τ, then the values $T' = \max_i T_i$ and $\varsigma_{[1:(X_0+k-1)]}$ will obtain ε' and τ'. As a consequence ρ can be effectively stabilised up to k for all $\mathcal{Z}$, and (2) will hold.

Finally statement (3) follows in the same manner as statement (3) in Proposition 4, again using the compactness of the set of all possible values of (P, π) to ensure that we can find T, ς and ϑ such that the requirement will be satisfied for every element of $\mathcal{Z}$. ■

Implicitly, our interest in later sections will primarily be in cases where φ is small, and κ is relatively small compared to S, such that for each pair (P, π) a small subset of $\mathcal{S}$ has total visit probability close to one.

The implication of Proposition 7 is that, whilst ρ will converge to a set of values (and may alter within that set), since we have assumed that PASA is altered according to points 3 and 4 of Appendix A.5, the properties of $\hat{\rho}$ are such that the value Ξ will converge for all pairs in the set $\mathcal{Z}$, and in a way such that a set of states with high visit probability fall into singleton cells. Note that, for states outside the set $\mathcal{I}$, many of our subsequent results will not actually require that Ξ converges with respect to its partitioning of these states. It seems likely that different arguments and slightly different assumptions could be made to obtain results which are not identical to, but carry many of the key implications of, the results as they are presented in this article.

3.3. Potential to Reduce Value Function Error Given Fixed π

The results in this subsection again apply to the case of policy evaluation only. In the tabular case, traditional RL algorithms such as SARSA provide a means of estimating Q^π which, assuming fixed π, will converge to the correct value as t becomes large. Once we introduce an approximation architecture, however, we no longer have any such guarantee, even if the estimate is known to converge. In relation to SARSA-F, there is little we can say which is non-trivial in relation to bounds on error in the VF estimate.[33]

Since the mapping generated by PASA converges to a fixed mapping (when suitably parametrised), then, assuming a fixed policy, if an RL algorithm such as SARSA is used to update $\hat{Q}_\theta$, the estimate $\hat{Q}_\theta$ will also converge (as noted in Section 2 above). If we know the convergence point we can then assess the limit of $\hat{Q}_\theta$ using an appropriate scoring function.

Our next result will provide bounds on the error in VF approximations generated in such a way. We will use PASA to guarantee that, under suitable conditions, a subset of the state space will be such that each element in that subset will fall into its own singleton cell. This has a powerful effect since, once this is the case, we can start to provide guarantees around the contribution to total VF error associated with those states.

The result will be stated in relation to the scoring functions defined in Section 1.4. We assume that the parameter γ selected for SARSA is the same as the parameter γ used to define each scoring function. We also assume SARSA has a fixed step size η. We will need four more definitions.

Definition 8 *For a particular transition function P and an arbitrary subset $\mathcal{I}$ of $\mathcal{S}$ define h (which is a function of π as well as of $\mathcal{I}$) as follows:*

$$h(\mathcal{I}, \pi) := \sum_{i: s_i \in \mathcal{I}} \psi_i.$$

Note that $h(\mathcal{I}, \pi)$ is the proportion of time that the agent will spend in the subset $\mathcal{I}$ when following the policy π. It must take a value in the interval $[0, 1]$.

Definition 9 *For any $\delta \geq 0$, define (a) P, (b) R and (c) π respectively as being δ-*deterministic *if (a) P can expressed as follows: $P = (1-\delta)P_1 + \delta P_2$, where P_1 is a deterministic transition function and P_2 is an arbitrary transition function, (b) for all s_i and a_j, $\mathrm{Var}(R(s_i, a_j)) \leq \delta$, and (c) π can expressed as follows: $\pi = (1-\delta)\pi_1 + \delta\pi_2$, where π_1 is a deterministic policy and π_2 is an arbitrary policy.*

33. A result does exist which provides a bound on the extent to which the error in the VF estimate generated by SARSA-F exceeds the minimum possible error $\min_\theta \mathrm{MSE}(\theta)$ for all possible values of the matrix θ. See Bertsekas and Tsitsiklis (1996). Since SARSA generates its estimates using temporal differences, it will not generally attain this minimum. However, there is still little that we can say about the magnitude of $\min_\theta \mathrm{MSE}(\theta)$.

The three parts of the definition mean that, as δ moves closer to zero, a δ-deterministic transition function, a δ-deterministic reward function and a δ-deterministic policy respectively become "more deterministic".

Definition 10 *For a particular pair (P, π) and any $\delta \geq 0$, we define $\mathcal{I}$ as being* δ-constrained *provided that, for every state s_i in $\mathcal{I}$, $\sum_{j=1}^{A} \pi(a_j|s_i) \sum_{i':s_{i'} \notin \mathcal{I}} P(s_{i'}|s_i, a_j) \leq \delta$.*

This definition means that, for the pair (P, π), if $\mathcal{I}$ is δ-constrained, the agent will transition from a state in $\mathcal{I}$ to a state outside $\mathcal{I}$ with probability no greater than δ.

Definition 11 *For a particular triple (P, R, π) we also define the four values (a) δ_P, (b) δ_R, (c) δ_π and (d) $\delta_\mathcal{I}$ as (a) $\min\{\delta : P \text{ is } \delta\text{-deterministic}\}$, (b) $\min\{\delta : R \text{ is } \delta\text{-deterministic}\}$, (c) $\min\{\delta : \pi \text{ is } \delta\text{-deterministic}\}$ and (d) $\min\{\delta : \mathcal{I} \text{ is } \delta\text{-constrained}\}$ respectively.*

Each of δ_P, δ_R, δ_π and $\delta_\mathcal{I}$ must fall on the interval $[0, 1]$. Note that, for any $\mathcal{I}$ satisfying $\min\{\psi_i : s_i \in \mathcal{I}\} > 1 - h(\mathcal{I}, \pi)$, for any $\delta \geq 0$ there will exist $\delta' \geq 0$ such that if P and π are both δ'-deterministic, then $\mathcal{I}$ must be δ-constrained (since for any $\delta < 1$, if $\delta' = 0$, and $\mathcal{I}$ is not δ-constrained, then, for some i and i', where $s_{i'} \notin \mathcal{I}$ and $s_i \in \mathcal{I}$, s_i must deterministically transition to $s_{i'}$, and hence $\psi_{i'} \geq \psi_i$, contradicting our just-stated assumption with respect to $\mathcal{I}$).

Theorem 12 *For a particular instance of the triple (P, R, π), suppose that $|R(s_i, a_j)|$ is bounded for all i and j and that the constant R_m denotes the maximum of $|\mathrm{E}(R(s_i, a_j))|$ for all i and j. Take any subset $\mathcal{I}$ of $\mathcal{S}$. If $X \geq |\mathcal{I}|\lceil \log_2 S \rceil$ and $\min\{\psi_i : s_i \in \mathcal{I}\} > 1 - h(\mathcal{I}, \pi)$ then, for all $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ there exists T, η, ϑ and a parameter vector ς such that, provided $t \geq T$, with probability equal to or greater than $1 - \varepsilon_2$ the VF estimate $\hat{Q}^{(t)}$ generated by SARSA-P will be such that:*

1. *If $w(s_i, a_j) = \psi_i \pi(a_j|s_i)$ then:*

$$\mathrm{MSE} \leq \left(2(1-h) + \delta_\mathcal{I} + \frac{\delta_\mathcal{I}^2 \gamma^2}{1-\gamma} + \frac{\delta_\mathcal{I}^2 \gamma^4}{(1-\gamma)^2} \right) \frac{2R_\mathrm{m}^2}{(1-\gamma)^2} + \varepsilon_1;$$

2. *If $w(s_i, a_j) = \psi_i \tilde{w}(s_i, a_j)$ for an arbitrary function $\tilde{w}$ satisfying $0 \leq \tilde{w}(s_i, a_j) \leq 1$ and $\sum_{j'=1}^{A} \tilde{w}(s_i, a_{j'}) \leq 1$ for all i and j then:*

$$L \leq \frac{4(1-h)}{(1-\gamma)^2} R_\mathrm{m}^2 + \varepsilon_1;$$

and:

$$\tilde{L} \leq \left(4(1-h) + \gamma^2 \left(1 + 2(1-\delta_P)(1-\delta_\pi) - 3(1-\delta_P)^2(1-\delta_\pi)^2\right)\right) \frac{R_\mathrm{m}^2}{(1-\gamma)^2} + \delta_R + \varepsilon_1.$$

Remark 13 *The bounds just stated generally only become important when h is close to 1 (i.e. the agent spends a large proportion of its time in a subset $\mathcal{I}$ of $\mathcal{S}$ which satisfies the theorem's conditions), and (for* MSE *and $\tilde{L}$) when the environment and policy are "close to deterministic". In the case when h approaches 1 and the environment and policy are arbitrarily close to deterministic, all bounds become arbitrarily close to zero.*

Proof Propositions 4 and 5 mean we can guarantee that T', ϑ and ς exist such that, for any τ and $\varepsilon_2' > 0$, Ξ will be (a) fixed for τ iterations with probability at least $1 - \varepsilon_2'$ and (b) that Ξ will be such that each element of $\mathcal{I}$ will be in a singleton cell.

Our assumption with respect to R, as well as standard results relating to stochastic approximation algorithms—see Kushner and Yin (2003) and our brief discussion in Appendix A.1—allow us to guarantee for SARSA with fixed state aggregation that, for any $\varepsilon_1' > 0$ and $\varepsilon_2'' > 0$, there exists τ and η such that, provided we have a fixed partition Ξ' for the interval τ, $\hat{Q}(s_i, a_j)$ will be within ε_1' of a convergence point $\hat{Q}_{\text{lim}}(s_i, a_j)$ associated with Ξ' for every i and j with probability at least $1 - \varepsilon_2''$.

This means that, for any ε_2 and ε_1', by choosing ε_2' and ε_2'' so that $(1-\varepsilon_2')(1-\varepsilon_2'') > 1-\varepsilon_2$ we can find T (and η, ϑ and ς) such that, for all $t > T$, $|\hat{Q}(s_i, a_j) - \hat{Q}_{\text{lim}}(s_i, a_j)| \leq \varepsilon_1'$ for all i and j with probability at least $1 - \varepsilon_2$, where $\hat{Q}_{\text{lim}}$ is the limit point associated with the partition Ξ described above. We will use this fact in our proof of each of the three inequalities. We will also use the fact that, in general, for any state s_i in a singleton cell, and for each a_j, for $\hat{Q}_{\text{lim}}$ we will have:

$$\begin{aligned}
&\mathrm{E}\left(\Delta\hat{Q}_{\text{lim}}(s_i, a_j)\middle| s^{(t')} = s_i, a^{(t')} = a_j\right) \\
&\qquad = \mathrm{E}\left(R(s_i, a_j) + \gamma\hat{Q}_{\text{lim}}\big(s^{(t'+1)}, a^{(t'+1)}\big) - \hat{Q}_{\text{lim}}(s_i, a_j)\middle| s^{(t')} = s_i, a^{(t')} = a_j\right) = 0,
\end{aligned}$$

where $\Delta\hat{Q}_{\text{lim}}(s_i, a_j)$ is equal to $\hat{Q}^{(t'+1)}(s_i, a_j) - \hat{Q}^{(t')}(s_i, a_j)$ assuming $\hat{Q}^{(t')} = \hat{Q}_{\text{lim}}$,[34] from which we can infer that:

$$\hat{Q}_{\text{lim}}(s_i, a_j) = \mathrm{E}\left(R(s_i, a_j) + \gamma\hat{Q}_{\text{lim}}\big(s^{(t'+1)}, a^{(t'+1)}\big)\middle| s^{(t')} = s_i, a^{(t')} = a_j\right).$$

For L, if $|\hat{Q}(s_i, a_j) - \hat{Q}_{\text{lim}}(s_i, a_j)| \leq \varepsilon_1'$ for every i and j with probability at least $1 - \varepsilon_2$, then the equation above immediately implies that each term in L corresponding to a state in $\mathcal{I}$ will be less than $4\varepsilon_1'^2$ with probability of at least $1-\varepsilon_2$. Furthermore, for states outside the set $\mathcal{I}$, the maximum possible value of each such term is:

$$\left(\sum_{t=1}^{\infty} \gamma^{t-1} 2R_{\text{m}} + 2\varepsilon_1'\right)^2 \leq \frac{4R_{\text{m}}^2}{(1-\gamma)^2} + \varepsilon_1'',$$

where for any $\varepsilon_1'' > 0$ there exists $\varepsilon_1' > 0$ so that the inequality is satisfied. By selecting ε_1' and ε_1'' such that $4\varepsilon_1'^2 + \varepsilon_1'' < \varepsilon_1$ this gives us the result for L (where we ignore the factor of h in relation to terms for states in $\mathcal{I}$, since this factor will only decrease the bound given that $h \leq 1$).

For $\tilde{L}$ we will use the temporary notation $\lambda \coloneqq (1 - \delta_P)(1 - \delta_\pi)$. Again, suppose that, with probability at least $1-\varepsilon_2$ we have $|\hat{Q}(s_i, a_j) - \hat{Q}_{\text{lim}}(s_i, a_j)| \leq \varepsilon_1'$ for every i and j. In the next equation, for notational convenience, we refer to $R(s_i, a_j)$ as $\tilde{R}$ and $\hat{Q}_{\text{lim}}(s^{(t'+1)}, a^{(t'+1)})$ (which is a random variable conditioned on $s^{(t')} = s_i$ and $a^{(t')} = a_j$) as $\tilde{Q}_{\text{lim}}'$. Given a state $s^{(t')} = s_i$ occupying a singleton cell, we will have the following, with probability at least

34. Note that it is straightforward to generalise here such that the implicit assumption of ergodicity is not required, however these details are omitted.

$1 - \varepsilon_2$, for each a_j:

$$\begin{aligned}
&\mathrm{E}\Big(\tilde{R}+\gamma\hat{Q}\big(s^{(t'+1)}, a^{(t'+1)}\big) - \hat{Q}(s_i,a_j)\Big)^2 \\
&\quad\le \mathrm{E}\left(\tilde{R} + \gamma\tilde{Q}'_{\mathrm{lim}} - \mathrm{E}\big(\tilde{R} + \gamma\tilde{Q}'_{\mathrm{lim}}\big) + 2\varepsilon'_1\right)^2 \\
&\quad= \mathrm{E}\left(\big(\tilde{R} + \gamma\tilde{Q}'_{\mathrm{lim}}\big)^2\right) - \Big(\mathrm{E}\big(\tilde{R} + \gamma\tilde{Q}'_{\mathrm{lim}}\big)\Big)^2 + 4\varepsilon'^2_1 \\
&\quad= \mathrm{E}\big(\tilde{R}^2\big) + 2\gamma\mathrm{E}\big(\tilde{R}\tilde{Q}'_{\mathrm{lim}}\big) + \gamma^2\mathrm{E}\big(\tilde{Q}'^2_{\mathrm{lim}}\big) \\
&\qquad\quad - \big(\mathrm{E}(\tilde{R})\big)^2 - 2\gamma\mathrm{E}(\tilde{R})\mathrm{E}\big(\tilde{Q}'_{\mathrm{lim}}\big) - \gamma^2\left(\mathrm{E}\big(\tilde{Q}'_{\mathrm{lim}}\big)\right)^2 + 4\varepsilon'^2_1 \\
&\quad= \mathrm{E}\big(\tilde{R}^2\big) - (\mathrm{E}(\tilde{R}))^2 + \gamma^2\mathrm{E}\big(\tilde{Q}'_{\mathrm{lim}}\big)^2 - \gamma^2\left(\mathrm{E}\big(\tilde{Q}'_{\mathrm{lim}}\big)\right)^2 + 4\varepsilon'^2_1,
\end{aligned} \tag{7}$$

where we've used the independence of R and P and of R and π. We can see that the first two terms together equal the variance of $R(s_i, a_j)$ and so are bounded above by δ_R. Suppose that $s_{i''}$ and $a_{j''}$ are the state-action pair corresponding to the deterministic transition and deterministic action following from the state and action s_i and a_j, i.e. according to the transition function P_1 and policy π_1 (which exist and are defined according to the definition of δ-deterministic for P and π respectively). We can then also expand the third and fourth terms in the final line of equation (7), temporarily omitting the factor of γ^2, to obtain:

$$\begin{aligned}
&\sum_{i'=1}^{S}\sum_{j'=1}^{A} P(s_{i'}|s_i,a_j)\pi(a_{j'}|s_{i'})\hat{Q}_{\mathrm{lim}}(s_{i'},a_{j'})^2 \\
&\qquad - \left(\sum_{i'=1}^{S}\sum_{j'=1}^{A} P(s_{i'}|s_i,a_j)\pi(a_{j'}|s_{i'})\hat{Q}_{\mathrm{lim}}(s_{i'},a_{j'})\right)^2 \\
&= \lambda\hat{Q}_{\mathrm{lim}}(s_{i''},a_{j''})^2 + \sum\nolimits_{\Omega} P(s_{i'}|s_i,a_j)\pi(a_{j'}|s_{i'})\hat{Q}_{\mathrm{lim}}(s_{i'},a_{j'})^2 \\
&\qquad - \left(\lambda\hat{Q}_{\mathrm{lim}}(s_{i''},a_{j''}) + \sum\nolimits_{\Omega} P(s_{i'}|s_i,a_j)\pi(a_{j'}|s_{i'})\hat{Q}_{\mathrm{lim}}(s_{i'},a_{j'})\right)^2,
\end{aligned}$$

where $\Omega := \{(i',j') : i' \neq i'' \text{ or } j' \neq j''\}$.

Expanding relevant terms, and noting that $\sum_{\Omega} P(s_{i'}|s_i,a_j)\pi(a_{j'}|s_{i'}) \le 1-\lambda$, our statement becomes:

$$\begin{aligned}
&\lambda\hat{Q}_{\mathrm{lim}}(s_{i''},a_{j''})^2 + \sum\nolimits_{\Omega} P(s_{i'}|s_i,a_j)\pi(a_{j'}|s_{i'})\hat{Q}_{\mathrm{lim}}(s_{i'},a_{j'})^2 - \lambda^2\hat{Q}_{\mathrm{lim}}(s_{i''},a_{j''})^2 \\
&\qquad + 2\lambda\hat{Q}_{\mathrm{lim}}(s_{i''},a_{j''})\sum\nolimits_{\Omega} P(s_{i'}|s_i,a_j)\pi(a_{j'}|s_{i'})\hat{Q}_{\mathrm{lim}}(s_{i'},a_{j'}) \\
&\qquad - \left(\sum\nolimits_{\Omega} P(s_{i'}|s_i,a_j)\pi(a_{j'}|s_{i'})\hat{Q}_{\mathrm{lim}}(s_{i'},a_{j'})\right)^2 \\
&\le \big(\lambda-\lambda^2\big)\frac{R_{\mathrm{m}}^2}{(1-\gamma)^2} + (1-\lambda)\frac{R_{\mathrm{m}}^2}{(1-\gamma)^2} + 2\lambda(1-\lambda)\frac{R_{\mathrm{m}}^2}{(1-\gamma)^2} = \big(1+2\lambda-3\lambda^2\big)\frac{R_{\mathrm{m}}^2}{(1-\gamma)^2} \\
&= \big(1+2(1-\delta_P)(1-\delta_\pi) - 3(1-\delta_P)^2(1-\delta_\pi)^2\big)\frac{R_{\mathrm{m}}^2}{(1-\gamma)^2} =: D,
\end{aligned}$$

where we've used the fact that $R_\mathrm{m}/(1-\gamma)$ is an upper bound on the magnitude of $\hat{Q}_\mathrm{lim}(s_i, a_j)$ for all i and j, and where we ignore the last term in the first statement since its contribution must be less than zero. For those states not in $\mathcal{I}$ we argue in exactly the same fashion as for L, which gives us:

$$\tilde{L} \leq \frac{4(1-h)}{(1-\gamma)^2} R_\mathrm{m}^2 + 4\varepsilon_1'^2 + \varepsilon_1'' + \gamma^2 D + \delta_R$$

with probability at least $1-\varepsilon_2$. Accordingly, for any ε_1 we can select suitable ε_1', ε_1'' so that the result is satisfied.

