Reinforcement Learning When All Actions Are Not Always Available

Yash Chandak,[1] **Georgios Theocharous,**[2] **Blossom Metevier,**[1] **Philip S. Thomas**[1]
[1]University of Massachusetts Amherst, [2]Adobe Research
{ychandak, bmetevier, pthomas}@cs.umass.edu, theochar@adobe.com

Abstract

The Markov decision process (MDP) formulation used to model many real-world sequential decision making problems does not efficiently capture the setting where the set of available decisions (actions) at each time step is stochastic. Recently, the stochastic action set Markov decision process (SAS-MDP) formulation has been proposed, which better captures the concept of a stochastic action set. In this paper we argue that existing RL algorithms for SAS-MDPs can suffer from potential divergence issues, and present new policy gradient algorithms for SAS-MDPs that incorporate variance reduction techniques unique to this setting, and provide conditions for their convergence. We conclude with experiments that demonstrate the practicality of our approaches on tasks inspired by real-life use cases wherein the action set is stochastic.

Introduction

In many real-world sequential decision making problems, the set of available decisions, which we call the *action set*, is stochastic. In vehicular routing on a road network (Gendreau, Laporte, and Séguin 1996) or packet routing on the internet (Ribeiro, Sidiropoulos, and Giannakis 2008), the goal is to find the shortest path between a source and destination. However, due to construction, traffic, or other damage to the network, not all pathways are always available. In online advertising (Tan and Srikant 2012; Mahdian, Nazerzadeh, and Saberi 2007), the set of available ads can vary due to fluctuations in advertising budgets and promotions. In robotics (Feng and Yan 2000), actuators can fail. In recommender systems (Harper and Skiba 2007), the set of possible recommendations can vary based on product availability. These examples capture the broad idea and motivate the question we aim to address: *how can we develop efficient learning algorithms for sequential decision making problems wherein the action set can be stochastic?*

Sequential decision making problems *without* stochastic action sets are typically modeled as *Markov decision processes* (MDPs). Although the MDP formulation is remarkably flexible, and can incorporate concepts like stochastic state transitions, partial observability, and even different (deterministic) action availability depending on the state, it cannot efficiently incorporate stochastic action sets. As a result, algorithms designed for MDPs are not well suited to our setting of interest. Recently, Boutilier et al. (2018) laid the foundations for *stochastic action set Markov decision processes* (SAS-MDPs), that extends MDPs to include stochastic action sets. They also showed how the Q-learning and value iteration algorithms, two classic algorithms for approximating optimal solutions to MDPs, can be extended to SAS-MDPs.

In this paper we show that the lack of convergence guarantees of the Q-learning algorithm, when using function approximators in the MDP setting can potentially get exacerbated in the SAS-MDP setting. We therefore derive policy gradient and natural policy gradient algorithms for the SAS-MDP setting and provide conditions for their almost-sure convergence. Critically, since the introduction of stochastic action sets introduces further uncertainty in the decision making process, variance reduction techniques are of increased importance. We therefore derive new approaches to variance reduction for policy gradient algorithms that are unique to the SAS-MDP setting. We validate our new algorithms empirically on tasks inspired by real-world problems with stochastic action sets.

Related Work

While there is extensive literature on solving sequential decision problems modeled as MDPs (Sutton and Barto 2018), there are few methods designed to handle stochastic action sets. Recently, Boutilier et al. (2018) laid the foundation for studying MDPs with stochastic action sets by defining the new SAS-MDP problem formulation, which we review in the background section. After defining SAS-MDPs, Boutilier et al. (2018) presented and analyzed the model-based value iteration and policy iteration algorithms and the model-free Q-learning algorithm for SAS-MDPs.

In the bandit setting, wherein individual decisions are optimized rather than sequences of dependent decisions, *sleeping bandits* extend the standard bandit problem formulation to allow for stochastic action sets (Kanade, McMahan, and Bryan 2009; Kleinberg, Niculescu-Mizil, and Sharma 2010). We focus on the SAS-MDP formulation rather than the sleeping bandit formulation because we are interested in sequential problems. Such sequential problems are more challenging because making optimal decisions requires one to reason about the long-term impact of decisions, which includes reasoning about how a decision will influence the probability that different actions (decisions) will be available in the future.

Although we focus on the *model-free* setting, wherein the dynamics of the environment are not known *a priori* to the agent optimizing its decisions, in the alternative *model-based* setting researchers have considered related problems in the area of *stochastic routing* (Papadimitriou and Yannakakis 1991; Polychronopoulos and Tsitsiklis 1996; Nikolova, Brand, and Karger 2006; Nikolova and Karger 2008). In stochastic routing problems, the goal is to find a shortest path on a graph with stochastic availability of edges. The SAS-MDP framework generalizes stochastic routing problems by allowing for sequential decision making problems that are not limited to shortest path problems.

Background

MDPs and SAS-MDPs (Boutilier et al. 2018) are mathematical formulations of sequential decision problems. Before defining SAS-MDPs, we define MDPs. We refer to the entity interacting with an MDP or SAS-MDP and trying to optimize its decisions as the *agent*.

Formally, an MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{B}, \mathcal{P}, \mathcal{R}, \gamma, d_0)$. $\mathcal{S}$ is the set of all possible states that the agent can be in, called the *state set*. Although our math notation assumes that $\mathcal{S}$ is countable, our primary results extend to MDPs with continuous states. $\mathcal{B}$ is a finite set of all possible actions that the agent can take, called the *base action set*. S_t and A_t are random variables that denote the state of the environment and action chosen by the agent at time $t \in \{0, 1, \dots\}$. $\mathcal{P}$ is called the *transition function* and characterizes how states transition: $\mathcal{P}(s, a, s') \coloneqq \Pr(S_{t+1} = s' | S_t = s, A_t = a)$. $R_t \in [-R_{\max}, R_{\max}]$, a bounded random variable, is the scalar reward received by the agent at time t, where $R_{\max}$ is a finite constant. $\mathcal{R}$ is called the *reward function*, and is defined as $\mathcal{R}(s, a) \coloneqq \mathbb{E}[R_t | S_t = s, A_t = a]$. The reward discount parameter, $\gamma \in [0, 1)$, characterizes how to utility of rewards to the agent decays based on how far in the future they occur. We call d_0 the *start state distribution*, which is defined as $d_0(s) \coloneqq \Pr(S_0 = s)$.

