Dealing with Sparse Rewards in Continuous Control Robotics via Heavy-Tailed Policies

Souradip Chakraborty1, Amrit Singh Bedi1, Alec Koppel2, Pratap Tokekar1, Dinesh Manocha1

Abstract—In this paper, we present a novel Heavy-Tailed Stochastic Policy Gradient (HT-PSG) algorithm to deal with the challenges of sparse rewards in continuous control problems. Sparse reward is common in continuous control robotics tasks such as manipulation and navigation, and makes the learning problem hard due to non-trivial estimation of value functions over the state space. This demands either reward shaping or expert demonstrations for the sparse reward environment. However, obtaining high-quality demonstrations is quite expensive and sometimes even impossible. We propose a heavy-tailed policy parametrization along with a modified momentum-based policy gradient tracking scheme (HT-SPG) to induce a stable exploratory behavior to the algorithm. The proposed algorithm does not require access to expert demonstrations. We test the performance of HT-PSG on various benchmark tasks of continuous control with sparse rewards such as 1D Mario, Pathological Mountain Car, Sparse Pendulum in OpenAI Gym, and Sparse MuJoCo environments (Hopper-v2). We show consistent performance improvement across all tasks in terms of high average cumulative reward. HT-SPG also demonstrates improved convergence speed with minimum samples, thereby emphasizing the sample efficiency of our proposed algorithm.

I. INTRODUCTION

Reinforcement learning (RL) has been employed with great success in several continuous control robotic tasks such as grasping [1], motion planning [2], and navigation [3]. The key underlying idea in RL is to explore in an unknown environment, collect rewards, and then move to maximize the reward collection. In the real world, designing dense rewards is challenging for robotic tasks such as manipulation and navigation [4]. Reward engineering for robotic tasks is difficult due to complex state space representations and usually requires manually-designed perception systems of the environment [5]. Hence, it makes sense to work directly with naturally specified sparse rewards [6]–[8]. For example, it is much easier to specify a binary reward (1 for successful completion of a task and 0 otherwise) than to come up with a dense reward structure. However, learning with sparse rewards is much more challenging because it results in the Hessian of the value function with respect to policy parameters being ill-conditioned. It also imposes the need to sample multiple trajectories in order to have a nontrivial estimate of the value function, which is sample inefficient [9].

Furthermore, learning from sparse rewards in continuous control robotic tasks becomes even more challenging (as mentioned in Fig. 1) because they exhibit continuous state and action spaces. For instance, in manipulation tasks, joint angles of robots are continuous, and in navigation tasks, the pose of robots and control inputs are continuous. RL in continuous control problems is hard because it’s hard to compute expectations with respect to continuous state distributions and continuous actions to evaluate value functions [10].

The issue of sparse rewards is usually dealt with in literature either through either reward shaping [12]–[14] or utilizing expert demonstrations [7], [15]–[18]. Intuitively, both of these approaches try to induce effective exploration into the sparse reward environment by providing surrogate rewards. Reward shaping approaches modify the reward to motivate the agent to visit unexplored states in the environment. For instance, authors in [19] induce such behaviors via intrinsic curiosity, and [20] utilizes information to motivate the exploration. Another line of work utilizes...
expert’s demonstrations to learn effectively in sparse reward environments [7], [8], [14], [21]. The main idea here is to either use available demonstration to clone an expert’s behavior (imitation learning) or just utilize demonstrations to provide additional rewards to guide the exploration [7], [8]. But the major limitation of these approaches depends on the quality of the expert demonstrations. If the demonstrations are not sub-optimal or not good, these approaches fail badly. Apart from that, obtaining a high-quality demonstration is quite expensive, especially in robotic environments [1].

In contrast to existing approaches to deal with sparse reward settings, in this work, we follow a different route and take motivation from the global convergence results in tabular MDP settings [22]. A crucial enabler for learning global optimal policies in [22] is the idea of persistent exploration, which helps to implicitly induce sufficient exploration in the state space. This ensures that the probability of taking any action in a given state is always non-zero, which would help to visit the complete state space and look for rewards. Recently, authors in [23] have extended the idea of persistent exploration to continuous spaces and have proposed to utilize heavy-tailed policies to avoid convergence of policy gradient methods to spurious local maxima. Taking motivation from [23], we ask the following question

“Can heavy-tailed policies make model-free RL sample efficient for practical robotics tasks that involve sparse reward structure without any expert demonstrations?”

We answer this question in affirmative in this paper and propose to utilize heavy-tailed policies (such as Cauchy) for policy parametrization along with a modified momentum-based policy gradient tracking to deal with the sparsity in reward. These heavy-tailed distributions appear heavily in fractal geometry [24], [25], finance [26], [27], pattern formation in nature [28], and networked systems [29], but has not been well investigated in RL framework. Intuitively, heavy-tailed policies induce an implicit exploration behavior into the trained policies (because of the high probability of taking tail actions), and help to learn effectively in sparse environments even without any expert demonstrations. We summarize the main contributions of this paper as follows:

- We propose a novel way to deal with sparse reward environments to train a policy in continuous state-action space environments. Our approach is fundamentally different because we introduce heavy-tailed policy parametrization and avoid using expert demonstrations, which is a common practice in the existing literature. This provides a way to work with sparse reward environments without any reward shaping or demonstrations, which is very difficult otherwise. Additionally, our formulation is flexibly designed to efficiently incorporate prior demonstrations as well, if available.