For MSE, we will decompose both Q^π and $\hat{Q}_\mathrm{lim}$ into different sets of sequences of states and actions. In particular, we will isolate the set of all finite sequences of states and actions which remain within the set $\mathcal{I}$, starting from a state in $\mathcal{I}$, up until the agent transitions to a state outside $\mathcal{I}$. For a single state $s_i \in \mathcal{I}$, and for all a_j, we will have:

$$\begin{aligned} Q^\pi(s_i, a_j) = {} & \underbrace{\xi^{(1)} + \sum_{t'=2}^{\infty} \Pr\left(s^{(t'')} \in \mathcal{I} \text{ for } t'' \leq t' \middle| s^{(1)} = s_i, a^{(1)} = a_j\right) \xi^{(t')}}_{=:C} \\ & + \underbrace{\Pr\left(s^{(2)} \notin \mathcal{I} \middle| s^{(1)} = s_i, a^{(1)} = a_j\right) x^{(2)}}_{=:U} \\ & + \underbrace{\sum_{t'=3}^{\infty} \Pr\left(s^{(t'')} \in \mathcal{I} \text{ for } t'' < t', s^{(t')} \notin \mathcal{I} \middle| s^{(1)} = s_i, a^{(1)} = a_j\right) x^{(t')}}_{=:V}, \end{aligned}$$

where $\xi^{(t')}$ is the expected reward at $t = t'$, conditioned upon $s^{(1)} = s_i$ and $a^{(1)} = a_j$, and conditioned upon the agent remaining within the set $\mathcal{I}$ for all iterations up to and including t'. The value $x^{(t')}$ is an expected discounted reward summed over all iterations following (and including) the first iteration t' for which the agent's state is no longer within the set $\mathcal{I}$ (each value is also conditioned upon $s^{(1)} = s_i$ and $a^{(1)} = a_j$). Each such value represents, in effect, a residual difference between the sum of terms involving ξ and $Q^\pi(s_i, a_j)$, and will be referred to below. It is for technical reasons that we separate out the term representing the case where $s^{(2)} \notin \mathcal{I}$. The reasons relate to the weighting $w(s_i, a_j) = \psi_i \pi(a_j|s_i)$ and will become clearer below. We will also have (by iterating the formula for $\hat{Q}_\mathrm{lim}$):

$$\begin{aligned} \hat{Q}_\mathrm{lim}(s_i, a_j) = C + {} & \underbrace{\Pr\left(s^{(2)} \notin \mathcal{I} \middle| s^{(1)} = s_i, a^{(1)} = a_j\right) x'^{(2)}}_{=:U'} \\ & + \underbrace{\sum_{t'=3}^{\infty} \Pr\left(s^{(t'')} \in \mathcal{I} \text{ for } t'' < t', s^{(t')} \notin \mathcal{I} \middle| s^{(1)} = s_i, a^{(1)} = a_j\right) x'^{(t')}}_{=:V'}, \end{aligned}$$

where each $x'^{(t')}$ similarly represents part of the residual difference between the summation over terms involving ξ and $\hat{Q}_\mathrm{lim}(s_i, a_j)$. We once again are permitted to assume, for sufficiently large t, that for any ε_1' and ε_2 we can obtain $|\hat{Q}^{(t)}(s_i, a_j) - \hat{Q}_\mathrm{lim}(s_i, a_j)| \leq \varepsilon_1'$ for all i and j with probability at least ε_2.

Again, we consider states inside and outside $\mathcal{I}$ separately and argue in exactly the same fashion as for L and $\tilde{L}$. This will leave us with (noting that the two values C, associated with Q^π and $\hat{Q}_{\text{lim}}$ respectively, will cancel out):

$$\begin{aligned}\text{MSE} &\leq \frac{4(1-h)R_{\text{m}}^2}{(1-\gamma)^2} + \varepsilon_1'' + \sum_{i:s_i\in\mathcal{I}} \psi_i \sum_{j=1}^{A} \pi(a_j|s_i)(U+V-U'-V'+\varepsilon_1')^2 \\ &= \frac{4(1-h)R_{\text{m}}^2}{(1-\gamma)^2} + \varepsilon_1'' \\ &\qquad + \sum_{i:s_i\in\mathcal{I}} \psi_i \sum_{j=1}^{A} \pi(a_j|s_i)(U^2+V^2+U'^2+V'^2+\varepsilon_1'^2+UV-UU'-UV'+\ldots),\end{aligned}$$

where we have abbreviated the final line, omitting most of the terms in the expansion of the squared summand. We will derive bounds in relation to U^2, V^2 and UV. Similar bounds can be derived for all other terms (in a more-or-less identical manner, the details of which we omit) to obtain the result. First we examine U^2. Note that every value $|x^{(t')}|$ and value $|x'^{(t')}|$ is bound by $R_{\text{m}}^2/(1-\gamma)^2$. We have, for each $s_i \in \mathcal{I}$:

$$\sum_{j=1}^{A} \pi(a_j|s_i)U^2 \leq \frac{R_{\text{m}}^2}{(1-\gamma)^2} \sum_{j=1}^{A} \pi(a_j|s_i) \Pr\left(s^{(2)} \notin \mathcal{I} \middle| s^{(1)} = s_i, a^{(1)} = a_j\right) \leq \delta_{\mathcal{I}} \frac{R_{\text{m}}^2}{(1-\gamma)^2}.$$

(Note, in the inequality just stated, the importance of the weighting π.)

For V^2 we have:

$$\begin{aligned}V^2 &= \left(\sum_{t'=3}^{\infty} \Pr\left(s^{(t'')} \in \mathcal{I} \text{ for } t'' < t', s^{(t')} \notin \mathcal{I} \middle| s^{(1)} = s_i, a^{(1)} = a_j\right) x^{(t')}\right)^2 \\ &\leq \left(\sum_{t'=3}^{\infty} (1-\delta_{\mathcal{I}})^{t'-2} \delta_{\mathcal{I}} \gamma^{t'-1} \sum_{u=t'}^{\infty} \gamma^{u-t'} R_{\text{m}}\right)^2 \leq \left(\frac{\delta_{\mathcal{I}}(1-\delta_{\mathcal{I}})\gamma^2}{1-(1-\delta_{\mathcal{I}})\gamma}\right)^2 \frac{R_{\text{m}}^2}{(1-\gamma)^2}.\end{aligned}$$

Similar arguments will yield:

$$|UV| \leq \frac{\delta_{\mathcal{I}}^2(1-\delta_{\mathcal{I}})\gamma^2}{1-(1-\delta_{\mathcal{I}})\gamma} \frac{R_{\text{m}}^2}{(1-\gamma)^2}.$$

As noted, bounds on all other terms can be derived in the same fashion. We can also bound any term which involves ε_1', such that, for any ε_1, we can choose ε_1' and ε_1'' (which will itself be a function of ε_1') to finally obtain:

$$\text{MSE} \leq \left(4(1-h) + 2\delta_{\mathcal{I}} + \frac{2\delta_{\mathcal{I}}^2(1-\delta_{\mathcal{I}})\gamma^2}{1-(1-\delta_{\mathcal{I}})\gamma} + \frac{2\delta_{\mathcal{I}}^2(1-\delta_{\mathcal{I}})^2\gamma^4}{(1-(1-\delta_{\mathcal{I}})\gamma)^2}\right) \frac{R_{\text{m}}^2}{(1-\gamma)^2} + \varepsilon_1,$$

which holds with probability at least $1-\varepsilon_2$ provided $t > T$.

The final result is a simplified, less tight, version of the above inequality, with each instance of $1-\delta_{\mathcal{I}}$ replaced by 1. ■

We again note the fact that the scoring function being weighted by ψ is crucial for this result. The theorem suggests that using PASA will be of most value when an environment (P, R) and a policy π are such that a subset $\mathcal{I}$ exists which has the properties that:

1. $|\mathcal{I}|$ is small compared to S; and
2. $h(\mathcal{I}, \pi)$ is close to 1.

Indeed if a subset $\mathcal{I}$ exists such that h is arbitrarily close to one and $X \geq |\mathcal{I}|\lceil \log_2 S \rceil$, then we can in principle obtain a VF estimate using PASA which has arbitrarily low error according to L. When P, π and R are deterministic, or at least sufficiently close to deterministic, then we can also make equivalent statements in relation to $\tilde{L}$ and MSE (the latter follows from using the observation in the last paragraph before Theorem 12). Whilst results which guarantee low MSE are, in a sense, stronger than those which guarantee low L or $\tilde{L}$, the latter can still be very important. Some algorithms seek to minimise L or $\tilde{L}$ by using estimates of these values to provide feedback. Hence if PASA can minimise L or $\tilde{L}$ it will compare favourably with any algorithm which uses such a method. (Furthermore, the results for L and $\tilde{L}$ of course have weaker conditions.) We discuss the extension of these results to a continuous state space setting in Section 3.4.

It is worth stressing that, for MSE for example, assuming all of the conditions stated in Theorem 12 hold—and assuming $w(s_i, a_j) = \psi_i \pi(a_j|s_i)$—then, provided P and R are unknown, and given SARSA-F with *any* fixed state aggregation with $X < S$, it is impossible to guarantee MSE $< R_{\mathrm{m}}^2/(1-\gamma)^2$ (i.e. the naïve upper bound).[35] This underscores the potential power of the result given suitable conditions.

The differences between each of the three bounds arise as a natural consequence of differences between each of the scoring functions. The bounds stated are not likely to be tight in general. It may also be possible to generalise the results slightly for different weightings $w(s_i, a_j)$ however any such generalisations are likely to be of diminishing value. All three bounds immediately extend to any projected form of any of the three scoring functions for the reason noted in Section 1.4.

Theorem 12 only ensures that, for a *particular* triple (P, R, π), we can find PASA parameters such that the result will hold. If we have an infinite set of triples (for example if we are drawing a random sample from a known prior for P), we cannot guarantee that there exists a single set of parameters such that the result will hold for all elements of the set.

However an alternative related theorem, which extends from Proposition 7, addresses this, and will be helpful for our discussion in Section 3.5. Suppose we have a set $\mathcal{Z}$ of triples (P, R, π). We can define $\delta_{P,\mathcal{Z}} := \sup\{\delta_P : (P, R, \pi) \in \mathcal{Z}\}$ and define $\delta_{R,\mathcal{Z}}$, $\delta_{\pi,\mathcal{Z}}$ and $\delta_{\mathcal{I},\mathcal{Z}}$ in an analogous way. It is possible to use identical arguments to Theorem 12, with each of δ_P, δ_R, δ_π and $\delta_{\mathcal{I}}$ replaced by $\delta_{P,\mathcal{Z}}$, $\delta_{R,\mathcal{Z}}$, $\delta_{\pi,\mathcal{Z}}$ and $\delta_{\mathcal{I},\mathcal{Z}}$ respectively, to obtain an equivalent theorem

35. This can be shown by constructing a simple example, which we briefly sketch. We know at least one cell exists with more than one state. We can assume that $\psi_i = 0$ (or is at least arbitrarily close to zero) for all but two states, both of which are inside this cell. Call these states s_1 and s_2, and assume $A = 1$ (this keeps the arguments simpler) so that we have a single action a_1. Suppose $R(s_1, a_1) = R_{\mathrm{m}}$ w.p. 1 and $R(s_2, a_1) = -R_{\mathrm{m}}$ w.p. 1, and the transition probabilities are such that s_1 transitions to s_2 with probability p, and s_2 transitions to s_1 *also* with probability p. We assume the prior for P is such that p may potentially assume an arbitrarily low value. For any fixed γ, by selecting p arbitrarily close to zero we will have MSE arbitrarily close to the non-zero lower bound we have just stated. Note that here we assume that $\eta = \eta(\gamma, p)$ is sufficiently small so that the VF estimate of SARSA converges to an arbitrarily small neighbourhood of zero. Similar arguments can be constructed to show that, for L and $\tilde{L}$, there is a similar minimum guaranteed lower bound of $R_{\mathrm{m}}^2/2(1-\gamma)^2$ (we omit the details).

which will apply to any *set* of triples (P, R, π) that satisfies the conditions of Proposition 7. This is important because it means we can place bounds on VF error, for a single SARSA-P algorithm, given only a *prior* for (P, R). The trade-off is the presence of slightly stronger assumptions. The proof is omitted as it is identical to Theorem 12, except that we rely on Proposition 7 instead of Propositions 4 and 5. Since we are using Proposition 7 we must assume we have adopted the alterations to PASA in points 3 and 4 of Appendix A.5 (see also the related comments in Section 3.2):

Theorem 14 *Take some value $\varphi \in [0, 1]$. Suppose we have a set of triples $\mathcal{Z}$, each element of which is defined on the same set of states and actions, such that, for some $\xi > 2\varphi$, every element of $\mathcal{Z}$ contains a subset of states $\mathcal{I}$ of size $|\mathcal{I}| \leq \kappa$ which satisfies $\sum_{i:s_i \notin \mathcal{I}} \psi_i \leq \varphi$ and $s_i \in \mathcal{I} \Rightarrow \psi_i \geq \xi$. Suppose also that each $|R(s_i, a_j)|$ is bounded for all i and j for every element of $\mathcal{Z}$ by a single constant. We let R_{m} (another constant) denote the supremum of $|\mathrm{E}\,(R(s_i, a_j))\,|$ for all i and j and all $\mathcal{Z}$. Then for all $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ there exists T, η, ϑ and a parameter vector ς such that, provided $t \geq T$ and $X \geq \kappa \lceil \log_2 S \rceil$, for each element of $\mathcal{Z}$ the VF estimate $\hat{Q}^{(t)}$ generated by SARSA-P (where PASA incorporates changes 3 and 4 from Appendix A.5) will be such that, with probability equal to or greater than $1 - \varepsilon_2$,* MSE, *L and $\tilde{L}$ will satisfy the bounds stated in Theorem 12, where we replace $1 - h$ with φ, and where we replace each of δ_P, δ_R, δ_π and $\delta_{\mathcal{I}}$ by $\delta_{P,\mathcal{Z}}$, $\delta_{R,\mathcal{Z}}$, $\delta_{\pi,\mathcal{Z}}$ and $\delta_{\mathcal{I},\mathcal{Z}}$ respectively.*

As a final technical note, alterations can be made to the PASA algorithm (with no impact on computational complexity) that can remove the $\lceil \log_2 S \rceil$ factor in Propositions 5 and 7 and therefore also in Theorems 12 and 14 (the alteration involves, in effect, merging non-singleton cells in the partition Ξ). However such an alternative algorithm is more complex to describe and unlikely to perform noticeably better in a practical setting.

3.4. Extension to Continuous State Spaces

The results in Sections 3.1 to 3.3 can be extended to continuous state spaces, whilst retaining all of their most important implications. We will discuss this extension informally before introducing the necessary formal definitions.

It is typical, when tackling a problem with a continuous state space, to convert the agent's input into a discrete approximation, by mapping the continuous state space to the elements of some partition of the state space. Indeed, any computer simulation of an agent's input implicitly involves such an approximation.

It is not possible, in the absence of quite onerous assumptions (for example regarding continuity of transition, policy and reward kernels), to guarantee that, in the presence of such a discrete approximation, MSE, L or $\tilde{L}$, suitably redefined for the continuous case, are arbitrarily low.[36] However we can extend our discrete case analysis as follows. We assume that we begin with a discrete approximation of the continuous state space consisting of D *atomic cells*. SARSA with a fixed state aggregation approximation architecture corresponding to the D atomic cells will have associated values for MSE, L and $\tilde{L}$, which we

36. The extension of the formal definitions for MSE, L and $\tilde{L}$ to a continuous state space setting is reasonably self evident in each case. We do not provide definitions however some relevant details are at Appendix A.2.

denote MSE_0, L_0 and $\tilde{L}_0$. Each value MSE_0, L_0 and $\tilde{L}_0$ should be considered as a minimal "baseline" error, the minimum error possible assuming our initial choice of atomic cells.

As a rule of thumb, the finer our approximation, the lower MSE_0, L_0 and $\tilde{L}_0$ will typically be. To ensure low minimum error, we will often choose each atomic cell to be very small initially, with the implication that D will be very large (the value D can be considered, in a sense, as analogous to the value S in the discrete case). If D becomes too large, however, we may be in the position where applying an RL algorithm such as SARSA *directly* to the atomic cells becomes computationally intractable. In such a case we can employ PASA just as in the discrete case. All of our analysis in Sections 3.1 and 3.2 again holds. This allows us to derive a similar result to Theorem 12 above, however instead of guaranteeing arbitrarily low MSE, L or $\tilde{L}$, we instead guarantee MSE, L or $\tilde{L}$ which is arbitrarily close to the baseline values of MSE_0, L_0 or $\tilde{L}_0$.

Whilst the introduction of PASA does not remove the need to *a priori* select a discretisation of the state space, it gives us freedom to potentially choose a very fine discretisation (and, moreover, to choose a discretisation which is largely arbitrary) without necessarily demanding that we provide an underlying RL algorithm with a correspondingly large number of weights. This being the case, the application and advantages of PASA remain very similar to the case of finite state spaces.[37]

We will now formalise these ideas. Assume that the continuous state space $\mathcal{S}$ is a compact subset of d-dimensional Euclidean space $\mathbb{R}^d$ (noting that an extension of these concepts to more general state spaces is possible). We continue to assume that $\mathcal{A}$ is finite. Consistent with Puterman (2014), we redefine π, P and R as policy, transition and reward kernels respectively. As per our discussion above we assume that a preprocessing step maps every state in the continuous state space $\mathcal{S}$ to an element of a finite set $\mathcal{D}$ (a discrete approximation of the agent's input) of size D, via a mapping $m : \mathcal{S} \to \mathcal{D}$.