We now turn to defining a SAS-MDP. Let the set of actions available at time t be a random variable, $\mathcal{A}_t \subseteq \mathcal{B}$, which we assume is always not empty, i.e., $\mathcal{A}_t \neq \emptyset$. Let φ characterize the conditional distribution of $\mathcal{A}_t$: $\varphi(s, \alpha) \coloneqq \Pr(\mathcal{A}_t = \alpha | S_t = s)$. We assume that $\mathcal{A}_t$ is Markovian, in that its distribution is conditionally independent of all events prior to the agent entering state S_t given S_t. Formally, a SAS-MDP is $\mathcal{M}' = \{\mathcal{M} \cup \varphi\}$, with the additional requirement that $A_t \in \mathcal{A}_t$.

A policy $\pi : \mathcal{S} \times 2^{\mathcal{B}} \times \mathcal{B} \to [0, 1]$ is a conditional distribution over actions for each state: $\pi(s, \alpha, a) \coloneqq \Pr(A_t = a | S_t = s, \mathcal{A}_t = \alpha)$ for all $s \in \mathcal{S}, a \in \alpha, \alpha \subseteq \mathcal{B}$, and t, where $\alpha \neq \emptyset$. Sometimes a policy is parameterized by a weight vector θ, such that changing θ changes the policy. We write π^θ to denote such a parameterized policy with weight vector θ. For any policy π, we define the corresponding *state-action value function* to be $q^\pi(s, a) \coloneqq \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a, \pi]$, where conditioning on π denotes that $A_{t+k} \sim \pi(S_{t+k}, \mathcal{A}_{t+k}, \cdot)$ for all $\mathcal{A}_{t+k}$ and S_{t+k} for $k \in [t+1, \infty)$. Similarly, the *state-value function* associated with policy π is $v^\pi(s) \coloneqq \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, \pi]$. For a given SAS-MDP $\mathcal{M}'$, the agent's goal is to find an *optimal policy*, π^*, (or equivalently optimal policy parameters θ^*) which is any policy that maximizes the expected sum of discounted future rewards. More formally, an optimal policy is any $\pi^* \in \operatorname{argmax}_{\pi \in \Pi} J(\pi)$, where $J(\pi) \coloneqq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t | \pi]$ and Π denotes the set of all possible policies. For notational convenience, we sometimes use θ in place of π, e.g., to write v^θ, q^θ, or $J(\theta)$, since a weight vector θ induces a specific policy.

As shown by Boutilier et al. (2018), one way to model stochastic action sets using the *MDP formulation (rather than the SAS-MDP formulation)* is to define states such that one can infer $\mathcal{A}_t$ from S_t. Transforming an MDP into a new MDP with $\mathcal{A}_t$ embedded in S_t in this way can result in the size of the state set growing exponentially— by a factor of $2^{|\mathcal{B}|}$. This drastic increase in the size of the state set can make finding or approximating an optimal policy prohibitively difficult. Using the SAS-MDP formulation, the challenges associated with this exponential increase in the size of the state set can be avoided, and one can derive algorithms for finding or approximating optimal policies in terms of the state set of the original underlying MDP. This is accomplished using a variant of the *Bellman operator*, $\mathcal{T}$, which incorporates the concept of stochastic action sets:

$$\mathcal{T}^\pi v(s) = \sum_{\alpha \in 2^{\mathcal{B}}} \varphi(s, \alpha) \sum_{a \in \alpha} \pi(s, \alpha, a) \Big(\sum_{s' \in \mathcal{S}} P(s, a, s') \\ (R(s, a) + \gamma v(s')) \Big) \quad (1)$$

for all $s \in \mathcal{S}$. Similarly, one can extend the *Bellman optimality operator* (Sutton and Barto 2018):

$$\mathcal{T}^* v(s) = \sum_{\alpha \in 2^{\mathcal{B}}} \varphi(s, \alpha) \max_{a \in \alpha} \sum_{s' \in \mathcal{S}} P(s, a, s')(R(s, a) + \gamma v(s')).$$

Boutilier et al. (2018) showed that the stationary optimal policies exists for SAS-MDPs and can be represented using (state-specific) decision lists (or orderings/rankings) over the action set. As a policy takes into account the available set of actions, an optimal policy chooses the highest ranked action from those that are available. Building upon these results, Boutilier et al. (2018) proposed the following update for a tabular estimate, q, of q^{π^*}:

$$q(S_t, A_t) \leftarrow (1 - \eta) q(S_t, A_t) + \eta (R_t + \gamma \max_{a \in \mathcal{A}_{t+1}} q(S_{t+1}, a)). \quad (2)$$

Notice that the maximum is computed only over the available actions, $\mathcal{A}_{t+1}$, in state S_{t+1}. We refer to the algorithm using this update rule as *SAS-Q-learning*.

Potential Limitations of SAS-Q-Learning

Although SAS-Q-learning provides a powerful first model-free algorithm for approximating optimal policies for SAS-MDPs, it inherits several of the drawbacks of the Q-learning

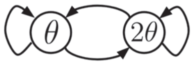

Figure 1: $\theta \to 2\theta$ MDP

algorithm for MDPs. Just like Q-learning, in a state S_t and with available actions $\mathcal{A}_t$, the SAS-Q-learning method chooses actions deterministically when not exploring: $A_t \in \arg\max_{a \in \mathcal{A}_t} q(S_t, a)$. This limits its practicality for problems where optimal policies are stochastic, which is often the case when the environment is partially observable or when the use of function approximation causes state aliasing (Baird 1995). Additionally, if the SAS-Q-learning update converges to an estimate, q, of q^{π^*} such that $\mathcal{T}v(s) = v(s)$ for all $s \in \mathcal{S}$, then the agent will act optimally; however, convergence to a fixed-point of $\mathcal{T}$ is seldom achieved in practice, and reducing the difference between $v(s)$ and $\mathcal{T}v(s)$ (what SAS-Q-learning aims to do) does not ensure improvement of the policy (Sutton and Barto 2018).