We will use $\Psi^{(t)}$ to represent the distribution of s at a particular time t given some starting point $s^{(1)}$. Given any fixed policy kernel π, we have a Markov chain with continuous state space. Provided that the pair (π, P) satisfies certain mild conditions we can rely on Theorem 13.0.1 of Meyn and Tweedie (2012) to guarantee that a stable distribution $\Psi^{(\infty)}$ exists, to which the distribution $\Psi^{(t)}$ will converge. We order the elements of $\mathcal{D}$ in some arbitrary manner and label each element as d_i $(1 \leq i \leq D)$. We now re-define the stable state probability vector ψ for the continuous case such that:

$$\psi_i = \int_{m^{-1}(d_i)} d\Psi^{(\infty)}(s).$$

It should be evident that all of the results in Sections 3.1 and 3.2 can be extended to the continuous case, with each result stated with reference to atomic cells, as opposed to

37. We can provide a more concrete example. We might imagine a bounded d-dimensional continuous state space. Assuming no prior knowledge of the problem, we assign states to atomic cells according to a uniform d-dimensional tiling. In order to ensure low values of MSE_0, L_0 and $\tilde{L}_0$, we elect to use extremely small tiles. However, as a result of the tiles being small, D—the number of individual tiles—is so large that SARSA cannot be applied directly. Hence we introduce PASA, in which case the problem becomes computationally tractable, and we have the benefit of the theoretical guarantees offered in this subsection.

individual states. We similarly redefine h as follows, where $\mathcal{I} \subset \mathcal{D}$:

$$h(\mathcal{I}, \pi) = \sum_{d_i \in \mathcal{I}} \psi_i.$$

The definition of δ-deterministic remains unchanged for π, as does the definition of δ_π. For the definitions of δ-deterministic for P and R, the definition of δ-constrained, and the definitions of δ_P, δ_R and $\delta_{\mathcal{I}}$, we redefine each of these in the obvious way with reference to atomic cells (and, as appropriate, $\Psi^{(\infty)}$ and π) as opposed to individual states.

Each scoring function in the continuous case can be defined with reference to an arbitrary weighting $w(s, a_j)$. As in the discrete case, however, we will need to assume, for our result, that either $w(s, a_j) = \pi(a_j|s)d\Psi^{(\infty)}(s)$ (for MSE) or $w(s, a_j) = \tilde{w}(m(s), a_j)d\Psi^{(\infty)}(s)$ (for L and $\tilde{L}$) for all s and j. In this context $\tilde{w} : \mathcal{D} \times \mathcal{A} \to [0, 1]$ is defined as an arbitrary function which must satisfy, in addition to $0 \leq \tilde{w}(d_i, a_j) \leq 1$ for all i and j, the constraint $\sum_{j'=1}^{A} \tilde{w}(d_i, a_{j'}) \leq 1$ for all i. Note that this definition of $\tilde{w}$ implies that the weighting is a constant function of s over each atomic cell. In general π may not satisfy this final constraint, however in practice usually we would expect to have $\pi(a_j, s') = \pi(a_j, s)$ for all s and s' such that $s \in d_i$ and also $s' \in d_i$ for some d_i. We can now generate an analogue to Theorem 12 for the case of continuous state spaces.

Theorem 15 *For a particular instance of the triple (P, R, π), suppose that each $|R(s, a_j)|$ for all s in $\mathcal{S}$ and for all j is bounded by a single constant. We let R_m (another constant) denote the supremum of $|\mathrm{E}\left(R(s, a_j)\right)|$ for all j and $s \in \mathcal{S}$. For any subset $\mathcal{I} \subset \mathcal{D}$, if $X \geq |\mathcal{I}|\lceil \log_2 D \rceil$ and $\min\{\psi_i : d_i \in \mathcal{I}\} > 1 - h(\mathcal{I}, \pi)$ then, for all $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ there exists T, ϑ and a parameter vector ς such that, provided $t \geq T$, with probability equal to or greater than $1 - \varepsilon_2$ the VF estimate $\hat{Q}^{(t)}$ generated by SARSA-P (using the PASA algorithm parametrised by X, ϑ and ς) will be such that:*

1. *If $w(s, a_j) = \pi(a_j|s)\, d\Psi^{(\infty)}(s)$ then:*

$$\mathrm{MSE} - \mathrm{MSE}_0 \leq \left(4(1-h) + \frac{33\delta_{\mathcal{I}}}{1-\gamma}\right)\frac{R_\mathrm{m}^2}{(1-\gamma)^2} + \varepsilon_1;$$

2. *If $w(s, a_j) = \tilde{w}(m(s), a_j)\, d\Psi^{(\infty)}(s)$ for an arbitrary function $\tilde{w}$ satisfying $0 \leq \tilde{w}(d_i, a_j) \leq 1$ and $\sum_{j'=1}^{A} \tilde{w}(d_i, a_{j'}) \leq 1$ for all i and j then:*

$$L - L_0 \leq \frac{4(1-h) + 29(1 - (1-\delta_P)(1-\delta_\pi))}{(1-\gamma)^2} R_\mathrm{m}^2 + \varepsilon_1;$$

and:

$$\tilde{L} - \tilde{L}_0 \leq \left(2(1-h) + \gamma^2\left(1 + 2(1-\delta_P)(1-\delta_\pi) - 3(1-\delta_P)^2(1-\delta_\pi)^2\right)\right)\frac{2R_\mathrm{m}^2}{(1-\gamma)^2} + \varepsilon_1.$$

The proof is similar to the proof of Theorem 12 and can be found at Appendix A.2. It should be clear that Remark 13 also applies, without significant change, to Theorem 15. A further theorem equivalent to Theorem 14 can also be generated in much the same way. Again the bounds (given, in particular, the details of the proof) are unlikely to be tight.

Note that each bound in Theorem 15 is, for technical reasons (see Appendix A.2), slightly different from the discrete case. However the implications of each bound are largely equivalent to the those in the discrete case. For example, a key implication of Theorem 15 is that, if $\mathcal{I}$ exists such that $\mathcal{I}$ consists of a small number of atomic cells and $h(\mathcal{I}, \pi)$ is large, we will be able to obtain arbitrarily low values for all three error functions compared to the baseline initially imposed by our mapping m. Similar to the discrete case this is of course implicitly subject to, in part, the values $\delta_{\mathcal{I}}$, δ_P and δ_π, depending on the scoring function of interest. The values of $\delta_{\mathcal{I}}$ and δ_P in the continuous case, defined, as they are, with reference to $\mathcal{D}$, $\Psi^{(\infty)}$ and π, are likely to depend on the continuity properties of P and π, as well as the size of the atomic cells. It may be possible to derive parts of Theorem 12 as a corollary to the result just outlined, however additional terms which appear in the continuous case bounds imply that this is not generally the case, at least insofar as the results have been stated here.

3.5. Application of Theorems 12 and 14 to Some Specific Examples

We can start to illustrate the consequences of Theorems 12 and 14 by examining some concrete examples. Again we are limiting our discussion to policy evaluation, though some of our observations will have consequences for policy improvement as well. Examples 1, 2 and 5 will show situations where we expect our results to guarantee that, for many policies which we might encounter, VF error can be substantially reduced by using PASA. Examples 3 and 4 will provide situations where our results will not be able to provide useful bounds on error (and by extension we cannot expect introducing PASA to have a meaningful impact). There is potential to extend our analysis far beyond the brief examples which we cover here.

Our first four examples are based on a "Gridworld" environment—see, for example, Sutton and Barto (2018)—although the principles we explore in each case will apply more generally. The agent is placed in a square $N \times N$ grid, where each point on the grid is a distinct state (such that $S = N^2$). It can select from four different actions: up, down, right and left. If the agent takes an action which would otherwise take it off the grid, its position remains unchanged. Certain points on the grid may be classed as "walls": points which, if the agent attempts to transition into them, its position will also remain unchanged.

Example 1 *Suppose that the agent can only occupy a small number n of points within a grid where N is very large (for example as a result of being surrounded by walls). See Figure 2a. Assume* we don't know in advance which points the agent can occupy. *Immediately we can use the Theorem 12 to guarantee for a particular P that, if $X \geq n \log_2 N^2$, we will have arbitrarily low VF error (for any of the three scoring functions subject to the stated conditions for w and subject, for $\tilde{L}$, to suitable constraints being placed on R) for any policy if we employ a suitably parametrised PASA algorithm. By employing Theorem 14 instead, we can extend this guarantee to a single PASA algorithm for all possible environments generated according to a suitably defined prior for P (one for which the number of accessible states will not exceed n).*

This example, though simple, is illustrative of a range of situations a designer may be presented with where only a small proportion of the state space is of any importance to the agent's problem of maximising performance, but either exactly which proportion is unknown, or it is difficult to tailor an architecture to suitably represent this proportion.

As we illustrate in the next example, however, environments where there are no such constraints can be equally good potential candidates. This is because many (perhaps even most) commonly encountered environments have a tendency to "funnel" agents using deterministic or near-deterministic policies into relatively small subsets of the state space.

Example 2 *Consider a grid with no walls except those on the grid boundary. Suppose that π is deterministic (such that the same action is always selected from the same point in the grid) and selected uniformly at random. Starting from any point in the grid, the average number of states the agent will visit before revisiting a state it has already visited (and thereby creating a cycle) will be loosely bounded above by 5. This follows from considering a geometric distribution with parameter $p = 1/4$, which is the probability of returning to the state which was just left. The bound clearly also applies to the number of states in the cycle the agent ends up entering, and is true for any N.*

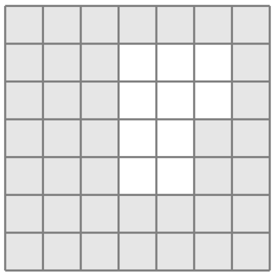
Hence, for any value of N, provided $X \geq 5 \log_2 N^2$, we can use Theorem 12 to guarantee that MSE, L and $\tilde{L}$ will all be arbitrarily low with a high probability (in the case of $\tilde{L}$ this is also conditional on the reward function R having low variance). Even if instead, for example, π is δ-deterministic where δ is small, the above arguments imply that the agent is still likely to spend extended periods of time in only a small subset of the state space.

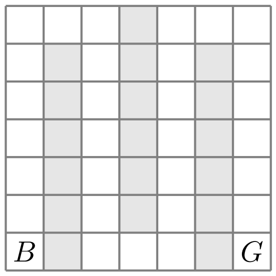
PASA may fail to provide an improvement, however, where the nature of an environment and policy is such that the agent must consistently navigate through a large proportion of the state space, as illustrated by the next two examples. Note that, notationally, $g(x) = \Omega(f(x))$ is equivalent to $f(x) = O(g(x))$.

Example 3 *Suppose that the same environment in Example 2 now has a single "goal state". Whenever the agent takes an action which will mean it would otherwise enter the goal state, it instead transitions uniformly at random to another position on the grid and receives a reward of 1. All other state-action pairs have a reward of zero.*

Suppose a policy is such that the average reward obtained per iteration is β (which must of course be less than 1). Every state must then have a lower bound on the probability with which it is visited of $\beta/(S-1)$, since every state must be visited at at least the same rate as the goal state, divided by all states excluding the goal state. Accordingly, for any subset $\mathcal{I}$ of the state space of size n, we must have $h \leq 1-(S-1-n)\beta/(S-1) = n\beta/(S-1)+(1-\beta)$. Hence h is constrained to be small if β is large (i.e., if the policy performs well) and n is small compared to S. Accordingly this environment would not be well suited to employing PASA.

Example 4 *Suppose that the same environment in Example 3 now has its goal state in the bottom right corner of the grid, and that, instead of transitioning randomly, the agent is sent to the bottom left corner of the grid (the "start state") deterministically when it tries to enter the goal state (again receiving a reward of 1). Suppose that $N = 3 + 4k$ for some integer $k \geq 0$, and also that walls are placed in every second column of the grid as shown in Figure 2b (with "doors" placed in alternating fashion at the top and bottom of the grid). Clearly an optimal policy in this case will be such that for all subsets $\mathcal{I}$ of the state space of*


(a) Example of an environment type well suited to PASA (assuming much larger grid dimensions than illustrated). Only a comparitively small number of states (not known in advance to the designer) can be accessed by the agent.



(b) Example of an environment type *not* well suited to PASA (assuming much larger grid dimensions than illustrated). G is the goal state and B is the state transitioned to when the agent attempts to enter the goal state. The placement of the walls implies that an optimal policy will force the agent to regularly visit $O(S)$ states.

Figure 2: Example Gridworld diagrams for $N = 7$ (the case $N = 7$ is convenient to illustrate diagrammatically however our interest is in examples with much larger N). Each white square is a point in the grid. Grey squares indicate walls.

size n, $h < 2n/S$. Hence we would require $X = \Omega(S)$ in order to be able to apply Theorem 12 to the optimal policy.

In general, to find examples for deterministic P where this is the case requires some contrivance, however. If we removed the walls, for example, an optimal policy would be such that SARSA-P would obtain a VF estimate with arbitrarily low error provided $X(S) \geq f(S)$ for a particular function f which is $O(\sqrt{S}\log_2 S)$. This is true for any fixed start and goal states.

Whilst not all environments are good candidates for the approach we've outlined, very many commonly encountered environment types would appear to potentially be well suited to such techniques. To emphasise this, interestingly, even environments and policies with no predefined structure at all have the property that an agent will often tend to spend most of its time in only a small subset of the state space. Our next example will illustrate this. It is a variant of the GARNET problem.[38]

Example 5 *Consider a problem defined as follows. Take some $\delta > 0$. We assume: (1) P is guaranteed to be δ-deterministic, (2) P has a uniform* prior *distribution, in the sense that, according to our prior distribution for P, the random vector $P(\cdot|s_i, a_j)$ is independently distributed for all (i, j) and each random variable $P(s_{i'}|s_i, a_j)$ is identically distributed for all (i, j, i'), and (3) π is δ-deterministic.*

38. Short for "generic average reward non-stationary environment test-bench." This is in fact a very common type of test environment, though it does not always go by this name, and different sources define the problem slightly differently. See, for example, Di Castro and Mannor (2010).

Generally, condition (2) can be interpreted as the transition function being "completely unknown" to the designer before the agent starts interacting with its environment. It is possible to obtain the following result in relation to environments of this type:

Lemma 16 *For all $\varepsilon_1 > 0$, $K > 1$ and ε_2 satisfying $0 < \varepsilon_2 < K - 1$, there is sufficiently large S, sufficiently small $\delta > 0$ and sufficiently small $\delta' > 0$ such that, for an arbitrary policy π, with probability no less than $1 - 1/(K - \varepsilon_2 - 1)\ln S$ conditioned upon the prior for P, we will have a set $\mathcal{I} \subset \mathcal{S}$ such that $|\mathcal{I}| \leq K\sqrt{\pi S/8}\ln S$, $h(\mathcal{I}, \pi) > 1 - \varepsilon_1$ and $\min_{i\in\mathcal{I}} \psi_i > 2(1 - h) + \delta'$.*

Details of the proof are provided in Appendix A.3. The result is stated in the limit of large S, however numerical analysis can be used to demonstrate that S does not need to be very large before there is a high probability that $\mathcal{I}$ exists so that $|\mathcal{I}|$ falls within the stated bound. Now, with the help of Theorem 14 we can state the following result in relation to Example 5, which can be obtained by simply applying Theorem 14 in conjunction with Lemma 16. (Since we are relying on Theorem 14 we must assume we are using a version of PASA which incorporates the changes in points 3 and 4 of Appendix A.5.) In particular we note that, as a result of Lemma 16, for all S, we can, for any $\varphi > 0$, select δ and ξ such that we will have $\sum_{i:s_i\notin\mathcal{I}} \psi_i \leq \varphi$ and $s_i \in \mathcal{I} \Rightarrow \psi_i \geq \xi > 2\varphi$. We can also, for all S, select δ such that each of δ_P, δ_π and $\delta_\mathcal{I}$ is arbitrarily close to zero, where for $\delta_\mathcal{I}$ we rely on the observation made in the last paragraph before Theorem 12. As a result we will have:

Corollary 17 *Suppose that, for Example 5, the prior for R is such that R is guaranteed to satisfy $|\mathrm{E}\left(R(s_i, a_j)\right)| \leq R_\mathrm{m}$ for all i and j, where R_m is a constant. Then for all $\varepsilon_1' > 0$, $\varepsilon_1'' > 0$, $K > 1$ and ε_2 satisfying $0 < \varepsilon_2 < K-1$, there is sufficiently large S and sufficiently small δ such that there exist values T, ς and ϑ such that, provided $X \geq K\sqrt{\pi S/8}\ln S\lceil\log_2 S\rceil$, SARSA-P will (with a PASA component which incorporates the changes in points 3 and 4 of Appendix A.5 and which is parametrised by ς, ϑ and X) generate for $t > T$, with probability no less than $(1-1/(K-\varepsilon_2-1)\ln S)(1-\varepsilon_1'')$ conditioned upon the prior for P, a VF estimate with $\mathrm{MSE} \leq \varepsilon_1'$ and $L \leq \varepsilon_1'$. If the prior for R is also such that R is guaranteed to be δ-deterministic, then the same result will hold for $\tilde{L}$.*

What the result tells us is that, even where there is no apparent structure at all to the environment, an agent's tendency to spend a large amount of time in a small subset of the state space can potentially be quite profitably exploited. Our experimental results (where we examine an environment type equivalent to that described by Example 5) will further confirm this.

The bound on X provided is clearly sub-linear to S, and may represent a significant reduction in complexity to the standard tabular case when S starts to take on a size comparable to many real world problems. (Whilst in practice, the key determinant of X will be available resources, the result of course implies the potential to deal with more complex problems—or complex representations—given fixed resources.) Furthermore the bound on X in Corollary 17 appears to be a loose bound and can likely be improved upon. Our result as stated only pertains to policies generated with no prior knowledge of P, however we can see that, for any R which is independent of P, for any policy π and for any $\varepsilon > 0$, an optimal policy π^* will be such that:

$$\Pr(\min\{K : |\mathcal{I}| = K, h(\mathcal{I}, \pi^*) > 1 - \varepsilon\} < x) \geq \Pr(\min\{K : |\mathcal{I}| = K, h(\mathcal{I}, \pi) > 1 - \varepsilon\} < x).$$

So Corollary 17 will also apply to optimal policies.

An implication of the result is that, assuming our condition on X holds, then SARSA-P will have arbitrarily low VF error with arbitrarily high probability for sufficiently large S and sufficiently small δ. Our result was not stated in quite this way for a technical reason. Namely, whilst Corollary 17 does not apply to SARSA-F, *fixed* state aggregation will *also* tend to have zero error if $X \neq O(\sqrt{S} \ln S)$, S is sufficiently large, δ is sufficiently small and all our other assumptions continue to hold. This is because the probability of more than one state in a set $\mathcal{I}$ falling into a single cell tends to zero if the number of cells grows at a faster rate than the size of $\mathcal{I}$. However, since the arrangement of the states amongst cells will be uniformly random (since Ξ is arbitrary and P is uniform), then for SARSA-F to have only one element of the set $\mathcal{I}$ to which Lemma 16 refers in each cell will require both that the set $\mathcal{I}$ exists *and* that each state in $\mathcal{I}$ happens to fall into its own cell. So SARSA-F will always have an additional (generally very small, and therefore highly detrimental) factor contributing to the probability of arbitrarily low error. In effect, the probability of each element of a set $\mathcal{I}$ of states falling into a unique fixed cell increases slowly, so that any guarantee pertaining to SARSA-F can only made with much lower probability than for an equivalent guarantee for SARSA-P as S is increased.

There is some scope to extend Lemma 16 beyond uniform transition function priors. However the theoretical complexity involved with generating formal results can increase significantly as we add more complexity to our prior for P. Furthermore, in some special cases, knowledge about the value of P (i.e. a non-uniform prior) can mean our results will not apply. If we know, for example, that $P(s_{i+1}|s_i, a_j) = 1$ for all $1 \leq i \leq S - 1$ and all j, and $P(s_1|s_S, a_j) = 1$ for all j, then Lemma 16 clearly doesn't hold (much like in Example 4). There are some slightly more general sets of priors for which the result, or parts of the result, can be shown to hold. These minor extensions can be obtained, for example, by using the notion of Schur convexity. This is addressed in Appendix A.4.

Before considering our experimental results, we can summarise this subsection by observing, informally, that there appear to be two "pathologies" which P might suffer from which will result in PASA being likely to have little impact. The first is if P is subject to large degrees of randomness resulting in the agent being sent to a large proportion of the state space, as illustrated by Example 3 and the importance of the value δ in Example 5.[39] The second is if the prior for P has an exaggerated tendency to direct the agent through a large proportion of the state space, as illustrated by Example 4 and the example provided in the previous paragraph. These observations will be reinforced by our experimental results.[40]

39. High degrees of randomness in π can also create an issue, however in practice agents seeking to exploit learned knowledge of the environment will typically have near-deterministic policies.

40. Our examples have all focussed on discrete state spaces. Whilst we will not explore the details, Examples 1, 3 and 4 have ready continuous state space analogues. The same is true of Example 2 provided that certain assumptions hold around the continuity of P and π (so that the agent's policy is such that it is likely to return to a state close to one it has already visited). The implications which arise from each example in a continuous state space setting reflect closely the discrete state space case.

Table 1: Summary of main theoretical results.

Result	Description of result (omitting conditions)	Requires
Proposition 4	PASA parameters exist which guarantee, given a single pair (P, π), that Ξ will converge.	nil
Proposition 5	The convergence point of Ξ in Proposition 4 will be such that the agent will spend a large amount of time in singleton cells.	Proposition 4
Proposition 7	Equivalent to Propositions 4 and 5, however applies to a set of pairs (P, π), and the statement regarding the limit Ξ is slightly weaker.	nil
Theorem 12	The convergence point of Ξ in Proposition 4 is such that, given a single triple (P, R, π), the values of MSE, L and $\tilde{L}$ will be bounded. These bounds, under suitable conditions, will be arbitrarily close to zero.	Propositions 4 and 5
Theorem 14	Equivalent to Theorem 12, however the bounds on MSE, L and $\tilde{L}$ will apply to all triples (P, R, π) in a set.	Proposition 7
Theorem 15	Extension of Theorem 12 to continuous state spaces (we claim the existence of an equivalent extension to 14).	Propositions 4 and 5[a]
Lemma 16	Given a uniform prior for P, an agent will spend an arbitrary large proportion of the time in a subset of the state space containing $O(\sqrt{S}\ln S)$ states with high probability.	Results in Section A.3 (Lemmas 18 and 19)
Corollary 17	Given the same conditions as Lemma 16, MSE, L and $\tilde{L}$ will be arbitrarily low provided $X \geq K\sqrt{\pi S/8}\ln S\lceil\log_2 S\rceil$ for a constant K as S becomes large.	Lemma 16 and Theorem 14

[a] The result in fact requires more general versions of Propositions 4 and 5 applicable to continuous state spaces. Such extensions are straightforward in the context of our arguments however, and we do not provide these more general results formally.

4. Experimental Results

Our main objective in this section will be to determine via experiment the impact that PASA can have on actual performance. This will help clarify whether the theoretical properties of PASA which guarantee decreased VF error in a policy evaluation setting will translate to improved performance in practice. Whilst our principle intention here is to validate our theoretical analysis and demonstrate the core potential of PASA, more wide-ranging experimental investigation would comprise an interesting topic for further research.

We have examined three types of problem: (a) a variant of the GARNET class of problem (substantially equivalent to Example 5), (b) a Gridworld problem and (c) a logistics problem. All three of the environment types we define in terms of a prior on (P, R), which allows for some random variation between individual environments. In all of our experiments we compare the performance of SARSA-P to SARSA-F (both with the same number X of cells). In some cases we tested SARSA-F with more than one state aggregation, and in most cases we have also tested, for comparison, SARSA with *no* state aggregation.

We will see that in all cases (with the exception of one isolated result which appears to have been affected by experimental noise) SARSA-P exhibits better performance than SARSA-F, in some cases substantially so (for the GARNET problem we also demonstrate that, as predicted, SARSA-P results in lower MSE for randomly generated fixed policies). We will also see that in some key instances SARSA-P outperforms SARSA with no state aggregation. The parameters[41] of PASA were kept the same *for all environment types*, with the exceptions of X_0 and X (with X being changed for SARSA-F as well). The value of X_0 was always set to $X/2$. The parameters used are shown in Table 2.

Furthermore (as summarised in Table 5) SARSA-P requires only marginally greater computational time than SARSA-F, consistent with our discussion in Section 3.1. Whilst we have not measured it explicitly, the same is certainly true for memory demands.

Each experiment was run for 100 individual trials for both SARSA-P, SARSA-F and (where applicable) SARSA with no state aggregation, using the same sequence of randomly generated environments. Each trial was run over 500 million iterations.

For our experiments some minor changes have been made to the algorithm SARSA-P as we outlined it above (that is, changes which go beyond merely more efficiently implementing the same operations described in Algorithms 1 and 2). The changes are primarily designed to increase the speed of learning. These changes were not outlined above to avoid adding further complexity to both the algorithm description and parts of the theoretical analysis, however they in no way materially affect the manner in which the algorithm functions, and none of the changes affect the conclusions in Section 3.1. The changes are outlined in Appendix A.5. Whilst SARSA-P continues to outperform SARSA-F in the absence of these changes, without them SARSA-P tends to improve at a slower rate, making the difference observable from experiment less pronounced.

Unless otherwise stated, for SARSA-F, the state aggregation was generated arbitrarily, subject to cell sizes being as equally sized as possible. For SARSA-P, to generate the initial partition Ξ_0 we ordered states arbitrarily, then, starting with a partition containing a single

41. In this section we use the word "parameter" to refer to the handful of high-level values which can be optionally tuned for either an agent or environment (sometimes called "hyper-parameters"), such as η, ς, ϵ, γ and ϑ. It will not be used, for example, to refer to the matrix of values θ stored by SARSA (also sometimes referred to in the literature as "parameters").

cell containing every state, we recursively split the cells with indices indicated by the first X_0 elements of the following sequence:

$$(1, 1, 2, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7, 8, \dots).$$

Doing so results in roughly equally sized cells, which in practice, assuming the absence of any specific information regarding the environment, is preferable.

Table 2: PASA parameters used in all experimental domains.

Parameter	Type[a]	Description	Value(s)
ϵ	S	ϵ-greedy parameter	0.01, 0.001[†]
γ	S	Discount rate	0.98
η	S	SARSA step size	3×10^{-4}
X	F	Number of cells	Differs by experiment[b]
X_0	P	Number of base cells	$X/2$
ς	P	PASA step size	1×10^{-8}
ν	P	PASA update frequency	50,000
ϑ	P	ρ update tolerance	0.9

[a] The values in this column reflect whether the parameter relates to the SARSA algorithm (S) (in which case the parameter applies to all three of the SARSA-P, SARSA-F and SARSA with no state aggregation experiments), both SARSA-F and SARSA-P (F), or the PASA algorithm only (P).
[b] Details can be found by referring to the discussion of each experiment.
[†] This parameter value was tested for the MSE experiments for the GARNET problem, but was not used for any of the performance experiments.

4.1. GARNET Problems

The prior for P for this environment is the same as that described in Example 5, however for the purposes of this experiment we have taken $\delta = 0$. We selected $s^{(1)}$ at random. Note that the transition function for a particular π for environments defined in such a way will not necessarily be irreducible. The prior for R is such that $R(s_i, a_j) = S$ with probability ζ/S for all (i, j) and is zero otherwise, where $\zeta > 0$ is a parameter of the environment. The way we have defined R is such that, as S increases or ζ decreases, state-action pairs with positive reward become more and more sparsely distributed (and therefore more difficult for the agent to find). However the expected reward associated with selecting actions uniformly at random remains constant for different values of S provided ζ remains constant.

We ran two sets of experiments. In the first set we selected π randomly and held it fixed, allowing us to measure comparative MSE for SARSA-P and SARSA-F. Each π generated was ϵ-deterministic (see Section 3.3). Specifically, with probability $1 - \epsilon$, a deterministic (however initially randomly selected) action is taken and, with probability ϵ, an action is selected uniformly at random. In the second set of experiments we allowed π to be updated and measured overall performance for SARSA-P, SARSA-F and SARSA with no state aggregation. For the second set in every iteration the policy π was selected to be ϵ-greedy with respect to the current VF estimate. We ran each set of experiments with different values of S and ζ as described in Table 3.

Table 3: GARNET experiment parameters.

Parameter	Description	Value(s)
S	Number of states	250, 500, 1K, 2K*, 4K*, 8K*
A	Number of actions	2
ζ	Expected state-action pairs with $R_{ij} = S$	30, 3†
X	Number of cells[a]	70, 100, 140, 200*, 280*, 380*

[a] This parameter pertains only to SARSA-P and SARSA-F. The individual values were arrived at heuristically, however with the guidance of Lemma 16.

* These parameter values were not tested for the MSE experiments.

The results of the experiments relating to MSE can be seen in Table 4. These results demonstrate that, as the results in Section 3 predict, PASA has the effect of reducing MSE for a fixed policy. The effect of applying PASA becomes more pronounced as ϵ is reduced, which is also consistent with our predictions. We obtained similar results (not reproduced) for $\zeta = 3$. (Note that, even for relatively low values of S, calculating MSE can quickly become prohibitively expensive from a computational standpoint, hence why we have limited these experiments to $S \leq 1{,}000$.)

In relation to actual performance, Figures 3 demonstrates that SARSA-P outperforms SARSA-F in every experiment (with the exception of one apparent anomaly, which we assume is due to noise), and in many cases by a substantial margin. Furthermore, as S increases and ζ decreases, that is, as the problem becomes more complex requiring the agent to regularly visit more states, the disparity in performance between SARSA-P and SARSA-F widens. Part of the reason for this widening may be because the VF estimates of SARSA-F will experience rapid decay in accuracy as cells are forced to carry more states, whereas SARSA-P (for the reasons we examined in Section 3.5) will be less affected by this issue.[42]

We have included, for comparison, the performance of SARSA with no VF approximation. In some cases SARSA-P compares favourably with the case where there is no state aggregation, although as S becomes large SARSA with no state aggregation starts to out-perform SARSA-P. The GARNET environment represents unique challenges for *any* RL algorithm which employs VF approximation (since every single state has behaviour completely independent of all other states), such that the performance increase we see for SARSA with no VF approximation compared to SARSA-P may not be completely surprising. SARSA-P, of course, requires significantly fewer weights, and, in general, as S gets very large using SARSA with no state aggregation may become impossible. This issue is highlighted more strongly in the third of the three environments, considered in Section 4.3. This inability of SARSA to scale is, of course, a key motivator behind proposing SARSA-P. Given this, the comparison between SARSA-P and SARSA-F may be more instructive in

42. An examination of π over individual trials (not shown) suggests that SARSA-P has a stronger tendency to converge than SARSA-F (i.e. in spite of the fact that convergence cannot be guaranteed in general for this problem). This may be in part because policies which are clearly represented tend to be associated with a higher value, which in turn makes such policies more likely to be selected and therefore remain stable.

Table 4: Comparative MSE of SARSA-P and SARSA-F.

Parameter settings			Average $\sqrt{\text{MSE}}$[a]		
S	ζ	ϵ	SARSA-P	SARSA-F	% reduction[b]
250	30	0.01	0.48 ($\pm$0.012)	0.825 ($\pm$0.022)	**41.8%**
500	30	0.01	0.4 ($\pm$0.014)	0.584 ($\pm$0.017)	**31.5%**
1,000	30	0.01	0.337 ($\pm$0.008)	0.477 ($\pm$0.011)	**37%**
250	30	0.001	0.438 ($\pm$0.012)	0.977 ($\pm$0.031)	**55.2%**
500	30	0.001	0.325 ($\pm$0.016)	0.673 ($\pm$0.024)	**51.7%**
1,000	30	0.001	0.259 ($\pm$0.007)	0.554 ($\pm$0.016)	**67.9%**

[a] Average over 100 independent trials. Figures are based on the average MSE over the final fifth of each trial. The MSE was calculated by first calculating the true value function for each random policy π. Confidence intervals (95%) for $\sqrt{\text{MSE}}$ are shown in brackets.
[b] The extent to which average $\sqrt{\text{MSE}}$ was reduced by introducing PASA, expressed as a percentage.

this instance as the two algorithms share the same number of cells as a function of S (as well as similar—as discussed in Section 3.1—computational complexity).

4.2. Gridworld Problems

We examined two different Gridworld-type problems. As in Examples 1 to 4 in Section 3.5 we confined the agent to an $N \times N$ grid, where the agent can move up, down, left or right. We singled out r positions on the grid, which we call *reward positions*. These positions will, if the agent attempts to move into them, result in a reward of 1, and the agent will transition to another (deterministic) position on the grid, which we call a *start position*. Each reward position transitions to its own unique start position. The problem is defined such that the reward and start positions are uniformly distributed over the grid (subject to the constraint that each start and reward position cannot share its location with any other start or reward position). Provided r is relatively small compared to N^2, solutions are likely to involve a reasonably complex sequence of actions.

We chose $N = 32$ (so that $S = 1{,}024$). We ran two separate experiments for $r = 8$ and $r = 24$. We also ran a third experiment, again with $r = 24$, however in this case we altered P to be non-deterministic, specifically such that each reward position transitioned to a completely random point on the grid (instead of to its corresponding start position), in a rough analogue to Example 3.

It is common in RL experiments of this form for the designer to choose basis functions which in some way reflect or take advantage of the spatial characteristics of the problem. In our case we have chosen our X_0 base cells for SARSA-P in an arbitrary manner, as we deliberately do not want to tailor the algorithm to this particular problem type. We tested SARSA-F with two different state aggregations. One arbitrary (in line with that chosen for SARSA-P) and one with cells which reflected the grid-like structure of the problem, where states were aggregated into as-near-as-possibly equally sized squares.[43] Only results for the former are shown, as these were superior (possibly since equating proximal states

43. This type of state aggregation is commonly referred to as "tile coding". See, for example, Sutton and Barto (2018).

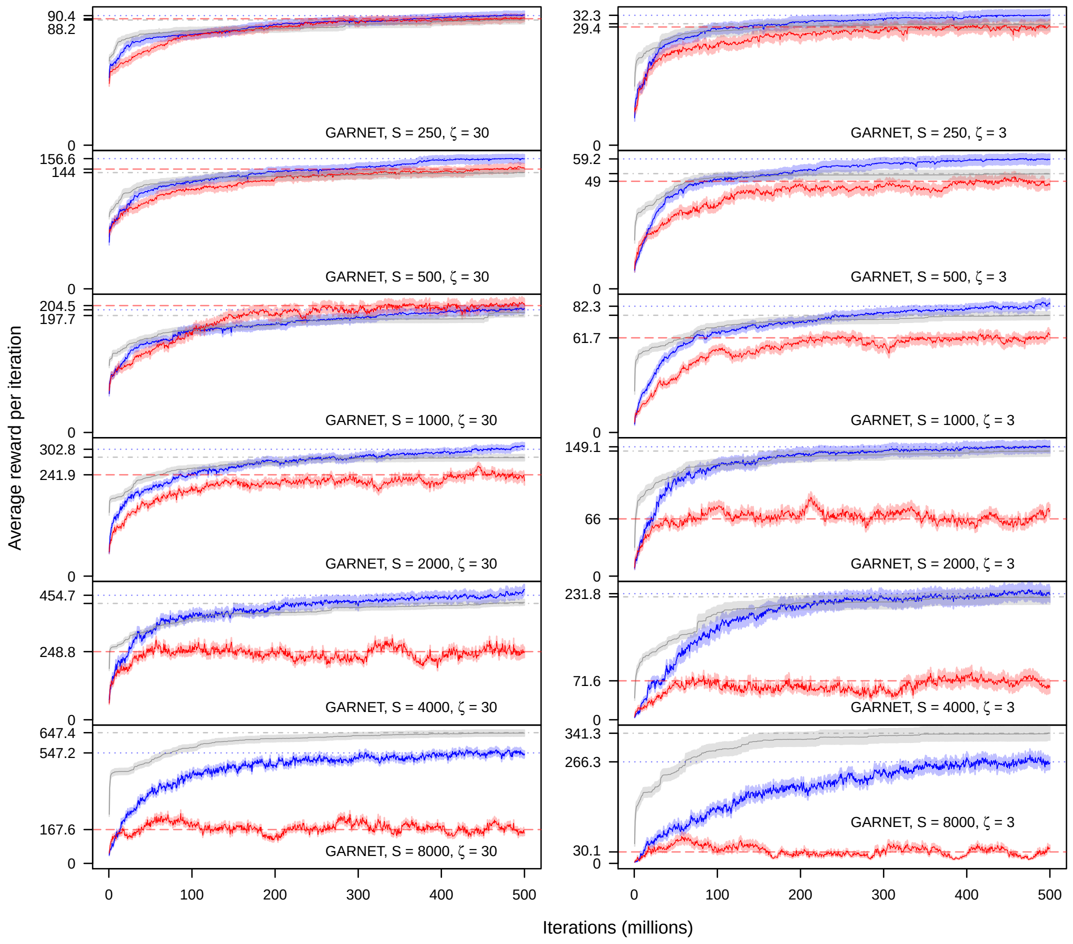


Figure 3: Comparative performance of SARSA-P (blue) and SARSA-F (red) as a function of t. Performance of SARSA with no state aggregation is also shown (grey). Horizontal lines indicate the average reward obtained over the final fifth of each trial. Notice that environments become progressively "more challenging" moving down or right through the figure. SARSA-P consistently outperformed SARSA-F (excluding the anomalous result for $S = 1{,}000$ and $\zeta = 30$, which appears to be affected by noise), and the disparity between the algorithms' performance increased as problem complexity increased. Slightly greater variance in reward obtained (for all three algorithms) can also be seen as problem complexity is increased.

can make the agent less likely to find rare short-but-optimal pathways through the state space). We also tested SARSA with no state aggregation. The algorithm parameter settings were $X_0 = 70$ and $X = 140$, with all other relevant parameters left unchanged from the GARNET problem settings.

The average performance of each algorithm as a function of time is shown in Figure 4. The results are significant insofar as SARSA-P has continued to outperform SARSA-F in a quite different, and more structured, setting than that encountered with the GARNET problem. This was despite some attempt to find an architecture for SARSA-F which was "tailored" to the problem (a process which, of course, implicitly requires an investment of time by the designer). The improved performance of SARSA-P was increased where $r = 8$. Hence the disparity in performance was even greater when r was set lower (i.e. when the problem requires more complex planning). Surprisingly, for this particular problem SARSA-P outperformed even SARSA with no state aggregation. We speculate that the reason for this is that the aggregation of states allows for a degree of generalisation which is not available to SARSA with no state aggregation.[44]

For the third experiment, consistent with our analysis in Section 3, SARSA-P's performance was well below its performance in the first experiment, despite the fact that policies exist in this environment which are likely to generate similar reward to that obtained in the first experiment. We would expect this drop in performance given this is not an environment well suited to using PASA (although the algorithm still performed far better than SARSA-F).