SAS-Q-learning does not perform gradient ascent or descent on *any* function, and it can cause divergence of the estimator q when using function approximation, just like Q-learning for MDPs (Baird 1995). In the setting where all actions are always available, SAS-Q-learning reduces to standard Q-learning. Therefore, for all the cases in this setting where Q-learning is unstable, SAS-Q-learning is also unstable. In the setting where all actions are *not* always available, there exist additional cases where Q-learning is stable but SAS-Q-learning is not. However, in such cases where Q-learning is stable, its solution might not be particularly useful as it does not incorporate the notion of stochasticity in the action set (Section 8, Fig.2, Boutilier et al. 2018).

To see this, consider the SAS variant of the classical $\theta \to 2\theta$ MDP (Tsitsiklis and Roy 1983) illustrated in Figure 1. In this example there are two states, s_1 (left in Figure 1) and s_2 (right), and two actions, a_1 = left and a_2 = right. The agent in this example uses function approximation (Sutton and Barto 2018), with weight vector $\theta \in \mathbb{R}^2$, such that $q(s_1, a_1) = \theta_1, q(s_2, a_1) = 2\theta_1$ and $q(s_1, a_2) = \theta_2, q(s_2, a_2) = 2\theta_2$. In either state, if the agent takes the left action, it goes to the left state, and if the agent takes the right action, it goes to the right state. In our SAS-MDP version of this problem, both actions are not always available. Let $R_t = 0$ always, and $\gamma = 1$. Consider the case where the weights of the q-approximation are initialized to $\theta = [-2, -5]$. Now suppose that a transition is observed from the left state to the right state, and after the transition the left action is *not* available to the agent. As per the SAS-Q-learning update rule provided in (2), $\theta_2 \leftarrow \theta_2 + \eta(r + \gamma 2\theta_2 - \theta_2)$. Since $r = 0$ and $\gamma = 1$, this is equivalent to $\theta_2 \leftarrow \theta_2 + \eta\theta_2$. Considering the off-policy setting where this transition is used repeatedly on its own, then irrespective of the learning rate, $\eta > 0$, the weight θ would diverge to $-\infty$. In contrast, had there been no constraint of using max over q given the available actions, the Q-learning update would have been, $\theta_2 \leftarrow \theta_2 + \eta(r + \gamma 2\theta_1 - \theta_2)$ because action a_1 has higher q-value than a_2 due to $\theta_1 > \theta_2$. This would make θ_2 converge to the value -4 (the correct answer is 0).

This provides an example of how the stochastic constraints on the set of available actions can be instrumental in causing the SAS-Q-learning method to diverge, and ignoring the stochastic constraint can prevent Q-learning from converging to the correct solution. We suspect more such cases can be constructed by adapting examples from non-SAS setup (Baird 1995, Gordon 1996, Chpt 11.2 Sutton and Barto 2018).

Policy Gradient Methods for SAS-MDPs

In this section we provide an alternative to the SAS-Q-learning algorithm by deriving policy gradient algorithms (Sutton et al. 2000) for the SAS-MDP setting. While the Q-learning algorithm minimizes the error between $\mathcal{T}v(s)$ and $v(s)$ for all states s (using a procedure that is not a gradient algorithm), policy gradient algorithms perform stochastic gradient ascent on the objective function J. That is, they use the update $\theta \leftarrow \theta + \eta\Delta$, where Δ is an unbiased estimator of $\nabla J(\theta)$.

Unlike the Q-learning algorithm, policy gradient algorithms for MDPs provide convergence guarantees to a critical point (local/global optima) even when using function approximation, and can approximate optimal stochastic policies. However, ignoring the fact that actions are not always available and using off-the-shelf algorithms for MDPs fails to fully capture the problem setting (Boutilier et al. 2018). It is therefore important that we derive policy gradient algorithms that are appropriate for the SAS-MDP setting, as they provide the first convergent model-free algorithms for SAS-MDPs when using function approximation. In the following lemma we extend the expression for the policy gradient for MDPs (Sutton et al. 2000; Thomas 2014) to handle stochastic action sets.

Lemma 1 (SAS Policy Gradient)**.** *For a SAS-MDP, for all $s \in \mathcal{S}$,*

$$\nabla J(\theta) = \sum_{t=0}^{\infty} \sum_{s \in \mathcal{S}} \gamma^t \Pr(S_t = s | \theta) \Big(\sum_{\alpha \in 2^{\mathcal{B}}} \varphi(s, \alpha) \sum_{a \in \alpha} q^\theta(s, a) \frac{\partial \pi^\theta(s, \alpha, a)}{\partial \theta} \Big).$$

Proof. See Appendix A. □

It follows from Lemma 1 that we can create unbiased estimates of $\nabla J(\theta)$, which can be used to update θ using the well-known stochastic gradient ascent algorithm. This algorithm is presented in Algorithm 12. Notably, this process does *not* require the agent to know φ. Also, similar to the SAS-Q-learning method, the policy can be parameterized such that it is not required to embed the available actions as a part of the state. One such parameterization is provided in Appendix F. Notice that in the special case where all actions are always available, the expression in Lemma 1 degenerates to the policy gradient theorem for MDPs (Sutton and Barto 2018). We now establish that SAS policy gradient algorithms are guaranteed to converge to locally optimal policies under the following standard assumptions,

Assumption A1 (Differentiable)**.** *For any state, action-set, and action triplet (s, α, a), policy $\pi^\theta(s, \alpha, a)$ is continuously differentiable in the parameter θ.*

Assumption A2 (Lipschitz smooth gradient)**.** *Let Θ denote the set of all possible parameters for policy π^θ, then for some constant L,*

$$\|\nabla J(\theta) - \nabla J(\bar{\theta})\| \leq L\|\theta - \bar{\theta}\| \forall \theta, \bar{\theta} \in \Theta.$$

Assumption A3 (Learning rate schedule)**.** *Let η_θ^t be the learning rate for updating policy parameters θ, then,*

$$\sum_{t=0}^{\infty} \eta_\theta^t = \infty, \quad \sum_{t=0}^{\infty} (\eta_\theta^t)^2 < \infty.$$

All the assumptions (A1-A3) are satisfied under standard policy parameterization techniques (linear-function/neural-networks with softmax) and appropriately set learning rates.

Lemma 2. *Under Assumptions* (A1)-(A3)*, the SAS policy gradient algorithm causes $\nabla J(\theta_t) \to 0$ as $t \to \infty$, with probability one.*

Proof. See Appendix B. □

Natural policy gradient algorithms (Kakade 2002) extend policy gradient algorithms to follow the *natural gradient* of J (Amari 1998). In essence, whereas policy gradient methods perform gradient ascent in the space of policy parameters by computing the gradient of J as a function of the parameters θ, natural policy gradient methods perform gradient ascent in the space of policies (which are probability distributions) by computing the gradient of J as a function of the policy, π. Thus, whereas policy gradient implicitly measures distances between policies by the Euclidean distance between their policy parameters, natural policy gradient methods measure distances between policies using notions of distance between probability distributions. In the most common form of natural policy gradients, the distances between policies are measured using a Taylor approximation of *Kullback-Leibler divergence* (KLD). By performing gradient ascent in the space of policies rather than the space of policy parameters, the natural policy gradient becomes invariant to how the policy is parameterized (Thomas, Dann, and Brunskill 2018), which can help to mitigate the vanishing gradient problem in neural networks and improve learning speed (Amari 1998).

The natural policy gradient (using a Taylor approximation of KLD to measure distances) is $\widetilde{\nabla} J(\theta) := F_\theta^{-1} \nabla J(\theta)$ where F_θ is the *Fisher information matrix* (FIM) associated with the policy π_θ. Although the FIM is a well-known quantity, it is typically associated with a parameterized probability distribution. Here, π_θ is a *collection* of probability distributions—one per state. This raises the question of what F_θ should be when computing the natural policy gradient. Following the work of Bagnell and Schneider (2003) for MDPs, we show that the FIM, F_θ, for computing the natural policy gradient for a SAS-MDP can also be derived by viewing π_θ as a distribution over possible *trajectories* (sequences of states, available action sets and executed actions).

Property 1 (Fisher Information Matrix)**.** *For a policy, parameterized using weights θ, let $\psi^\theta(s, \alpha, a) := \partial \log \pi^\theta(s, \alpha, a)/\partial\theta$, then the Fisher information matrix is,*

$$F_\theta = \sum_{t=0}^{\infty} \sum_{s \in \mathcal{S}} \gamma^t \Pr(S_t = s|\theta) \sum_{\alpha \in 2^{\mathcal{B}}} \Big(\varphi(s, \alpha) \sum_{a \in \alpha} \pi^\theta(s, \alpha, a) \psi^\theta(s, \alpha, a) \psi^\theta(s, \alpha, a)^\top \Big).$$

Proof. See Appendix C. □

Furthermore, Kakade (2002) showed that many terms in the definition of the natural policy gradient cancel, providing a simple expression for the natural gradient which can be estimated with time linear in the number of policy parameters per time step. We extend the result of Kakade (2002) to the SAS-MDP formulation in the following lemma:

Lemma 3 (SAS Natural Policy Gradient)**.** *Let w be a parameter such that,*

$$\frac{\partial}{\partial w} \mathbb{E}\left[\frac{1}{2} \sum_{t}^{\infty} \gamma^t \left(\psi^\theta(S_t, \mathcal{A}_t, A_t)^\top w - q^\theta(S_t, A_t) \right)^2 \right] = 0,$$

then for all $s \in \mathcal{S}$ in $\mathcal{M}'$, $\widetilde{\nabla} J(\theta) = w$.

Proof. See Appendix C. □

From Lemma 3, we can derive a computationally efficient natural policy gradient algorithm by using the well-known *temporal difference* algorithm (Sutton and Barto 2018), modified to work with SAS-MDPs, to estimate q^θ with the approximator $\psi^\theta(S_t, \mathcal{A}_t, A_t)^\top w$, and then using the update $\theta \leftarrow \theta + \eta w$. This algorithm, which is the SAS-MDP equivalent of NAC-TD (Bhatnagar et al. 2008; Degris, Pilarski, and Sutton 2012; Morimura, Uchibe, and Doya 2005; Thomas and Barto 2012), is provided in Algorithm 2 in Appendix E.

Adaptive Variance Mitigation

In the previous section, we derived (natural) policy gradient algorithms for SAS-MDPs. While these algorithms avoid the divergence of SAS-Q-learning, they suffer from the high variance of policy gradient estimates (Kakade and others 2003). As a consequence of the additional stochasticity that results from stochastic action sets, this problem can be even more severe in the SAS-MDP setting. In this section, we leverage insights from the Bellman equation for SAS-MDPs, provided in (1), to reduce the variance of policy gradient estimates.

One of the most popular methods to reduce variance is the use of a state-dependent baseline $b(s)$. Sutton et al. (2000) showed that, for any state-dependent baseline $b(s)$:

$$\nabla J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \psi^\theta(s, \alpha, a) \left(q^\theta(s, a) - b(s) \right) \right]. \quad (3)$$

For any random variables X and Y, we know that the variance of $X - Y$ is given by $\text{var}(X - Y) = \text{var}(X) +$

$\text{var}(Y) - 2\text{cov}(X, Y)$, where cov stands for covariance. Therefore, the variance of $X - Y$ is lesser than variance of X if $2\text{cov}(X, Y) > \text{var}(Y)$. As a result, any state dependent baseline $b(s)$ whose value is sufficiently correlated to the expected return, $q^\theta(s, a)$, can be used to reduce the variance of the sample estimator of (3). A baseline dependent on both the state and action can have higher correlation with $q^\theta(s, a)$, and could therefore reduce variance further. However, such action dependent baselines cannot be used directly, as they can result in biased gradient estimates. Developing such baselines remains an active area of research for MDPs (Thomas and Brunskill 2017; Grathwohl et al. 2017; Liu et al. 2017; Wu et al. 2018; Tucker et al. 2018) and is largely complementary to our purpose. Further, even the optimal state-dependent baseline (Greensmith, Bartlett, and Baxter 2004), which leads to the minimum variance gradient estimator, is not feasible to compute and only under certain restrictive assumptions reduces to the common choice of state-value function estimator, $\hat{v}(s)$. Therefore, in the following, we propose multiple baselines that are easy to compute, and then combine them optimally.