4.3. Logistics Related Problems

Many applications for RL come from problems in operations research (Powell, 2011; Powell et al., 2012). This is because such problems often involve planning over a sequence of steps, and maximising a single value which is relatively straightforward to quantify. Problems relating to logistics (e.g. supply chain optimisation) are a good example of this.

Our third environment involves a transport moving some type of stock from a depot to several different stores. We suppose that we have N stores, that each store (including the depot) has a storage rental cost (per unit stock held there), and that each store has a deterministic sales rate (the sales rate at the depot is zero). The agent has control of the transport (a truck), and also has the ability to order additional stock to the depot. There is a fixed cost associated with transporting from any given point to another. The following, therefore, are the agent's possible actions (only one of which is performed in each iteration): (a) ordering an additional unit of stock to the depot, (b) loading the truck, (c) moving the truck to one of the $N + 1$ locations (treated as $N + 1$ distinct actions), and (d) unloading the truck.

In what follows we will use $U(a, b)$, where $a < b$, $a \in \mathbb{R}$ and $b \in \mathbb{R}$ to denote a continuous uniform random variable in the range from a to b. In our experiment we have taken $N = 4$, capacities $= (12, 3, 4, 3, 6)$ (the first entry applies to the depot), transport cost $\sim$

44. More specifically, SARSA-P learns to associate states which have never (or have only very rarely) been visited with a slightly positive reward, due to its aggregation of many states over rarely visited areas of the state space. This encourages it to explore rarely-visited regions in favour of frequently visited regions with zero reward. SARSA with no state aggregation, in contrast, takes longer to revise its initial weights (in this case set to zero) due to the absence of generalisation between states.

$U(-1.2, -0.6)$, order cost $= -2$, sale revenue $= 7$, sales rate $= 1$ at all stores, and rent $= (U(-0.2, -0.05), U(-0.05, -0.01), U(-0.08, -0.03), U(-0.08, -0.01), U(-0.4, -0.001))$ (the first entry again applies to the depot). We stress that the randomness, where applicable, in these variables relates to the prior distribution of (P, R). There is no randomness once an instance of the environment is created. In arranging states into their initial cells for SARSA-P we have assumed we know nothing of the inherent problem structure (i.e. we selected an arbitrary cell arrangement). For SARSA-F we again trialled both arbitrary and tile coded state aggregations (where each stock level was divided into equal intervals; only the tile-coded results are shown, as these were superior, however the difference in performance was minimal).

We have kept the same agent parameters as in the GARNET and Gridworld problems, again with $X_0 = 70$ and $X = 140$. Note that the number of states which can be occupied is $13 \times 4 \times 5 \times 4 \times 7 \times 4 \times 2 =$ 72,800. The majority of these states are likely to be rarely or never visited, or are of low importance based on the intrinsic structure of the problem. Complicating the problem further, the input provided to the agent is a binary string made up of 18 digits, consisting of the concatenation of each individual integer input variable converted to a binary expansion (which is a natural way to communicate a sequence of integers). As a result we effectively have $S = 2^{18} =$ 262,144 (many of these states cannot be occupied by the agent, however the designer might be assumed to have no knowledge of the environment beyond the fact that each input consists of a binary string of length 18).

Over the 100 trials, SARSA-P obtained an average reward of 0.785 (which is close to optimal) compared to SARSA-F which was only able to obtain average reward of -0.019 (see Figure 4). Hence, for this problem, the disparity between the two algorithms was reasonably dramatic.

This is not a complex problem to solve by other means, despite the large state space.[45] However the example helps to further illustrate that SARSA-P can significantly outperform SARSA-F despite roughly equivalent computational demands and without requiring any prior information about the structure of the environment. Further experimentation could provide an indication as to how well SARSA-P performs on even more complex logistics problems, however the indications coming from this modest problem appear promising.

4.4. Comments on Experimental Results

Table 5 provides a summary of the experimental results. The table shows that, at a low computational cost, we are able to significantly improve performance by using PASA to update an otherwise naïve state aggregation architecture in a range of distinct problem types, and with minimal adjustment of algorithm parameters. The experiments where PASA had less impact were of a nature consistent with what we predicted in Section 3. Optimising parameters such as ς, ϵ or even the value of X might be expected to further increase the disparity between SARSA-P and SARSA-F.

It is interesting to consider the mechanics via which PASA appears able to increase performance, in addition to simply decreasing VF error. This is a complex question, and a rigorous theoretical analysis would appear to pose some challenges, however we still can attempt to provide some informal insight into what is occurring.

45. This is part of the reason why we have not tested SARSA with no state aggregation for this problem.

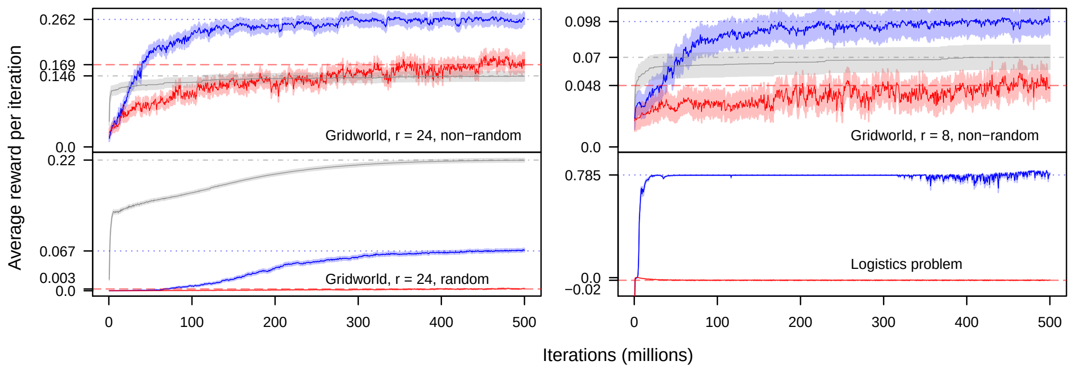


Figure 4: Comparative performance of SARSA-P (blue) and SARSA-F (red) as a function of t on Gridworld and logistics problems. Performance of SARSA with no state aggregation is also shown for Gridworld problems (grey). Horizontal lines indicate the average reward obtained over the final fifth of each trial. SARSA-P outperforms both SARSA-F and SARSA with no state aggregation by some margin on the Gridworld problem with deterministic start points. The disparity between SARSA-P and SARSA-F is greater when the number of available rewards is decreased. Introducing start points which are selected uniformly at random greatly diminishes the performance of SARSA-P (however it also diminishes the performance of SARSA-F). The logistics problem reveals a dramatic disparity in performance. SARSA-P consistently finds an optimal or near-optimal solution whilst SARSA-F fails to find a policy with average reward greater than zero.

SARSA-P is able to estimate the VF associated with a *current* policy with greater accuracy than SARSA-F. To some extent this is at the cost of less precise estimates of alternative policies, however this cost may be comparatively small, since adding additional error to an estimate which *already* has high error may have little practical impact. The fact that SARSA-F (unlike SARSA-P) is forced to share weights, even amongst only a small number of states (which is likely to be inevitable with *any* fixed linear approximation architecture, not just state aggregation, unless the designer makes strong initial assumptions regarding P and π), appears to rapidly decay VF estimate accuracy as problems become complex, and also makes the chattering described by Gordon (1996) particularly problematic, especially for complex environments. Observing individual trials across our experiments, it appears that SARSA-F does occasionally find strong policies, but that these are consistently lost as actions which deviate from this policy start to get distorted VF estimates.

The capacity of SARSA-P to place important states into very small cells is integral to its effectiveness. It potentially allows SARSA-P to perform a high precision search over localised regions of the policy space, whilst not being at a significant disadvantage to SARSA-F when searching (with much less precision) over the whole policy space. Furthermore, the addition of PASA creates a much greater tendency for strong policies, when they are discovered, to remain stable (though not so stable that the agent is preventing from exploring the potential for improvements to its current policy).

Finally, as we saw in the Gridworld environments, there may be instances where the advantages of *generalisation* which arise from function approximation may be leveraged by employing PASA, without suffering to the same extent the consequences which typically come from a lack of precision in the VF estimate.

5. Conclusion

One of the key challenges currently facing RL remains understanding how to effectively extend core RL concepts and methodologies (embodied in many of the classic RL algorithms such as Q-learning and SARSA) to problems with large state or action spaces. As noted in the introduction, there is currently interest amongst researchers in methods which allow agents to learn VF approximation architectures, in the hope that this will allow agents to perform well, and with less supervision, in a wider range of environments. Whilst a number of different approaches and algorithms have been proposed towards this end, what we've termed "unsupervised" techniques of adapting approximation architectures remain relatively unexplored.

We have developed an algorithm which is an implementation of such an approach. Our theoretical analysis of the algorithm in Section 3 suggests that, in a policy evaluation setting, there are types of environment (which are likely to appear commonly in practice) in which our algorithm—and potentially such methods more generally—can on average significantly decrease error in VF estimates. This is possible despite minimal additional computational demands. Furthermore, our experiments in Section 4 suggest that this reduction in VF error can be relied upon to translate into improved performance. In our view, the theoretical and experimental evidence presented suggests that such techniques are a promising candidate for further research, and that the limited attention they have received to date may be an oversight.

Besides exploring improvements to PASA, or even alternative implementations of the principles and ideas underlying PASA, we consider that some of the more interesting avenues for further research would involve seeking to extend the same techniques to problems involving (a) large, or potentially continuous, action spaces, (b) factored Markov decision problems,[46] and (c) partially observable Markov decision problems.[47] The principles we have explored in this article suggest no obvious barriers to extending the techniques we have outlined to these more general classes of problem.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable and insightful comments.

46. Factored MDPs—see, for example, Boutilier et al. (1995)—in essence allow components of the environment to evolve independently.
47. Partially observable MDPs—see, for example, Kaelbling et al. (1998)—encompass problems where the agent does not have complete information regarding its current state $s^{(t)}$.

Table 5: Summary of results of SARSA-P and SARSA-F comparative performance.

Experiment	S	X	Comments	Aver. reward per iter. [a]		% incr.	Aver. μs per iter.[b]		% incr.
				SARSA-P	SARSA-F		SARSA-P	SARSA-F	
GARNET	250	70	$\zeta = 30$	90	87.8	**2.5**%	2.64	2.51	5.2%
	500	100	$\zeta = 30$	155.9	143.3	**8.8**%	2.45	2.41	1.8%
	1,000	140	$\zeta = 30$	196.8	203.5	**−3.3**%	2.5	2.41	3.4%
	2,000	200	$\zeta = 30$	301.4	240.8	**25.2**%	2.52	2.45	2.7%
	4,000	280	$\zeta = 30$	452.4	247.7	**82.6**%	2.56	2.49	2.8%
	8,000	380	$\zeta = 30$	544.6	166.7	**226.6**%	2.57	2.5	2.9%
	250	70	$\zeta = 3$	32.2	29.2	**10**%	2.43	2.39	1.7%
	500	100	$\zeta = 3$	58.9	48.8	**20.7**%	2.42	2.41	0.7%
	1,000	140	$\zeta = 3$	81.9	61.3	**33.4**%	2.48	2.42	2.3%
	2,000	200	$\zeta = 3$	148.4	65.6	**126**%	2.49	2.44	2%
	4,000	280	$\zeta = 3$	230.7	71.2	**223.9**%	2.52	2.47	2%
	8,000	380	$\zeta = 3$	264.9	30	**782.8**%	2.65	2.65	0%
Gridworld	1,024	140	$r = 24$	0.261	0.168	**55**%	2.76	2.62	5.2%
	1,024	140	$r = 8$, random trans.	0.098	0.048	**105.1**%	2.64	2.62	0.7%
	1,024	140	$r = 24$	0.066	0.003	> **2K**%	2.57	2.51	2.4%
Logistic	> 262K	140		**0.785**	**−0.019**	n/a[c]	1.95	1.8	8.3%

[a] We take the average reward per iteration over the last fifth of each trial.

[b] An estimate was generated of the microseconds required for a single iteration by both SARSA and, where relevant, PASA. The majority of experiments were run on an Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz for both algorithm variants. Timings are principally indicative, given that the implementations (of PASA in particular) haven't been optimised, and the measurements would have been affected by exogenous noise.

[c] Average reward in this environment (given, in particular, that it can assume negative values) doesn't lend itself to a percentage comparison.

List of symbols

$\mathcal{A}$	The action space, or set of all actions. 7
A	The size of the action space. 7
d	Value generated during update to SARSA. 13
$\mathcal{D}$	The set of atomic cells in some discretisation of a continuous state space. 36
D	The number of atomic cells in the set $\mathcal{D}$. 36
η	Learning rate for SARSA. 12
F_R	For a give state-action pair (s_i, a_j), the cumulative distribution function associated with the random variable $R(s_i, a_j)$. 7
$h(\mathcal{I}, \pi)$	The proportion of time that an agent spends in a set of states $\mathcal{I}$ when following the policy π. 28
I	The indicator function. Specifically, $I_A = 1$ if A is true. 9
L_γ	A scoring function. Estimates squared distance from VF using Bellman operator. 8
$\tilde{L}_\gamma$	A scoring function. Estimates squared distance from VF using temporal difference sampling. 8
m	A mapping from the points in a continuous state space to individual atomic cells. 36
Ψ	A probability distribution over a continuous state space. 36
MSE_γ	A scoring function. Measures mean squared difference from the true VF. 8
$\mu_{j,k}$	The sum of the stable state probabilities ψ_i of each state s_i in the cell $\mathcal{X}_{j,k}$ for a given policy π. 22
ν	PASA parameter, which determines the length of the interval between updates to ρ. 16
O	Suppose $f : D \to \mathbb{R}$ and $g : D \to \mathbb{R}$. If $f(x)$ is $O(g(x))$, or equivalently $f(x) = O(g(x))$, or further equivalently $g(x) = \Omega(f(x))$, then there exists $\alpha > 0$ and $\beta > 0$ such that $f(x) \leq \alpha g(x) + \beta$ for all $x \in D$. 3
Ω	Refer to definition of O. 39
P	The transition function, a mapping from each state-action pair to a probability distribution over $\mathcal{S}$. 7
π	The agent's policy. 7
ψ	A vector (of length S) of the stable state probability of visiting each state under a particular policy. 9
Q^π_γ	The value function for a given discount factor and fixed policy π. 7
$\hat{Q}$	An estimate of the VF stored by an RL algorithm. 8
R	The reward function, a mapping from each state-action pair to a random variable. 7

R_{m}	An assumed upper bound on the magnitude of the expected value of the reward function R. 7
ρ	Vector of integers (split vector) used by PASA to define a sequence of cell splits of an initial ordered state partition. 15
$\mathcal{S}$	The state space, or set of all states. 7
S	The size of the state space. 7
$[1{:}k]$	The set of integers j such $1 \leq j \leq k$. 21
Σ	Vector of boolean values stored by PASA to indicate whether a cell is a singleton cell. 16
T_γ^π	The Bellman operator. 8
θ	Matrix of parameters used in parametrised value function approximation. 12
ϑ	PASA parameter, which determines the threshold which must be exceeded before an update is made to ρ. 17
$\bar{u}$	Vector stored by PASA used to estimate the visit probability of each set of states in $\bar{\Xi}$. 16
u	Vector stored by PASA used to estimate the visit probability of each cell in the current partition Ξ. 16
w	Weighting used for each state-action pair when calculating total score. 8
$\tilde{w}$	Weighting used for each state-action pair when calculating total score, with additional constraints. 9
$\bar{\mathcal{X}}_j$	An element of $\bar{\Xi}$ with index j. 15
Ξ	A partition of $\mathcal{S}$, used to help define cells in a state aggregation architecture. 13
$\bar{\Xi}$	A set of subsets of $\mathcal{S}$, used by PASA to generate a partition Ξ. 15
$\mathcal{X}_k$	The cell in a state aggregation architecture with index k. 13
$\mathcal{X}_{j,k}$	In the PASA algorithm, the jth cell in the kth partition generated during the update of Ξ. 17
X	The number of basis functions in a parametrised value function approximation. Also the number of cells in a state aggregation architecture and the size of Ξ. 12
X_0	The number of initial base cells for PASA. 15

Appendix A.

A.1. Assumptions for Theorem 2.2, Chapter 8 of Kushner and Yin (2003)

The theorem relates—adopting the authors' notation, simplified somewhat, in the next equation—to stochastic processes over discrete time steps n of the form:

$$\theta_{n+1} = \Pi_H(\theta_n + \varepsilon Y_n).$$

where $\varepsilon > 0$ is a fixed constant and where Π_H is a truncation operator (note that θ as used in this subsection is distinct from θ as used in the definition of linear approximation architectures). In our case, equating n with t and ε with ς, and referring back to equation (5):

$$\theta_n \equiv \theta^{(t)} = \bar{u}_k^{(t)} \qquad \text{and} \qquad Y_n \equiv Y^{(t)} = \bar{x}_k^{(t)} - \bar{u}_k^{(t)}. \tag{8}$$

We are interested in applying the theorem in the context of Proposition 4. Note in what follows that, for all t, clearly $0 \leq \bar{u}_k^{(t)} \leq 1$ (such that $-1 \leq Y^{(t)} \leq 1$). This implies that we will be able to apply the form of Theorem 2.2 for which the trajectory of the estimate $\bar{u}_k$ is unbounded, such that we can ignore the truncation operator (i.e. treat it as the identity operator).

We address each of the relevant assumptions (using the authors' numbering) required for the result.[48] Assumption (A1.1) requires that $\{Y^{(t)}; t\}$ is uniformly integrable which holds. Assumptions (A1.2) and (A1.3) are not applicable. Assumption (A1.4) we will return to momentarily. Suppose that $\xi^{(t)}$ and $\beta^{(t)}$ are two sequences of random variables. Assumption (A1.5) requires that there exists a function g (measurable on the filtration defined by the sequence of values $\theta^{(u)}$, $Y^{(u-1)}$ and $\xi^{(u)}$ for $1 \leq u \leq t$) such that:

$$\mathrm{E}\left(Y^{(t)} \middle| \theta^{(u)}, Y^{(u-1)}, \xi^{(u)} \text{ for } 1 \leq u \leq t\right) = g\big(\theta^{(t)}, \xi^{(t)}\big) + \beta^{(t)},$$

which holds (we can ignore $\beta^{(t)}$ as in this special case it will always be zero) since the function for $Y^{(t)}$ stated above follows this form if the term $\bar{x}_k^{(t)}$ is interpreted as the random variable $\xi^{(t)}$. Our definition of $\beta^{(t)}$ implies that Assumption (A1.4), which we will not restate, trivially holds. Assumptions (A1.6)-(A1.9) we will not state in full but will be satisfied in our specific case provided g is a continuous function of $\bar{u}_k$, the sequence $\xi^{(t)}$ is bounded within a compact set, the sequence $Y^{(t)}$ is bounded, and provided:

$$\lim_{s\to\infty} \lim_{t\to\infty} \frac{1}{s} \sum_{t'=t}^{s} \mathrm{E}\Big(g\big(\theta, \xi^{(t')}\big)\Big) = \mu$$

for some μ for all $\theta \in [0, 1]$, all of which hold. Finally, assumption (A1.10) requires the sequence $\bar{u}_k$ is bounded with probability one, which is also satisfied given our observation immediately after equation (8) above.