We now introduce a baseline for SAS-MDPs that lies between state-dependent and state-action-dependent baselines. Like state-dependent baselines, these new baselines do not introduce bias into gradient estimates. However, like action-dependent baselines these new baselines include *some* information about the chosen actions. Specifically, we propose baselines that depend on the state, S_t, and available action set $\mathcal{A}_t$, but not the precise action, A_t.

Recall from the SAS Bellman equation (1) that the state-value function for SAS-MDPs can be written as, $v^\theta(s) = \sum_{\alpha \in 2^\mathcal{B}} \varphi(s, \alpha) \sum_{a \in \alpha} \pi^\theta(s, \alpha, a) q^\theta(s, a)$. While we cannot directly use a baseline dependent on the action sampled from π^θ, we *can* use baseline dependent on the sampled action *set*. We consider a new baseline which leverages this information about the sampled action set α. This baseline is $\bar{q}(s, \alpha) := \sum_{a \in \alpha} \pi^\theta(s, \alpha, a) \hat{q}(s, a)$, where $\hat{q}$ is a learned estimator of the state-action value function, and $\bar{q}$ represents its expected value under the current policy, π^θ, conditioned on the sampled action set α.

In principle, we expect $\bar{q}(S_t, \mathcal{A}_t)$ to be more correlated with $q^\theta(S_t, A_t)$ as it explicitly conditions on the action set and does not compute an average over all action sets possible, like $\hat{v}$. Practically, however, estimating q values can be harder than estimating v. This can be attributed to the fact that with the same number of training samples, the number of parameters to learn in $\hat{q}$ is more than those in an estimate of v^θ. This poses a new dilemma of deciding when to use which baseline. To get the best of both, we consider using a weighted combination of $\hat{v}(S_t)$ and $\bar{q}(S_t, \mathcal{A}_t)$. In the following property we establish that using any weighted combination of these two baselines results in an unbiased estimate of the SAS policy gradient.

Property 2 (Unbiased estimator)**.** *Let* $\hat{J}(s, \alpha, a, \theta) := \psi^\theta(s, \alpha, a)\big(q^\theta(s, a) + \lambda_1 \hat{v}(s) + \lambda_2 \bar{q}(s, \alpha)\big)$ *and* $d^\pi(s) := (1 - \gamma) \sum_t^\infty \gamma^t \Pr(S_t = s)$*, then for any values of* $\lambda_1 \in \mathbb{R}$ *and* $\lambda_2 \in \mathbb{R}$*,*

$$\nabla J(\theta) = \mathbb{E}\left[\hat{J}(s, \alpha, a, \theta) \middle| d^\pi, \varphi, \pi\right].$$

Proof. See Appendix D. □

Algorithm 1: Stochastic Action Set Policy Gradient (SAS-PG)

```
1  $\mathbf{A} = [\lambda_1, \lambda_2]^\top = [-0.5, -0.5]^\top$        ▷ Initialize $\lambda$'s
2  for $episode = 0, 1, 2...$ do
       # Collect transition batch using $\pi^\theta$
3      $\mathbb{B} = \{(s_0, \alpha_0, a_0, r_0), ..., (s_T, \alpha_T, a_T, r_T)\}$
4      $\hat{G}(s_t) = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$
       # Perform update on parameters using batch $\mathbb{B}$
5      $\psi^\theta(s, \alpha, a) = \frac{\partial \log \pi^\theta(s, \alpha, a)}{\partial \theta}$
6      $\varpi \leftarrow \varpi + \eta_\varpi(\hat{G}(s) - \hat{v}^\varpi(s)) \frac{\partial \hat{v}^\varpi(s)}{\partial \varpi}$
7      $\omega \leftarrow \omega + \eta_\omega(\hat{G}(s) - \bar{q}^\omega(s, \alpha)) \frac{\partial \bar{q}^\omega(s, \alpha)}{\partial \omega}$
8      $\theta \leftarrow \theta + \eta_\theta(\hat{G}(s) + \lambda_1 \hat{v}^\varpi(s) +$
         $\lambda_2 \bar{q}^\omega(s, \alpha))\psi^\theta(s, \alpha, a)$        ▷ Update $\pi^\theta$
       # Automatically tune hyper-parameters for
         variance reduction using $\mathbb{B}$
9      $\mathbf{B} = [\psi^\theta(s, \alpha, a)\hat{v}^\varpi(s), \psi^\theta(s, \alpha, a)\bar{q}^\omega(s, \alpha)]^\top$
10     $\mathbf{C} = [\psi^\theta(s, \alpha, a)\hat{G}(s)]^\top$
11     $\hat{\mathbf{A}} \leftarrow -(\mathbb{E}[\mathbf{B}^\top \mathbf{B}])^{-1} \mathbb{E}[\mathbf{B}^\top \mathbf{C}]$
12     $\mathbf{A} \leftarrow \eta_\lambda \mathbf{A} + (1 - \eta_\lambda)\hat{\mathbf{A}}$        ▷ Update $\lambda$'s
```

The question remains: what values should be used for λ_1 and λ_2 for combining $\hat{v}$ and $\bar{q}$? Similar problems of combining different estimators has been studied in statistics literature (Graybill and Deal 1959; Meir and others 1994) and more recently for combining control variates (Wang et al. 2013; Geffner and Domke 2018). Building upon their ideas, rather than leaving λ_1 and λ_2 as open hyperparameters, we propose a method for automatically adapting $\mathbf{A} = [\lambda_1, \lambda_2]$ for the specific SAS-MDP and current policy parameters, θ. The following lemma presents an analytic expression for the value of $\mathbf{A}$ that minimizes a sample-based estimate of the variance of $\hat{J}$.