The theorem can also be applied to demonstrate the convergence of SARSA with a fixed linear approximation architecture, fixed policy and a fixed step size parameter η. The special case of state aggregation architectures is of importance for Theorems 12 and 14, as well as Theorem 15.

48. See page 245 of Kushner and Yin (2003).

We will not examine the details, however it is relatively straightforward to examine each of the stated assumptions and demonstrate that each holds in this case. The principle differences are: (a) an additional step is required to solve the system of XA simultaneous equations formed by each of the formulae for $\theta_{kj}^{(t+1)} - \theta_{kj}^{(t)}$ for each pair of indices $1 \leq k \leq X$ and $1 \leq j \leq A$ (so as to find the limit set of the relevant ODE), (b) in the case of general linear approximation architectures, certain conditions must hold in relation to the basis functions—in particular to ensure a suitable limit set of the ODE exists, see page 45 of Kushner and Yin (2003)—which certainly do hold in the special case of state aggregation approximation architectures, and (c) caution needs to be exercised around the function R, since certain functions will violate the required assumptions, hence our assumption in Theorems 12, 14 and Theorem 15 that $|R(\cdot, a_j)|$ is uniformly bounded; weaker conditions exist which would be adequate—conditions which can be inferred from the assumptions stated in Kushner and Yin (2003) which we have referred to above.

Whilst we have not made the details explicit, an example of employing this type of approach (although here in the case where the step size parameter η is a function of t and is slowly decreased in size, and where a general linear approximation architecture is assumed) can be found at page 44 of Kushner and Yin (2003), where the authors describe the convergence of TD(λ) under quite general conditions on the state space (which, for our purposes, extend to both our discrete state space formalism and the continuous state space formalism we adopt in Section 3.4). Their discussion is readily applicable to SARSA with fixed state aggregation and fixed step sizes. See also, for a related discussion, Melo et al. (2008) and Perkins and Precup (2003).

A.2. Proof of Theorem 15

Assume that $\hat{Q}_0 = \hat{Q}_0^{(t)}$ is the VF estimate generated by SARSA with a fixed state aggregation approximation architecture corresponding to the D atomic cells (whilst $\hat{Q} = \hat{Q}^{(t)}$ is the VF estimate generate by SARSA-P).

Note that our discussion regarding the convergence of SARSA with fixed state aggregation and fixed step sizes in Appendix A.1 extends to continuous state spaces such as those described in our continuous state space formalism, provided a stable state distribution $\Psi^{(\infty)}$ exists on $\mathcal{S}$, which we've assumed. We can therefore define $\hat{Q}_{\lim}$ in the same way as in Theorem 12. We can also argue in the same manner as the proof of Theorem 12 to establish that there exists T', η, ϑ and ς such that, for any $\varepsilon_1' > 0$ and $\varepsilon_2 > 0$, with probability at least $1 - \varepsilon_2$ we have $|\hat{Q}(s, a_j) - \hat{Q}_{\lim}(s, a_j)| \leq \varepsilon_1'$ for every $s \in \mathcal{S}$ and $1 \leq j \leq A$ whenever $t > T'$. We can argue equivalently for $\hat{Q}_0$, in relation to a separate limit $\hat{Q}_{\lim,0}$, such that T'' and η'' exist so that $|\hat{Q}_0(s, a_j) - \hat{Q}_{\lim,0}(s, a_j)| \leq \varepsilon_1'$ holds for $t > T''$ with probability at least $1 - \varepsilon_2$. In this way there will exist T such that both inequalities will hold, with probability at least $1 - \varepsilon_2$, for $t > T$.

We now examine $L - L_0$.[49] We define $B = B(s, a_j)$ as follows:

$$B(s, a_j) \coloneqq \mathrm{E}\left(R\big(s^{(t)}, a^{(t)}\big) + \gamma \hat{Q}_{\mathrm{lim}}\big(s^{(t+1)}, a^{(t+1)}\big)\Big|s^{(t)} = s, a^{(t)} = a_j\right) - \hat{Q}_{\mathrm{lim}}(s, a_j).$$

And we define:

$$\begin{aligned} C(s, a_j, d_i) \coloneqq\ & \mathrm{E}\left(R\big(s^{(t)}, a^{(t)}\big) + \gamma \hat{Q}_{\mathrm{lim}}\big(s^{(t+1)}, a^{(t+1)}\big)\Big|s^{(t)} = s, a^{(t)} = a_j\right) \\ & - \mathrm{E}\left(R\big(s^{(t)}, a^{(t)}\big) + \gamma \hat{Q}_{\mathrm{lim}}\big(s^{(t+1)}, a^{(t+1)}\big)\Big|s^{(t)} \in d_i, a^{(t)} = a_j\right), \end{aligned}$$

where the expectation in the second term is over $\Psi^{(\infty)}$ (as well as over the distributions of R and P). We will have, for $t > T$:

$$\begin{aligned} L &\leq \int_{\mathcal{S}} \sum_{j=1}^{A} \tilde{w}(m(s), a_j) B(s, a_j)^2 \, d\Psi^{(\infty)}(s) + \varepsilon_1 \\ &\leq \frac{4(1-h)}{(1-\gamma)^2} R_{\mathrm{m}}^2 + \sum_{i:d_i \in \mathcal{I}} \int_{d_i} \sum_{j=1}^{A} \tilde{w}(d_i, a_j) C(s, a_j, d_i)^2 \, d\Psi^{(\infty)}(s) + \varepsilon_1, \end{aligned}$$

where for any ε_1 we can select ε_1' so that the inequality holds with probability at least $(1-\varepsilon_2)^{1/2}$.

By equivalent reasoning, we will have exactly the same inequality for L_0, with $\hat{Q}_{\mathrm{lim}}$ replaced by $\hat{Q}_{\mathrm{lim},0}$ (due to the fact that the VF architecture for $\hat{Q}$ and $\hat{Q}_0$ are the same over the set $\mathcal{I}$). We will use the following shorthand (as used in the proof of Theorem 12): $\lambda = (1-\delta_P)(1-\delta_\pi)$. We define the event K as being the event that the agent transitions, after iteration t, to the "deterministic" next atomic cell d'' and action a'' (i.e. the agent transitions according to the transition function P_1 and policy π_1, the latter of which exists and is defined according to the definition in Section 3.3, and the former of which can be easily defined by extending the relevant Section 3.3 definition to transitions between atomic cells, consistent with the definition of δ-deterministic in the context of continuous state spaces). The value $\hat{Q}_{\mathrm{lim}}$ associated with the pair (d'', a'') we denote as $\hat{Q}''_{\mathrm{lim}}$. We will have:

$$\begin{aligned} C(s, a_j, d_i) \leq\ & \underbrace{\left|\mathrm{E}\big(R(s^{(t)}, a_j)\big|s^{(t)} = s, a^{(t)} = a_j\big) - \mathrm{E}\big(R(s^{(t)}, a_j)\big|s^{(t)} \in d_i, a^{(t)} = a_j\big)\right|}_{=:F} \\ & + \underbrace{\left|\mathrm{P}\big(K\big|s^{(t)} = s, a^{(t)} = a_j\big) - \lambda\right| \gamma |\hat{Q}''_{\mathrm{lim}}|}_{=:G} \\ & + \underbrace{\mathrm{E}\left(I_{K^c}\gamma\big|\hat{Q}_{\mathrm{lim}}(s^{(t+1)}, a^{(t+1)})\big|\Big|s^{(t)} = s, a^{(t)} = a_j\right) + \gamma(1-\lambda)|M|}_{=:H}, \end{aligned}$$

49. This bound differs the most of the three bounds from the discrete case. The reason for the additional terms is principally technical. Namely, we at no point guarantee that the value of $\hat{Q}_{\mathrm{lim},0}$ is "close to" the value of $\hat{Q}_{\mathrm{lim}}$, which demands that we must constrain the state-action pairs which occur as a result of a certain action in a cell contained in $\mathcal{I}$, in order to ensure that $L - L_0$ is small. The other noticeable difference from the bounds in Theorem 12 is the removal of δ_R from the bound relating to $\tilde{L}$. In the continuous case this term cancels since the bound pertains to $\tilde{L} - \tilde{L}_0$.

where $M = M(a_j, d_i)$ is a residual term (the magnitude of which we will bound in our discussion below). We need to evaluate terms in the right hand side of the following inequality:

$$\int_{d_i} \sum_{j=1}^{A} \tilde{w}(d_i, a_j) C(s, a_j, d_i)^2 \, d\Psi^{(\infty)}(s)$$
$$\leq \int_{d_i} \sum_{j=1}^{A} \tilde{w}(d_i, a_j) \left(F^2 + G^2 + H^2 + 2FG + 2FH + 2GH\right) \, d\Psi^{(\infty)}(s).$$

Our strategy is to argue that all of the terms involving F, G and H, with the exception of F^2, will become small provided λ is close to one. The value of F^2, moreover, will be identical for L_0 for every $d_i \in \mathcal{I}$. This will allow us to bound $L - L_0$. Note that terms involving G will be small if λ is near one because in such a case the VF estimate of $\hat{Q}_{\text{lim}}$ (for the initial state-action pair) closely resembles the temporal difference observed when the event K occurs. Terms involving H will be small if λ is near one because this implies that instances where the VF estimate does *not* closely resemble the temporal difference are rare. We will work through the terms in sequence. First G^2. We denote $J \coloneqq \mathrm{P}(K|s^{(t)} = s, a^{(t)} = a_j)$:

$$\int_{d_i} \sum_{j=1}^{A} \tilde{w}(d_i, a_j) G^2 \, d\Psi^{(\infty)}(s) = \sum_{j=1}^{A} \tilde{w}(d_i, a_j) \int_{d_i} \left(J^2 - 2J\lambda + \lambda^2\right) (\gamma \hat{Q}''_{\text{lim}})^2 \, d\Psi^{(\infty)}(s)$$
$$\leq \sum_{j=1}^{A} \tilde{w}(d_i, a_j) \psi_i \gamma^2 \frac{\lambda - \lambda^2}{(1-\gamma)^2} R_{\text{m}}^2 \leq \psi_i \gamma^2 \frac{\lambda - \lambda^2}{(1-\gamma)^2} R_{\text{m}}^2,$$

where in the first inequality we use the fact that $\psi_i \lambda^2 \leq \int_{d_i} \mathrm{P}(K|s^{(t)} = s)^2 \, d\Psi^{(\infty)}(s) \leq \psi_i \lambda$ for all i. Note the manner in which $\tilde{w}(d_i, a_j)$ was brought outside the integral (the properties of the integrand of course permit this). This can be done for every term, so we omit this step from now on.

For H^2, using $N \coloneqq \hat{Q}_{\text{lim}}(s^{(t+1)}, a^{(t+1)})$, and with the understanding that all expectations are conditioned on $s^{(t)} = s$ and $a^{(t)} = a_j$:

$$\int_{d_i} H^2 \, d\Psi^{(\infty)}(s)$$
$$= \int_{d_i} \left(\mathrm{E}(I_{K^c}\gamma|N|)^2 + 2\mathrm{E}(I_{K^c}\gamma|N|)\gamma(1-\lambda)|M| + \gamma^2(1-\lambda)^2 M^2\right) \, d\Psi^{(\infty)}(s)$$
$$\leq \frac{\gamma^2}{(1-\gamma)^2} R_{\text{m}}^2 \int_{d_i} \left(\mathrm{E}(I_{K^c})^2 + 2(1-\lambda)\mathrm{E}(I_{K^c}) + (1-\lambda)^2\right) \, d\Psi^{(\infty)}(s)$$
$$\leq \frac{\gamma^2}{(1-\gamma)^2} R_{\text{m}}^2 \left((1-\lambda) + 2(1-\lambda)^2 + (1-\lambda)^2\right) \leq \psi_i \gamma^2 \frac{4(1-\lambda)}{(1-\gamma)^2} R_{\text{m}}^2.$$

Similarly (using $|P(K') - \lambda| = P(K')(1-\lambda) + (1 - P(K'))\lambda$ for any event K'):

$$\int_{d_i} FG \, d\Psi^{(\infty)}(s) \leq \gamma \frac{2}{1-\gamma} R_{\text{m}}^2 \int_{d_i} \left| \mathrm{P}\big(K|s^{(t)} = s, a^{(t)} = a_j\big) - \lambda \right| \, d\Psi^{(\infty)}(s)$$
$$\leq \psi_i \gamma \frac{4\lambda(1-\lambda)}{1-\gamma} R_{\text{m}}^2$$

and:

$$\int_{d_i} FH \, d\Psi^{(\infty)}(s) \leq \gamma \frac{2}{1-\gamma} R_{\mathrm{m}}^2 \int_{d_i} \Big(\mathrm{E}\big(I_{K^c} \big| s^{(t)} = s, a^{(t)} = a_j\big) + (1-\lambda) \Big) \, d\Psi^{(\infty)}(s)$$
$$\leq \psi_i \gamma \frac{4(1-\lambda)}{1-\gamma} R_{\mathrm{m}}^2.$$

Finally, using the fact that $|\mathrm{P}(K|s^{(t)} = s, a^{(t)} = a_j) - \lambda| \leq 1$ for all j, and again with the understanding that all expectations are conditioned on $s^{(t)} = s$ and $a^{(t)} = a_j$:

$$\int_{d_i} GH \, d\Psi^{(\infty)}(s) \leq \gamma^2 \frac{2}{(1-\gamma)^2} R_{\mathrm{m}}^2 \int_{d_i} |\mathrm{P}(K) - \lambda| \left(\mathrm{E}(I_{K^c}) + (1-\lambda) \right) \, d\Psi^{(\infty)}(s)$$
$$\leq \gamma^2 \frac{2}{(1-\gamma)^2} R_{\mathrm{m}}^2 \int_{d_i} \left(\mathrm{E}(I_{K^c}) + (1-\lambda) \right) \, d\Psi^{(\infty)}(s) \leq \psi_i \gamma^2 \frac{4(1-\lambda)}{(1-\gamma)^2} R_{\mathrm{m}}^2.$$

To simplify the inequality we multiply some terms by $1/(1-\gamma) \geq 1$, and replace λ and γ by one for others. This gives us, for each $d_i \in \mathcal{I}$:

$$\int_{d_i} \sum_{j=1}^{A} \tilde{w}(d_i, a_j) C(s, a_j, d_i)^2 \, d\Psi^{(\infty)}(s) \leq \psi_i \frac{29(1-\lambda)}{(1-\gamma)^2} R_{\mathrm{m}}^2 + \int_{d_i} \sum_{j=1}^{A} \tilde{w}(d_i, a_j) F^2 \, d\Psi^{(\infty)}(s).$$

We can argue in an identical fashion for each $d_i \in \mathcal{I}$ for L_0, and obtain the same inequality (where F^2 will be an identical function of s). Therefore, for $t > T$, with probability at least $1 - \varepsilon_2$:

$$L - L_0 \leq \frac{4(1-h)}{(1-\gamma)^2} R_{\mathrm{m}}^2 + \varepsilon_1 + \frac{29(1-\lambda)}{(1-\gamma)^2} R_{\mathrm{m}}^2$$
$$\leq \frac{4(1-h) + 29(1 - (1-\delta_P)(1-\delta_\pi))}{(1-\gamma)^2} R_{\mathrm{m}}^2 + \varepsilon_1,$$

where we have used the fact that each term in both L and L_0 which does not cancel must be greater than zero (such that coefficients are not doubled, they are the same as for L).

We now consider $\tilde{L} - \tilde{L}_0$. We define $\tilde{B}(s, a_j)$ as:

$$\tilde{B}(s, a_j) \coloneqq \mathrm{E}\left(R\big(s^{(t)}, a_j\big) + \gamma \hat{Q}_{\mathrm{lim}}\big(s^{(t+1)}, a^{(t+1)}\big) - \hat{Q}_{\mathrm{lim}}(s, a_j) \middle| s^{(t)} = s, a^{(t)} = a_j \right)$$

Using this definition, for any $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ we can select ε_1' (defined above) such that for all $t > T$, with probability at least $(1-\varepsilon_2)^{1/2}$, we have:

$$\tilde{L} \leq \int_{\mathcal{S}} \sum_{j=1}^{A} \tilde{w}(d_i, a_j) \tilde{B}(s, a_j)^2 \, d\Psi^{(\infty)}(s) + \varepsilon_1$$
$$\leq \frac{4(1-h)}{(1-\gamma)^2} R_{\mathrm{m}}^2 + \varepsilon_1 + \sum_{i: d_i \in \mathcal{I}} \sum_{j=1}^{A} \tilde{w}(d_i, a_j) \int_{d_i} \tilde{B}(s, a_j)^2 \, d\Psi^{(\infty)}(s)$$

Suppose we denote $\mathrm{E}_{d_i}(f(s)) \coloneqq \int_{d_i} f(s) \, d\Psi^{(\infty)}(s)$. The next step is much like that in the proof of the bound on $\tilde{L}$ in Theorem 12—see equation (7). Again we use the shorthand

$N = \hat{Q}_{\lim}(s^{(t+1)}, a^{(t+1)})$. For any d_i we can write, cancelling relevant terms:

$$\begin{aligned}
&\mathrm{E}_{d_i}\left(\tilde{B}(s,a_j)^2\right) \\
&= \underbrace{\mathrm{E}_{d_i}\left\{\mathrm{E}\big(R(s^{(t)},a_j)\big|s^{(t)}=s,a^{(t)}=a_j\big)^2\right\} - \left\{\mathrm{E}_{d_i}\mathrm{E}\big(R(s^{(t)},a_j)\big|s^{(t)}=s,a^{(t)}=a_j\big)\right\}^2}_{=:\tilde{A}} \\
&\quad + \underbrace{\gamma^2\mathrm{E}_{d_i}\left\{\mathrm{E}\big(N\big|s^{(t)}=s,a^{(t)}=a_j\big)^2\right\} - \gamma^2\left\{\mathrm{E}_{d_i}\mathrm{E}\big(N\big|s^{(t)}=s,a^{(t)}=a_j\big)\right\}^2}_{=:\tilde{C}}.
\end{aligned}$$

We can define $\tilde{B}_0$ in the same way as $\tilde{B}$, however with $\hat{Q}_{\lim}$ replaced by $\hat{Q}_{\lim,0}$. Proceeding via identical arguments we have $\mathrm{E}_{d_i}(\tilde{B}_0(s,a_j)^2) = \tilde{A} + \tilde{C}_0$, where $\tilde{C}_0$ is the same as $\tilde{C}$ however with $\hat{Q}_{\lim}$ replaced by $\hat{Q}_{\lim,0}$. We can reason in an identical manner to our proof of the bound on $\tilde{L}$ in Theorem 12 to conclude that:

$$|\tilde{C}| \leq \psi_i\gamma^2\frac{1+2(1-\delta_P)(1-\delta_\pi)-3(1-\delta_P)^2(1-\delta_\pi)^2}{(1-\gamma)^2}R_{\mathrm{m}}^2.$$

The same holds for $|\tilde{C}_0|$ of course. Hence we will have, with probability at least $1-\varepsilon_2$:

$$\begin{aligned}
\tilde{L} - \tilde{L}_0 &\leq \frac{4(1-h)}{(1-\gamma)^2}R_{\mathrm{m}}^2 + \varepsilon_1 + \sum_{j=1}^{A}\tilde{w}(d_i,a_j)\int_{\mathcal{S}}\left(\tilde{B}(s,a_j)^2 - \tilde{B}_0(s,a_j)^2\right)\,d\Psi^{(\infty)}(s) \\
&= \frac{4(1-h)}{(1-\gamma)^2}R_{\mathrm{m}}^2 + \varepsilon_1 + \sum_{j=1}^{A}\tilde{w}(d_i,a_j)\sum_{i=1}^{D}(\tilde{C}-\tilde{C}_0) \\
&\leq \frac{4(1-h)}{(1-\gamma)^2}R_{\mathrm{m}}^2 + \varepsilon_1 + \gamma^2\frac{2+4(1-\delta_P)(1-\delta_\pi)-6(1-\delta_P)^2(1-\delta_\pi)^2}{(1-\gamma)^2}R_{\mathrm{m}}^2 \\
&= \big(2(1-h) + \gamma^2\left(1+2(1-\delta_P)(1-\delta_\pi)-3(1-\delta_P)^2(1-\delta_\pi)^2\right)\big)\frac{2R_{\mathrm{m}}^2}{(1-\gamma)^2} + \varepsilon_1.
\end{aligned}$$

Finally we turn to MSE$-$MSE$_0$. As in the discussion for the other two scoring functions, there exists T such that, in this case, for $t > T$:

$$\begin{aligned}
\mathrm{MSE} &= \int_{\mathcal{S}}\sum_{j=1}^{A}\pi(a_j|s)\left(Q^\pi(s,a_j) - \hat{Q}(s,a_j)\right)^2\,d\Psi^{(\infty)}(s) \\
&\leq \int_{\mathcal{S}}\sum_{j=1}^{A}\pi(a_j|s)\left(Q^\pi(s,a_j) - \hat{Q}_{\lim}(s,a_j)\right)^2\,d\Psi^{(\infty)}(s) + \varepsilon_1
\end{aligned}$$

where for any ε_1 we can select ε_1' (defined above) so that the inequality holds with probability at least $(1-\varepsilon_2)^{1/2}$. We can write:

$$\mathrm{MSE} \leq \frac{4(1-h)}{(1-\gamma)^2}R_{\mathrm{m}}^2 + \sum_{i=1}^{D}\int_{d_i}\sum_{j=1}^{A}\pi(a_j|s)\left(Q^\pi(s,a_j) - \hat{Q}_{\lim}(s,a_j)\right)^2\,d\Psi^{(\infty)}(s) + \varepsilon_1 \qquad (9)$$

We now argue in a similar fashion to Theorem 12. Suppose we have some pair (s, a_j) such that $s \in d$ and $d \in \mathcal{I}$. We can separate the value $Q^\pi - \hat{Q}_{\text{lim}}$ into those sequences of states and actions which remain in $\mathcal{I}$ and those which leave $\mathcal{I}$ at some point. We use the values $\xi^{(t')}$ and $\chi^{(t')}$ to represent the expected discounted reward obtained after exactly t' iterations conditioned on $a^{(1)} = a_j$, further conditioned on $s^{(t)} = s$ and $s^{(t)} \in d$ for some $d \in \mathcal{D}$ (with $s^{(t)}$ otherwise selected according to the distribution $\Psi^{(\infty)}$) respectively, and conditioned upon the agent remaining in $\mathcal{I}$ for all $t'' \leq t'$. We will have, for the pair (s, a_j):

$$\begin{aligned} Q^\pi(s, a_j) = &\underbrace{\xi^{(1)} + \sum_{t'=2}^{\infty} \Pr\left(s^{(t'')} \in m^{-1}(\mathcal{I}) \text{ for } t'' \leq t' \middle| s^{(1)} = s, a^{(1)} = a_j\right) \xi^{(t')}}_{=:C_s} \\ &+ \underbrace{\Pr\left(s^{(2)} \notin m^{-1}(\mathcal{I}) \middle| s^{(1)} = s, a^{(1)} = a_j\right) x^{(2)}}_{=:U_s} \\ &+ \underbrace{\sum_{t'=3}^{\infty} \Pr\left(s^{(t'')} \in m^{-1}(\mathcal{I}) \text{ for } t'' < t', s^{(t')} \notin m^{-1}(\mathcal{I}) \middle| s^{(1)} = s, a^{(1)} = a_j\right) x^{(t')}}_{=:V_s} \end{aligned}$$

And similarly, assuming s is distributed according to $\Psi^{(\infty)}$:

$$\begin{aligned} \hat{Q}_{\text{lim}}(s, a_j) = &\underbrace{\chi^{(1)} + \sum_{t'=2}^{\infty} \Pr\left(s^{(t'')} \in m^{-1}(\mathcal{I}) \text{ for } t'' \leq t' \middle| d^{(1)} = m(s), a^{(1)} = a_j\right) \chi^{(t')}}_{=:C_d} \\ &+ \underbrace{\Pr\left(s^{(2)} \notin m^{-1}(\mathcal{I}) \middle| d^{(1)} = m(s), a^{(1)} = a_j\right) x'^{(2)}}_{=:U_d} \\ &+ \underbrace{\sum_{t'=3}^{\infty} \Pr\left(s^{(t'')} \in m^{-1}(\mathcal{I}) \text{ for } t'' < t', s^{(t')} \notin m^{-1}(\mathcal{I}) \middle| d^{(1)} = m(s), a^{(1)} = a_j\right) x'^{(t')}}_{=:V_d} \end{aligned}$$

Note that the values $x^{(t')}$ and $x'^{(t')}$ are, as was the case in Theorem 12, residual terms which reflect the expected total discounted future reward once the agent has left $\mathcal{I}$ for the first time. They are also conditioned on $a^{(1)} = a_j$, and conditioned on $s^{(t)} = s$ and $s^{(t)} \in d$ (with $s^{(t)}$ otherwise selected according to the distribution $\Psi^{(\infty)}$) respectively. Unlike ξ and χ, they reflect the sum of expected discounted reward over an infinite number of iterations. The significance of these values are that we are able to bound them, which we do below. The reason that $x^{(2)}$ and $x'^{(2)}$ are expressed separately as V_s and V_d respectively in the expression above is because (similar to the case in Theorem 12) the first action a_j is not guaranteed to be chosen according to the policy π. We now have:

$$\begin{aligned} \left(Q^\pi(s, a_j) - \hat{Q}_{\text{lim}}(s, a_j)\right)^2 &= (C_s - C_d + U_s - U_d + V_s - V_d)^2 \\ &= C_s^2 + C_d^2 + U_s^2 + U_d^2 + V_s^2 + V_d^2 - 2C_sC_d + \dots \end{aligned} \tag{10}$$

Our strategy, similar to the case for L, will be to show that, in our formula for MSE, any term involving U_s, U_d, V_s or V_d will be small (provided $\delta_{\mathcal{I}}$ is near zero). The remaining terms (which will involve some combination of C_s and C_d only) will cancel when we consider MSE_0. Considering the quantity $|U_s|$, we will have:

$$\begin{aligned}
&\int_{d_i} \sum_{j=1}^{A} \pi(a_j|s)|U_s|\, d\Psi^{(\infty)}(s) \\
&= \int_{d_i} \sum_{j=1}^{A} \pi(a_j|s)\mathrm{Pr}\left(s^{(2)} \notin m^{-1}(\mathcal{I}) \middle| s^{(1)} = s, a^{(1)} = a_j\right) \left|x^{(2)}\right| d\Psi^{(\infty)}(s) \\
&\le \frac{R_{\mathrm{m}}}{1-\gamma} \int_{d_i} \sum_{j=1}^{A} \pi(a_j|s)\mathrm{Pr}\left(s^{(2)} \notin m^{-1}(\mathcal{I}) \middle| s^{(1)} = s, a^{(1)} = a_j\right) d\Psi^{(\infty)}(s) \le \psi_i \frac{\delta_{\mathcal{I}} R_{\mathrm{m}}}{1-\gamma}.
\end{aligned}$$

The argument is substantially the same for $|U_d|$ (the key difference being that the integrand in this latter case is independent of s). Now, focussing on $|V_s|$, we note that:[50]

$$\begin{aligned}
&\int_{d_i} \sum_{j=1}^{A} \pi(a_j|s)|V_s|\, d\Psi^{(\infty)}(s) \\
&\le \int_{d_i} \sum_{j=1}^{A} \pi(a_j|s) \sum_{t'=3}^{\infty} \mathrm{Pr}\left(s^{(t')} \notin m^{-1}(\mathcal{I}) \middle| s^{(t'-1)} \in m^{-1}(\mathcal{I})\right) \underbrace{\left|x^{(t')}\right|}_{\le \gamma^{t'} \frac{R_{\mathrm{m}}}{1-\gamma}} d\Psi^{(\infty)}(s) \\
&\le \frac{R_{\mathrm{m}}}{1-\gamma} \sum_{t'=3}^{\infty} \gamma^{t'} \int_{d_i} \mathrm{Pr}\left(s^{(t')} \notin m^{-1}(\mathcal{I}) \middle| s^{(t'-1)} \in m^{-1}(\mathcal{I})\right) \sum_{j=1}^{A} \pi(a_j|s)\, d\Psi^{(\infty)}(s) \\
&\le \psi_i \frac{R_{\mathrm{m}}}{1-\gamma} \sum_{t'=3}^{\infty} \gamma^{t'} \delta_{\mathcal{I}} \le \psi_i \frac{\delta_{\mathcal{I}} R_{\mathrm{m}}}{(1-\gamma)^2}.
\end{aligned}$$

(Note that, compared to the discrete case arguments in Theorem 12, we have deliberately simplified this inequality by loosening the bound more than is strictly required, to make both the arguments and the equations more succinct.) The same bound is true for $|V_d|$ (again using a substantially equivalent argument). Since each of C_s, C_d, U_s, U_d, V_s and V_d must be bound (at least) by $R_{\mathrm{m}}/(1-\gamma)$, the final contribution to MSE of every term in the last line of (10) which contains U_s, U_d, V_s or V_d is bound by $\delta_{\mathcal{I}} R_{\mathrm{m}}^2/(1-\gamma)^3$. Now we simply observe that, for MSE_0, we will have the same equation as (9), except that U_s, U_d, V_s and V_d are replaced by the distinct values $U_{s,0}$, $U_{d,0}$, $V_{s,0}$ and $V_{d,0}$ respectively (all are subject to the same bounds discussed for MSE). The values C_s and C_d are the same for MSE_0. Not including C_s^2, C_d^2 and $C_s C_d$ there are 33 terms (including duplicates) in the expansion in (10). This gives us, with probability at least $1 - \varepsilon_2$:

$$\mathrm{MSE} - \mathrm{MSE}_0 \le \left(4(1-h) + \frac{33\delta_{\mathcal{I}}}{1-\gamma}\right) \frac{R_{\mathrm{m}}^2}{(1-\gamma)^2} + \varepsilon_1.$$

50. The properties of the integrand allow us to switch the order of the infinite sum and the integral in the third line.

A.3. Proof of Lemma 16

An introductory discussion will help us establish some of the concepts required for the proof. We can make the following observation. If π and P are deterministic, and we pick a starting state s_1, then the agent will create a path through the state space and will eventually revisit a previously visited state, and will then enter a cycle. Call the set of states in this path (including the cycle) $\mathcal{L}_1$ and call the set of states in the cycle $\mathcal{C}_1$. Denote as L_1 and C_1 the number of states in the path (including the cycle) and the cycle respectively. Of course $L_1 \geq C_1 \geq 1$.

If we now place the agent in a state s_2 (arbitrarily chosen) it will either create a new cycle or it will terminate on the path or cycle created from s_1. Call $\mathcal{L}_2$ and $\mathcal{C}_2$ the states in the second path and cycle respectively (and L_2 and C_2 the respective numbers of states, noting that $C_2 = 0$ is possible if the new path terminates on $\mathcal{L}_1$, and in fact that $L_2 = C_2 = 0$ is also possible, if $s_2 \in \mathcal{L}_1$). If we continue in this manner we will have S sets $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_S\}$. Call $\mathcal{C}$ the union of these sets and denote as C the number of states in $\mathcal{C}$. We denote as J_i the event that the ith path created in such a manner terminates on itself, and note that, if this does not occur, then $C_i = 0$. For the next result we continue to assume that π and P are deterministic and also that condition (2) in Example 5 holds.

Lemma 18 *$\mathrm{E}(C_1) = \sqrt{\pi S/8} + O(1)$ and $\mathrm{Var}(C_1) = (32 - 8\pi)S/24 + O(\sqrt{S})$.*

Proof Choose any state s_1. We must have (where probability is conditioned on the prior distribution for P):

$$\Pr(C_1 = i, L_1 = j) = \frac{S-1}{S}\frac{S-2}{S}\cdots\frac{S-j+1}{S}\frac{1}{S} = \frac{(S-1)!}{S^j(S-j)!}.$$

This means that, for large S, the expected value of C_1 can be approximately expressed, making use of Stirling's approximation $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, as:

$$\begin{aligned}
\mathrm{E}(C_1) &= \sum_{j=1}^{S} j \sum_{k=j}^{S} \frac{(S-1)!}{S^k(S-k)!} = \sum_{j=1}^{S} \frac{j(j+1)}{2}\frac{(S-1)!}{S^j(S-j)!} \\
&= \frac{(S-1)!}{2}\sum_{j=1}^{S}\frac{(S-j+1)(S-j+2)}{S^{S-j+1}(j-1)!} \\
&= \frac{S!}{2S^{S+1}}\left(\sum_{j=0}^{S-1}\frac{(S-j)^2S^j}{j!} + \sum_{j=0}^{S-1}\frac{(S-j)S^j}{j!}\right) \\
&= \frac{\sqrt{2\pi S}\left(\frac{S}{e}\right)^S}{2S^{S+1}}\left(\frac{Se^S}{2} + O\left(\sqrt{S}e^S\right)\right) = \sqrt{\frac{\pi}{8}S} + O(1).
\end{aligned}$$

We have used the fact that a Poisson distribution with parameter S will, as S becomes sufficiently large, be well approximated by a normal distribution with mean S and standard deviation $\sqrt{S}$. In this case we are taking the expectation of the negative of the distance from the mean, $S - j$, over the interval from zero to $-S+1$. Hence we can replace the first and second sum in the third equality by the second and first raw moment respectively of a half normal distribution with variance S. The second moment of a half normal is half the variance of the underlying normal distribution, here $S/2$. (The error associated with the Stirling approximation is less than order 1.)

Similarly for the variance, we first calculate the expectation of C_1^2:

$$\begin{aligned}
\mathrm{E}(C_1^2) &= \sum_{j=1}^{S} j^2 \sum_{k=j}^{S} \frac{(S-1)!}{S^k(S-k)!} = \sum_{j=1}^{S} \left(\frac{j^3}{3} + \frac{j^2}{2} + \frac{j}{6}\right) \frac{(S-1)!}{S^j(S-j)!} \\
&= \frac{S!}{S^{S+1}} \sum_{j=0}^{S-1} \left(\frac{(S-j)^3}{3} + \frac{(S-j)^2}{2} + \frac{S-j}{6}\right) \frac{S^j}{j!} \\
&= \frac{\sqrt{2\pi S}\left(\frac{S}{e}\right)^S}{S^{S+1}} \left(\sqrt{\frac{2}{\pi}} \frac{2S^{\frac{3}{2}}}{3} e^S + O\left(Se^S\right)\right) = \frac{4}{3}S + O(\sqrt{S}).
\end{aligned}$$

As a result:

$$\mathrm{Var}(C_1) = \frac{4}{3}S - \frac{\pi S}{8} + O(\sqrt{S}) = \left(\frac{32-8\pi}{24}\right) S + O(\sqrt{S}).$$

■

Note that the expectation can also be derived from the solution to the "birthday problem": the solution to the birthday problem[51] gives the expectation of L_1, and since each cycle length (less than or equal to L_1) has equal probability when conditioned on this total path length, we can divide the average by 2. Maintaining our assumptions from the previous result we have:

Lemma 19 $\mathrm{E}(C) < \mathrm{E}(C_1)(\ln S + 1)$ *and* $\mathrm{Var}(C) = O(S \ln S)$.

Proof We will have:

$$\begin{aligned}
\mathrm{E}(C) &= \sum_{i=1}^{S} \mathrm{E}(C_i) = \sum_{i=1}^{S} \Pr(J_i) \sum_{j=1}^{S} j \Pr(C_i = j | J_i) \\
&\leq \sum_{i=1}^{S} \frac{1}{i} \sum_{j=1}^{S} j \Pr(C_1 = j) < \mathrm{E}(C_1)(\ln S + 1).
\end{aligned}$$

51. For a description of the problem and a formal proof see, for example, page 114 of Flajolet and Sedgewick (2009).

And for the variance:

$$\begin{aligned}
\mathrm{Var}(C) &= \sum_{i=1}^{S}\mathrm{Var}(C_i) + 2\sum_{i=2}^{S}\sum_{j=1}^{i-1}\mathrm{Cov}(C_iC_j) \leq \sum_{i=1}^{S}\mathrm{Var}(C_i)\\
&\leq \sum_{i=1}^{S}\mathrm{E}(C_i^2) = \sum_{i=1}^{S}\Pr(J_i)\sum_{j=1}^{S} j^2\Pr(C_i = j|J_i)\\
&\leq \sum_{i=1}^{S}\frac{1}{i}\sum_{j=1}^{S} j^2\Pr(C_1 = j) < \mathrm{E}(C_1^2)(\ln S + 1)\\
&= \left(\mathrm{Var}(C_1) + \mathrm{E}(C_1)^2\right)(\ln S + 1),
\end{aligned}$$

where we have used the fact that the covariance term must be negative for any pair of lengths C_i and C_j, since if C_i is greater than its mean the expected length of C_j must decrease, and vice versa. We can substitute $\Pr(J_i)$ with $1/i$ in the sum based on the reasoning that, in the absence of any information regarding $\mathcal{L}_j$ for $1 \leq j < i$, we must have $\Pr(L_j \geq k) \geq \Pr(L_i \geq k)$ for all j and $k \geq 0$, which implies that there is at least as much chance of the path $\mathcal{L}_i$ terminating on any other already generated path as of self-terminating for all i, so that $1/i$ is an upper bound on $\Pr(J_i)$. ■

Thus far the definitions of this subsection only apply to deterministic P and π. We can extend all the definitions to arbitrary P and π as follows. Define $P_{\text{det}} = P_{\text{det}}(P)$ as follows:

$$P_{\text{det}}(s_{i'}|s_i, a_j) := \begin{cases} 1 & \text{if } i' = \min\{i'' : P(s_{i''}|s_i, a_j) = \max_{i'''} P(s_{i'''}|s_i, a_j)\} \\ 0 & \text{otherwise} \end{cases}$$

and $\pi_{\text{det}} = \pi_{\text{det}}(\pi)$ as follows:

$$\pi_{\text{det}}(a_j|s_i) := \begin{cases} 1 & \text{if } j = \min\{j' : \pi(a_{j'}|s_i) = \max_{j''} \pi(a_{j''}|s_i)\} \\ 0 & \text{otherwise} \end{cases}$$

Both of these distributions can be most easily interpreted by considering them to be deterministic versions of their arguments, where the most probable transition or action of the argument is taken to be the deterministic transition or action (with, as an arbitrary rule, the lowest index chosen in case of more than one transition or action being equally most-probable). We now extend all relevant definitions introduced in this subsection such that, for example, $\mathcal{C}(P) := \mathcal{C}(P_{\text{det}}(P))$. Our definitions can now be applied to pairs (P, π) as defined in Example 5. We can also now prove Lemma 16:

Proof of Lemma 16 Using Chebyshev's inequality, and Lemmas 18 and 19, for any $K > 1$ and ε_2 satisfying $0 < \varepsilon_2 < K-1$, we can choose S sufficiently high so that $C > K\sqrt{\pi S/8}\ln S$ with probability no greater than $1/(K - \varepsilon_2 - 1)\ln S$. To see this, take $Y := \sqrt{\pi S/8}\ln S$, $\mu := \mathrm{E}(C)$ and $\sigma := \sqrt{\mathrm{Var}(C)}$. Note that for any $\varepsilon' > 0$ we can obtain $\mu \leq (1 + \varepsilon')Y$ for sufficiently large S. Similarly for any $\varepsilon'' > 0$ we can obtain $(1 + \varepsilon'')Y \geq \sqrt{\ln S}\sigma$ for sufficiently large S. We have:

$$\begin{aligned}
\Pr(C > KY) &\leq \Pr\left(|C - \mu| > KY - \mu\right) \leq \Pr\left(|C - \mu| > (K - \varepsilon' - 1)Y\right) \\
&\leq \Pr\left(|C - \mu| > \frac{(K - \varepsilon' - 1)\sqrt{\ln S}\sigma}{1 + \varepsilon''}\right) \\
&\leq \frac{(1 + \varepsilon'')^2}{(K - \varepsilon' - 1)^2 \ln S} \leq \frac{1}{(K - \varepsilon_2 - 1)^2 \ln S},
\end{aligned}$$

where in the first and second inequalities we assume S is sufficiently large so that all but the highest order terms in Lemma 19 can be ignored, and noting that, for any $\varepsilon_2 > 0$ and $K > 1$, we can find $\varepsilon' > 0$ and $\varepsilon'' > 0$ so that the final inequality is satisfied.