Lemma 4 (Adaptive variance mitigation)**.** *If* $\mathbf{A} = [\lambda_1, \lambda_2]^\top$, $\mathbf{B} = [\psi^\theta(s, \alpha, a)\hat{v}(s), \psi^\theta(s, \alpha, a)\bar{q}(s, \alpha)]^\top$, *and* $\mathbf{C} = [\psi^\theta(s, \alpha, a)q^\theta(s, a)]^\top$, *where* $\mathbf{A} \in \mathbb{R}^{2\times 1}, \mathbf{B} \in \mathbb{R}^{d\times 2}$, *and* $\mathbf{C} \in \mathbb{R}^{d\times 1}$*, then the* $\mathbf{A}$ *that minimizes the variance of* $\hat{J}$ *is given by*

$$\mathbf{A} = -\left(\mathbb{E}\left[\mathbf{B}^\top \mathbf{B}\right]\right)^{-1} \mathbb{E}\left[\mathbf{B}^\top \mathbf{C}\right]. \tag{4}$$

Proof. See Appendix D. □

Lemma 4 provides the values for λ_1 and λ_2 that result in the minimal variance of $\hat{J}$. Note that the computational cost associated with evaluating the inverse of $\mathbb{E}\left[\mathbf{B}^\top \mathbf{B}\right]$ is negligible because its dimension is always $\mathbb{R}^{2\times 2}$, independent of the number of policy parameters. Also, Lemma 4 provides the optimal values of λ_1 and λ_2, which still must be approximated using sample-based estimates of $\mathbf{B}$ and $\mathbf{C}$.

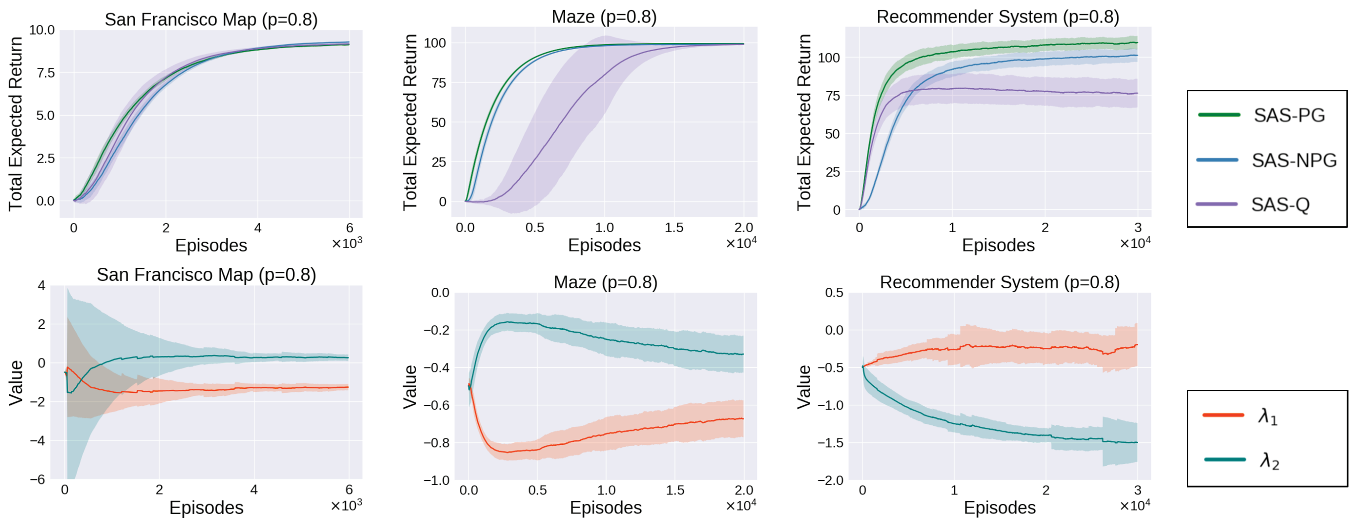

Figure 2: (Top) Best performing learning curves on the domains considered. The probability of any action being available in the action set is 0.8. (Bottom) Autonomously adapted values of λ_1 and λ_2 associated with $\hat{v}$ and $\bar{q}$, respectively, for the SAS-PG results. Shaded regions correspond to one standard deviation obtained using 30 trials.

Furthermore, one might use double sampling for $\mathbf{B}$ to get unbiased estimates of the variance minimizing value of $\mathbf{A}$ (Baird 1995). However, as Property 2 ensures that estimates of $\hat{J}$ for any value of λ_1 and λ_2 are always unbiased, we opt to use all the available samples for estimating $\mathbb{E}[\mathbf{B}^\top\mathbf{B}]$ and $\mathbb{E}[\mathbf{B}^\top\mathbf{C}]$.

Algorithm

Pseudo-code for the SAS policy gradient algorithm is provided in Algorithm 12. Let the estimators of v^θ and q^θ be $\hat{v}^\varpi$ and $\hat{q}^\omega$, which are parameterized using ϖ and ω, respectively. Let π^θ corresponds to the policy parameterized using θ. Let $\eta_\varpi, \eta_\omega, \eta_\theta$ and η_λ be the learning-rate hyper-parameters. We begin by initializing the λ values to -0.5 each, such that it takes an average of both the baselines and subtracts it off from the sampled return. In Lines 3 and 4, we execute π^θ to observe the trajectory and compute the return. Lines 6 and 7 correspond to the updates for parameters associated with $\hat{v}^\varpi$ and $\hat{q}^\omega$, using their corresponding TD errors (Sutton and Barto 2018). The policy parameters are then updated using a combination of both the baselines. We drop the γ^t dependency for data efficiency (Thomas 2014). As per Lemma 4, for automatically tuning the values of λ_1 and λ_2, we create the sample estimates of the matrices $\mathbf{B}$ and $\mathbf{C}$ using the transitions from batch $\mathbb{B}$, in Lines 9 and 10. To update the values of λ's, we compute $\hat{\mathbf{A}}$ using the sample estimates of $\mathbb{E}[\mathbf{B}^\top\mathbf{B}]$ and $\mathbb{E}[\mathbf{B}^\top\mathbf{C}]$. While computing the inverse, a small diagonal noise is added to ensure that inverse exists. As everything is parameterized using smooth function, we know that the subsequent estimates of $\mathbf{A}$ should not vary a lot. Since we only have access to the sample estimate of $\mathbf{A}$, we leverage the Polyak-Rupert averaging in Line 12 for stability. Due to space constraints, the algorithm for SAS natural policy gradient is deferred to Appendix E.