We now observe that, if conditions (1) and (3) stated in Example 5 hold, $\psi_i \leq 2\delta$ for all i for which $s_i \notin \mathcal{C}$. This is because ψ_i will be less than or equal to the probability of the agent transitioning from any state $s_{i''}$ to any state $s_{i'}$ where: (a) $s_{i'}$ is not the state which $s_{i''}$ transitions to as dictated by P_{det} and π_{det}, and (b) $s_{i'}$, according to P_{det} and π_{det}, will eventually transition (after any number of iterations) to s_i. This probability is less than or equal to 2δ (the factor of two arises since the agent may transition in a way which is not dictated by P_{det} and π_{det} either due to randomness in P or randomness in π). As a result for any given S we can set δ sufficiently low so that $\sum_{i:s_i\in\mathcal{C}} \psi_i$ is arbitrarily close to one uniformly over the possible values of C. If we define $\mathcal{I}$ as the set of states in $\mathcal{C}$, then this set $\mathcal{I}$ will have the three properties required for the result. ■

A.4. Some Minor Extensions to Corollary 17

We are interested in the possibility of generalising, albeit perhaps only slightly, Lemma 16 and, as a consequence, Corollary 17. We will make reference to notation introduced in Appendix A.3. We continue to assume a deterministic transition function, and a fixed deterministic policy π. In our discussion around uniform priors, we were also able to assume that the sequence of states $s_1, \ldots, s_S$ used to generate the sets $\mathcal{C}_1, \ldots, \mathcal{C}_S$ was selected arbitrarily. In a more general setting we may need to assume that this sequence is generated according to some specific probability distribution. (In contrast, since, in the discussion below, we will generally assume that the prior distribution of the transition probabilities is identical for different actions, we will be able to continue to assume that π is arbitrary, provided it is chosen with no knowledge of P.)

Appealing to techniques which make use of the notion of Schur convexity (Marshall et al., 2011), it's possible to show that, if the random vector $P(\cdot|s_i, a_j)$ is independently distributed for all (i, j), and $\Pr(P(s_{i''}|s_i, a_j) = 1) = \Pr(P(s_{i''}|s_{i'}, a_{j'}) = 1)$ for all (i, j, i', j', i''), then, given an arbitrary policy π, and assuming the sequence of starting states $s_1, \ldots, s_S$ are distributed uniformly at random, $\mathrm{E}(L_1)$ and $\mathrm{Var}(L_1)$ are minimised where the prior for P is uniform. Using this fact, Corollary 17 can be extended to such priors (we omit the details, though the proof uses arguments substantially equivalent to those used in Corollary 17). If we continue to assume an arbitrary policy and that the starting states are selected uniformly at random, we can consider a yet more general class of priors, using a result from Karlin and Rinott (1984). Their result can be used to demonstrate that, of the set of priors which satisfy the following three conditions—(1) that $\Pr(P(s_{i'}|s_i,a_j)=1)=\Pr(P(s_{i'}|s_i,a_{j'})=1)$ for all (i,j,i',j'), (2) that:

$$\Pr(P(s_{i_3}|s_{i_1},a_j)=1)>\Pr(P(s_{i_4}|s_{i_1},a_j)=1)$$
$$\Rightarrow \Pr(P(s_{i_3}|s_{i_2},a_j)=1)>\Pr(P(s_{i_4}|s_{i_2},a_j)=1)$$

for all (i_1,i_2,i_3,i_4,j), and (3) the random vector $P(\cdot|s_i,a_j)$ is independently distributed for all (i,j)—the uniform prior will again maximise the values $\mathrm{E}(L_1)$ and $\mathrm{Var}(L_1)$. Since $C_i \leq L_i$, an equivalent result to Corollary 17 (though not necessarily with the same constant $\sqrt{\pi/8}$) can similarly be obtained for this even larger set of priors (we again omit the details and note that the arguments are substantially equivalent to those in Corollary 17).

Both these results assume a degree of similarity in the transition prior probabilities for each state. A perhaps more interesting potential generalisation is as follows. We can define a *balanced* prior for a deterministic transition function P as any prior such that the random vector $P(\cdot|s_i,a_j)$ is independently distributed for all (i,j), and we have $\Pr(P(s_{i'}|s_i,a_j)=1)=\Pr(P(s_{i'}|s_i,a_{j'})=1)$, $\Pr(P(s_{i'}|s_i,a_j)=1)=\Pr(P(s_i|s_{i'},a_j)=1)$ and $\Pr(P(s_i|s_i,a_j)=1)\geq\frac{1}{S}$ for all (i,j,i',j'). In essence, the prior probability of transitioning from state s_i to $s_{i'}$ is the same as transitioning in the reverse direction from $s_{i'}$ to s_i. This sort of prior would be reflective of many real world problems which incorporate some notion of a geometric space with distances, such as navigating around a grid (examples 1, 2 and 5 all have balanced priors). The difference to the uniform prior is that we now have a notion of the "closeness" between two states (reflected in how probable it is to transition in either direction between them). Similar to the uniform prior there is no inherent "flow" creating cycles which have larger expected value than C in the uniform case.

It is not hard to conceive of examples where $\mathrm{E}(C_1)$ may be significantly reduced for a particular balanced prior compared to the uniform case. It would furthermore appear plausible that, amongst the set of all balanced priors, $\mathrm{E}(L_1)$ would be maximised for the uniform prior. Indeed investigation using numerical optimisation techniques demonstrates this is the case for $S \leq 8$, even when a fixed arbitrary starting state is selected relative to the balanced set of transition probabilities. The techniques used for the generalisations stated above, however, cannot be used to prove a similar result for balanced priors.[52] Notwithstanding this, we conjecture, based on our numerical analysis, that the uniform prior does maximise $\mathrm{E}(L_1)$ for all S, which would carry the implication, since $C_1 \leq L_1$, that Lemma 18 can be used to argue $\mathrm{E}(C_1)=O(\sqrt{S})$ and $\mathrm{Var}(C_1)=O(S)$.

Note that, even if this conjecture holds, we cannot extend Corollary 17 to balanced priors, which we can see with a simple example. Set $\Pr(P(s_i|s_i,a_j)=1)\geq 1-\varepsilon$ for all (i,j) where ε is small. Provided that the transition matrix associated with π and P is irreducible, then $C=S$ for all S.

Notwithstanding that a formal result equivalent to Corollary 17 is unavailable for balanced priors, we should still be able to exploit the apparent tendency of the agent to spend

52. The earlier stated results follow in both cases from the stronger statement that $\Pr(L_1>k)$ is maximised for all k by a uniform prior, from which our conclusions regarding the moments follow. In the case of balanced priors such a strong result does not hold, which can be seen by—for large S, and taking $k=S$—comparing a uniform prior to a prior where all transitions outside of a single fixed cycle covering all S states have probability zero, and where the prior probability of a transition in either direction along this cycle is $(1-1/S)/2$.

a majority of the time in a small subset of the state space. The main difference is that, much like in Example 2, this small subset may change slowly over time.

A.5. Minor Alterations to PASA for Experiments

The changes made were as follows:

1. The underlying SARSA algorithm is adjusted so that $d^{(t)}$ in equation (4) is weighted by the reciprocal of $\pi(s_i, a_j)$, with the effect that rarely taken actions will have a greater impact on changes to θ. This can compensate for a slowing-down of learning which can result from setting ϵ at small values;

2. The role of ϑ in PASA is adjusted slightly. The rule for updating ρ in equation (6) becomes:

$$\rho_k = \begin{cases} j & \text{if } (1-\Sigma_{\rho_k})u_{\rho_k} < \max\{u_i : i \leq X_0 + k - 1, \Sigma_i = 0\} \times \vartheta \\ \rho_k & \text{otherwise} \end{cases}$$

   (the final operator in the first line, originally addition, has been replaced with multiplication).[53] In practice ϑ would be set near to, but less than, one, as opposed to near zero as under the original formulation. This change means that the algorithm can make finer changes when comparing collections of cells with low probabilities, but still remains stable for cells with larger probabilities, which would have estimates more likely to be volatile;

3. A mechanism is introduced whereby, if ρ is changed at the end of an interval of length ν, the values of θ are changed to reflect this. Specifically, if we define, for some j:

$$\Psi = \left\{ k : \mathcal{X}_k^{(n\nu-1)} \cap \mathcal{X}_j^{(n\nu)} \neq \emptyset \right\},$$

   for some $n \in \mathbb{N}$ then we will set, for all $1 \leq l \leq A$:

$$\theta_{jl}^{(n\nu)} = \frac{1}{|\Psi|} \sum_{k \in \Psi} \theta_{kl}^{(n\nu-1)} + \eta I_{\left\{ s^{(n\nu-1)} \in \mathcal{X}_j^{(n\nu-1)} \right\}} I_{\{a^{(n\nu-1)} = a_l\}} d^{(n\nu-1)}. \tag{11}$$

   By definition this only needs to be done at the end intervals of length ν (in practice ν is large). The second term (i.e. outside the summation) in equation (11) comes from equation (3) and just reflects the standard SARSA update. At most, an additional temporary copy of Ξ (to permit a comparison between the set of cells defined at $t = n\nu - 1$ and $t = n\nu$) and a temporary vector of real values of size X (to store copies of single columns from θ) are required to perform the necessary calculations. The nature of each cell $\mathcal{X}_j^{(n\nu)}$ is such that it will either (a) be equal to $\bigcup_{k \in \mathcal{K}} \mathcal{X}_k^{(n\nu-1)}$ for some non-empty subset of indices $\mathcal{K}$ or (b) a strict subset of a single cell $\mathcal{X}_k^{(n\nu-1)}$. Performing this operation tends to speed up learning, as there is less tendency to transfer previously learned weights to unrelated areas of the state space (which might occur as a result of updates to Ξ by PASA as it was originally defined);

53. Note that defining PASA in this new, alternative way poses challenges when seeking to prove convergence (see, for example, Proposition 4).

4. A further mechanism is introduced such that, whenever $t \bmod \nu = 0$ (i.e. for any iteration in which ρ and Ξ are updated), copies of $\bar{\Xi}$ and $\bar{u}$, which we can call $\bar{\Xi}'$ and $\bar{u}'$ respectively, are made immediately before $\bar{\Xi}$ is updated (and are discarded at the end of the update process). The copies are used as part of the process whereby $\bar{\Xi}$ and Ξ are updated.

   Note that there is a natural bijection between the set of all possible elements of $\bar{\Xi}$ (not including the base cells) and the set of all possible pairs of cells generated by a cell split during the splitting process (equate each element of $\bar{\Xi}$ which is not a base cell with the "uppermost" cell created as a consequence of a cell split). At each stage of the splitting process, for $1 \leq k \leq X - X_0$, each element of the set of pairs of cells corresponding to an available split may or may not implicitly correspond to an element of $\bar{\Xi}'$. When making a decision to split a cell, the algorithm, instead of comparing the value of $u_{i_{\max}}$ to the value of u_{ρ_k} (as described in Section 2.4), will compare the value of $u_{i_{\max}}$ to the maximum value of u for all cells in Ξ_{k-1} for which splits implicitly exist already in $\bar{\Xi}'$ (we will denote this maximum as $u_{i_{\text{old}}}$). The threshold ϑ is applied as normal. Specifically, if $u_{i_{\max}}$ exceeds the value of $u_{i_{\text{old}}}$ by at least this threshold then ρ_k is set to equal $i_{\max}$. Otherwise, ρ_k is set to equal i_{old}. (As always we take the lowest index where more than one index equals the maximum.)

   Furthermore, if the "uppermost" of the pair of new cells generated by the chosen cell split is an element of $\bar{\Xi}'$—we may suppose that the new "uppermost" cell corresponds to the element of $\bar{\Xi}'$ with index l—then the value of $\bar{u}_{X_0+k}$ will be made equal to $\bar{u}'_l$, rather than calculated in the manner detailed in Section 2.4. Otherwise, $\bar{u}_{X_0+k}$ will be made equal to $\bar{u}_{\rho_k}/2$. This is to avoid this value being unnecessarily distorted by the standard process PASA uses to estimate the visit frequency of newly split cells.[54]

   The effect of introducing this change is that it can help encourage stability. In particular it can often prevent cell visit frequencies being re-estimated simply due to cells being split in a different order. The change can help to speed up convergence at virtually no computational cost;

5. Finally, the stochastic approximation algorithm in equation (5) is replaced such that $\bar{u}$ is approximated in the following way. We introduce a new variable u_{counter}. Each iteration we update the value as follows: $u_{\text{counter}} = u_{\text{counter}} + I_{\{s^{(t)} \in \bar{\mathcal{X}}_j\}}$. In any iteration such that $t \bmod \nu = 0$ instead of equation (5) we apply the following formula:

$$\bar{u}_j^{(t+1)} = \bar{u}_j^{(t)} + \varsigma \left(u_{\text{counter}}/\nu - \bar{u}_j^{(t)} \right)$$

   and reset u_{counter} to zero. In moving to the approximation, the parameter ς needs to be re-weighted to reflect the change. The reason we replace equation (5) is for practical reasons relating to implementation.[55]

54. Alternative implementations of PASA which use a tree-based approach to govern cell splits—similar to that described in, for example, Nouri and Littman (2009) or Munos and Moore (2002)—can circumvent the need for some of these cumbersome technical details. (Such alternative implementations are beyond our present scope.)
55. On conventional computers, summation can be performed much more quickly than multiplication. By restricting the multiplication step in equation (5) to iterations at the end of each interval of size ν, we

References

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2915–2923, 2016.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

Andrey Bernstein and Nahum Shimkin. Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains. *Machine Learning*, 81(3):359–397, 2010.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996. ISBN 1886529108.

Andrea Bonarini, Alessandro Lazaric, and Marcello Restelli. Learning in complex environments through multiple adaptive partitions. In *Proceedings of the ECAI*, volume 6, pages 7–12, 2006.

Craig Boutilier, Richard Dearden, Moises Goldszmidt, et al. Exploiting structure in policy construction. In *IJCAI*, volume 14, pages 1104–1113, 1995.

Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*, volume 39. CRC press, 2010.

Dotan Di Castro and Shie Mannor. Adaptive bases for reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 312–327. Springer, 2010.

Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University press, 2009.

Geoffrey J Gordon. Chattering in SARSA(λ). Technical report, CMU Learning Lab, 1996.

Geoffrey J Gordon. Reinforcement learning with function approximation converges to a region. In *Advances in Neural Information Processing Systems*, pages 1040–1046, 2001.

Marcus Hutter. Extreme state aggregation beyond Markov decision processes. *Theoretical Computer Science*, 650:73–91, 2016.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4868–4878, 2018.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

Samuel Karlin and Yosef Rinott. Random replacement schemes and multivariate majorization. *Lecture Notes-Monograph Series*, pages 35–40, 1984.

can speed up the algorithm. Since ς is typically very small, the change has negligible effect on the value of $\bar{u}$.

Philipp W Keller, Shie Mannor, and Doina Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 449–456, 2006.

Harold Kushner and G George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*, volume 35. Springer Science & Business Media, 2003.

Sridhar Mahadevan, Stephen Giguere, and Nicholas Jacek. Basis adaptation for sparse nonlinear reinforcement learning. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 654–660, 2013.

Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: Theory of Majorization and its Applications*, volume 143. Springer, second edition, 2011. doi: 10.1007/978-0-387-68276-1.

Francisco S Melo, Sean P Meyn, and M Isabel Ribeiro. An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th International Conference on Machine Learning*, pages 664–671, 2008.

Ishai Menache, Shie Mannor, and Nahum Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238, 2005.

Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability.* Springer Science & Business Media, 2012.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Andrew W Moore and Christopher G Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21(3):199–233, 1995.

Rémi Munos and Andrew Moore. Variable resolution discretization in optimal control. *Machine Learning*, 49(2-3):291–323, 2002.

Ali Nouri and Michael L Littman. Multi-resolution exploration in continuous spaces. In *Advances in Neural Information Processing Systems*, pages 1209–1216, 2009.

Ronald Ortner. Adaptive aggregation for reinforcement learning in average reward Markov decision processes. *Annals of Operations Research*, 208(1):321–336, 2013.

Theodore J Perkins and Doina Precup. A convergent form of approximate policy iteration. In *Advances in Neural Information Processing Systems*, pages 1627–1634, 2003.

Warren B Powell. *Approximate Dynamic Programming: Solving The Curses of Dimensionality.* John Wiley & Sons, second edition, 2011.

Warren B Powell, Hugo P Simao, and Belgacem Bouzaiene-Ayari. Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal on Transportation and Logistics*, 1(3):237–284, 2012.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Stuart I Reynolds. Adaptive resolution model-free reinforcement learning: Decision boundary partitioning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 783–790, 2000.

Gavin A Rummery and Mahesan Niranjan. On-line Q-learning using connectionist systems. Technical report, University of Cambridge, 1994.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Reinforcement learning with soft state aggregation. In *Advances in Neural Information Processing Systems*, pages 361–368, 1995.

Alexander L Strehl, Lihong Li, and Michael L Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10(Nov):2413–2444, 2009.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

Shimon Whiteson, Matthew E Taylor, and Peter Stone. Adaptive tile coding for value function approximation. Technical report, Computer Science Department, University of Texas at Austin, 2007.

Huizhen Yu and Dimitri P Bertsekas. Basis function adaptation methods for cost approximation in MDP. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 74–81. IEEE, 2009.