Empirical Analysis

In this section we use empirical studies to answer the following three questions: (**a**) How do our proposed algorithms, SAS policy gradient (SAS-PG) and SAS natural policy gradient (SAS-NPG), compare to the prior method SAS-Q-learning? (**b**) How does our adaptive variance reduction technique weight the two baselines over the training duration? (**c**) What impact does the probability of action availability have on the performances of SAS-PG, SAS-NPG, and SAS-Q-learning? To evaluate these aspects, we first briefly introduce three domains inspired by real-world problems.

Routing in San Francisco. This task models the problem of finding shortest paths in San Francisco, and was first presented with stochastic actions by Boutilier et al. (2018). Stochastic actions model the concept that certain paths in the road network may not be available at certain times. A positive reward is provided to the agent when it reaches the destination, while a small penalty is applied at every time step. We modify the domain presented by Boutilier et al. (2018) so that the starting state of the agent is not one particular node, but rather is uniformly randomly chosen among all possible locations. This makes the problem more challenging, since it requires the agent to learn the shortest path from every node. All the states (nodes) are discrete, and edges correspond to the action choices. Each edge is made available with some fixed probability. The overall map is shown in Appendix.

Robot locomotion task in a maze. In this domain, the agent has to navigate a maze using unreliable actuators. The agent starts at the bottom left corner and a goal reward is given when it reaches the goal position, marked by a star (see Appendix for the figure). The agent is penalized at each time step to encourage it to reach the goal as quickly as possible. The state space is continuous, and corresponds to real-valued

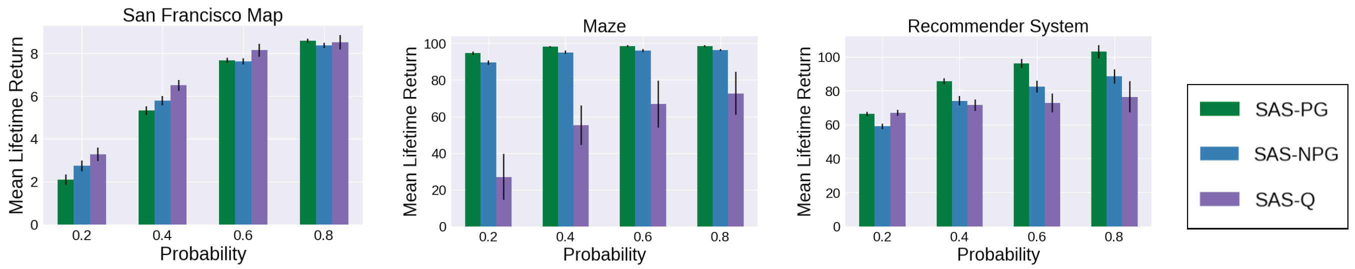

Figure 3: Best performances of different algorithms across different values of probabilities for action availability. The error bars correspond to one standard deviation obtained using 30 trials.

Cartesian coordinates of the agent's position. The agent has 16 actuators pointing in different directions. Turning each actuator on moves the agent in the direction of the actuator. However, each actuator is unreliable, and is therefore only available with some fixed probability.

Product recommender system. In online marketing and sales, product recommendation is a popular problem. Due to various factors such as stock outage, promotions, delivery issues etc., not all products can be recommended always. To model this, we consider a synthetic setup of providing recommendation to a user from a batch of 100 products, each available with some fixed probability and associated with a stochastic reward corresponding to profit. Each user has a real-valued context, which forms the state space, and the recommender system interacts with a randomly chosen user for 5 steps. The goal for the recommender system is to suggest products that maximize total profit. Often the problem of recommendation is formulated as a contextual bandit or collaborative filtering problem, but as shown by Theocharous, Thomas, and Ghavamzadeh (2015) these approaches fail to capture the long term value of the prediction. Hence we resort to the full RL setup.

Results

Here we only discuss the representative results for the three major questions of interest. Plots for detailed evaluations are available in Appendix F.

(a) For the routing problem in San Francisco, as both the states and actions are discrete, the q-function for each state-action pair has a unique parameter. When no parameters are shared, SAS-Q-learning will not diverge. Therefore, in this domain, we notice that SAS-Q-learning performs similarly to the proposed algorithms. However, in many large-scale problems, the use of function approximators is crucial for estimating the optimal policy. For the robot locomotion task in the maze domain and the recommender system, the state space is not discrete and hence function approximators are required to obtain the state features. As we saw in Section , the sharing of state features can create problems for SAS-Q-learning. The increased variance in the performance of SAS-Q-learning is visible in both the Maze and the Recommender system domains in Figure 2. While the SAS-Q eventually performs the same on the Maze domain, its performance improvement saturates quickly in the recommender system domain thus resulting in a sub-optimal policy.

(b) To provide visual intuition for the behavior of adaptive variance mitigation, we report the values of λ_1 and λ_2 over the training duration in Figure 2. As several factors are combined through (4) to influence the λ values, it is hard to pinpoint any individual factor that is responsible for the observed trend. However, note that for both the routing problem in San Francisco and the robot navigation in maze, the goal reward is obtained on reaching the destination and intermediate actions do not impact the total return significantly. Intuitively, this makes the action set conditioned baseline $\bar{q}$ similarly correlated to the observed return as the state only conditioned baseline, $\hat{v}$, but at the expense of estimating significantly more number of parameters. Thus the importance for $\bar{q}$ is automatically adapted to be closer to zero. On the other hand, in recommender system, each product has a significant amount of associated reward. Therefore, the total return possible during each episode has a strong dependency on the available action set and thus the magnitude of weight for $\bar{q}$ is much larger than that for v.

(c) To understand the impact of the probability of an action being available, we report the best performances for all the algorithms for different probability values in Figure 3. We notice that in the San Francisco routing domain, SAS-Q-learning has a slight edge over the proposed methods. This can be attributed to the fact that off-policy samples can be re-used without causing any divergence problems as state features are not shared. For the maze and the recommender system tasks, where function approximators are necessary, the proposed methods significantly out-perform SAS-Q.

Conclusion

Building upon the SAS-MDP framework of Boutilier et al. (2018), we studied an under-addressed problem of dealing with MDPs with stochastic action sets. We highlighted some of the limitations of the existing method and addressed them by generalizing policy gradient methods for SAS-MDPs. Additionally, we introduced a novel baseline and an adaptive variance reduction technique unique to this setting. Our approach has several benefits. Not only does it generalize the theoretical properties of standard policy gradient methods, but it is also practically efficient and simple to implement.

Acknowledgement

The research was supported by and partially conducted at Adobe Research. We are also immensely grateful to the three anonymous reviewers who shared their insights and feedback, specially to the second reviewer who helped improve the counter example.

References

Amari, S.-I. 1998. Natural gradient works efficiently in learning. *Neural computation* 10(2):251–276.

Bagnell, J. A., and Schneider, J. G. 2003. Covariant policy search. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence.*

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

Bhatnagar, S.; Ghavamzadeh, M.; Lee, M.; and Sutton, R. S. 2008. Incremental natural actor-critic algorithms. In *Advances in neural information processing systems*, 105–112.

Boutilier, C.; Cohen, A.; Daniely, A.; Hassidim, A.; Mansour, Y.; Meshi, O.; Mladenov, M.; and Schuurmans, D. 2018. Planning and learning with stochastic action sets. In *IJCAI*.

Degris, T.; Pilarski, P. M.; and Sutton, R. S. 2012. Model-free reinforcement learning with continuous action in practice. In *Proceedings of the 2012 American Control Conference*.

Feng, Y., and Yan, H. 2000. Optimal production control in a discrete manufacturing system with unreliable machines and random demands. *IEEE Transactions on Automatic Control*.

Geffner, T., and Domke, J. 2018. Using large ensembles of control variates for variational inference. In *Advances in Neural Information Processing Systems*.

Gendreau, M.; Laporte, G.; and Séguin, R. 1996. Stochastic vehicle routing. *European Journal of Operational Research*.

Gordon, G. J. 1996. Chattering in sarsa(lambda). *A CMU Learning Lab Internal Report*.

Grathwohl, W.; Choi, D.; Wu, Y.; Roeder, G.; and Duvenaud, D. 2017. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*.

Graybill, F. A., and Deal, R. 1959. Combining unbiased estimators. *Biometrics* 15(4):543–550.

Greensmith, E.; Bartlett, P. L.; and Baxter, J. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research* 5(Nov):1471–1530.

Harper, G. W., and Skiba, S. 2007. User-personalized media sampling, recommendation and purchasing system using real-time inventory database. US Patent 7,174,312.

Kakade, S. M., et al. 2003. *On the sample complexity of reinforcement learning*. Ph.D. Dissertation, University of London London, England.

Kakade, S. M. 2002. A natural policy gradient. In *Advances in neural information processing systems*, 1531–1538.

Kanade, V.; McMahan, H. B.; and Bryan, B. 2009. Sleeping experts and bandits with stochastic action availability and adversarial rewards. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS*.

Kleinberg, R.; Niculescu-Mizil, A.; and Sharma, Y. 2010. Regret bounds for sleeping experts and bandits. *Machine learning*.

Liu, H.; Feng, Y.; Mao, Y.; Zhou, D.; Peng, J.; and Liu, Q. 2017. Action-depedent control variates for policy optimization via stein's identity. *arXiv preprint arXiv:1710.11198*.

Mahdian, M.; Nazerzadeh, H.; and Saberi, A. 2007. Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM conference on Electronic commerce*. ACM.

Meir, R., et al. 1994. *Bias, variance and the combination of estimators: The case of linear least squares*. Citeseer.

Morimura, T.; Uchibe, E.; and Doya, K. 2005. Utilizing the natural gradient in temporal difference reinforcement learning with eligibility traces. In *International Symposium on Information Geometry and its Application*, 256–263.

Nikolova, E., and Karger, D. R. 2008. Route planning under uncertainty: The canadian traveller problem. In *AAAI*.

Nikolova, E.; Brand, M.; and Karger, D. R. 2006. Optimal route planning under uncertainty. In *ICAPS*, volume 6, 131–141.

Papadimitriou, C. H., and Yannakakis, M. 1991. Shortest paths without a map. *Theoretical Computer Science* 84(1):127–150.

Polychronopoulos, G. H., and Tsitsiklis, J. N. 1996. Stochastic shortest path problems with recourse. *Networks: An International Journal* 27(2):133–143.

Ribeiro, A.; Sidiropoulos, N. D.; and Giannakis, G. B. 2008. Optimal distributed stochastic routing algorithms for wireless multihop networks. *IEEE Transactions on Wireless Communications*.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Tan, B., and Srikant, R. 2012. Online advertisement, optimization and stochastic networks. *IEEE Transactions on Automatic Control*.

Theocharous, G.; Thomas, P. S.; and Ghavamzadeh, M. 2015. Ad recommendation systems for life-time value optimization. In *Proceedings of the 24th International Conference on World Wide Web*, 1305–1310. ACM.

Thomas, P. S., and Barto, A. G. 2012. Motor primitive discovery. In *Procedings of the IEEE Conference on Development and Learning and Epigenetic Robotics*, 1–8.

Thomas, P. S., and Brunskill, E. 2017. Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines. *arXiv preprint arXiv:1706.06643*.

Thomas, P.; Dann, C.; and Brunskill, E. 2018. Decoupling gradient-like learning rules from representations. In *International Conference on Machine Learning*.

Thomas, P. 2014. Bias in natural actor-critic algorithms. In *International Conference on Machine Learning*, 441–448.

Tsitsiklis, J., and Roy, B. 1983. An analysis of temporal-difference with function approximation. *IEEE Trans. Autom. Control* 42(5):834–836.

Tucker, G.; Bhupatiraju, S.; Gu, S.; Turner, R. E.; Ghahramani, Z.; and Levine, S. 2018. The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031*.

Wang, C.; Chen, X.; Smola, A. J.; and Xing, E. P. 2013. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*.

Wu, C.; Rajeswaran, A.; Duan, Y.; Kumar, V.; Bayen, A. M.; Kakade, S.; Mordatch, I.; and Abbeel, P. 2018. Variance reduction for policy gradient with action-dependent factorized baselines. *arXiv preprint arXiv:1803.07246*.