- We observed that just replacing Gaussian policy parametrization with heavy-tailed (Cauchy) parametrization results in unstable behavior during the training. Hence, we propose a modified version of the momentum-based tracking method proposed in [30] to control the variance of the stochastic gradient estimates.

- Finally, we show the efficacy of the proposed algorithm on various continuous control task problems. The proposed algorithm shows consistent performance improvement over a variety of benchmark problems (cf. Sec. [IV]).

**Reward Shaping:** Reward shaping is the most intuitive way to deal with sparse rewards. The idea was first appeared in [13] and further developed in recent works [14], [19], [20], [31]. The main idea revolves around intrinsic curiosity [19] and information gain based shaping [20]. Besides being simple, these methods come with the challenge of designing the additional reward functions which require expert supervision and demonstrations which are expensive. Additionally, it also induces expert-specific bias to the learning systems which ultimately leads the agent to explore only certain parts of the environment hindering the overall improvement.

**Imitation Learning:** Another line of work focuses on cloning an expert behavior called imitation learning (IL) [32]. Inverse reinforcement learning (IRL) is one way to do IL by extracting rewards from the given set of expert’s trajectories for a given task [16], [17]. This issue of reward estimation was resolved by generative adversarial imitation learning (GAIL) algorithm by utilizing a discriminator to provide reward functions [33]. But the main drawback of IL-based approaches is that they do not utilize the feedback from the environment and behave according to the policies learned from demonstrations. Our approach in this work is fundamentally different, and we propose a method that works without demonstrations and can also incorporate prior demonstrations efficiently in the methodology, is available.

**Learning from Demonstration:** The idea here is to utilize expert’s demonstrations to guide the standard learning procedure in RL algorithms [7], [8], [18], [34], [35]. Authors in [21], [36] proposed to include expert demonstrations to replay buffers and utilize them to accelerate the learning. The authors in [7] proposed an effective way to combine information from expert’s policy to guide the exploration in the policy gradient algorithms. Mainly, the original reward function is modified to also include a term that accounts for the distance of current policy to the expert’s policy. But as mentioned previously, the major drawback here is also their dependence upon the availability of demonstrations, which are hard to get in practice for continuous control problems. For instance, expert’s demonstrations in [8] are obtained by running TRPO with dense rewards and then later used to train a policy with sparse rewards in the same environment. This could be difficult to achieve in practice. Therefore, we propose to modify the policy parametrization in continuous control environments to induce the required exploration in the learning procedure.

**Heavy-Tailed Policy Parametrization:** The idea of parametrizing policies via heavy-tailed distribution has appeared in the reinforcement learning literature [23], [37]. The authors in [37] proposed to utilize beta distribution for policy parametrization but are restricted to dense reward
structure environments. Authors in [23] have focused on the
development of heavy-tailed policy gradient to avoid
convergence to local maxima and do not explicitly deal
with sparse rewards. This work focus on sparse reward
continuous control environments and extensive experimental
evaluations to support the importance of heavy-tailed policy
parametrization.

The paper is organized as follows. We start with the prob-
lem formulation in Sec. II followed by proposed algorithm
in Sec. III. We present experimental results in Sec. IV, and
then conclude the paper in Sec. V.

II. MARKOV DECISION PROBLEMS WITH SPARSE
REWARDS

When we formulate the continuous control robotics prob-
lems via reinforcement learning (RL), an autonomous robot
interacts with the underlying environment by visiting differ-
ent states in the state space \( S \). It starts from a particular
state \( s \in S \), selects an action \( a \in A \) from the action
space, and then transitions to another state \( s' \in S \) in
the state space. The next state is assumed to follow an
unknown Markov transition density \( P(s'|s,a) \). Then after
reaching state \( s' \), agent received an instantaneous reward
of \( r(s,a) \) which quantifies the merit of decision \( a \) at state
\( s \). Mathematically, this framework is defined as Markov
Decision Process (MDP) given by \( \mathcal{M} := \{ S, A, P, r, \gamma \} \),
where \( \gamma \in (0,1) \) is the discount factor which decides
the importance of future rewards for each instant. The state space
\( S \subseteq \mathbb{R}^q \) and actions space \( A \subseteq \mathbb{R}^p \) is continuous. Hence,
we hypothesize that the agent selects actions \( a_t \sim \pi(\cdot|s) \) over
a time invariant distribution denoted by \( \pi(\cdot|s) \) for a given state
\( s \). The distribution \( \pi(\cdot|s) \) is called a policy which controls
the probability of taking a particular action \( a \) in given state
\( s \). The goal in the RL problem is to search for policy
\( \pi(\cdot|s) \) such that the average cumulative reward return (called value)
is maximized given by:

\[
V^{\pi}(s) = \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_t \sim \pi(\cdot|s_t) \right],
\]  

(1)

where \( V^{\pi}(s) \) is the value function with respect to
state \( s \), and \( s_0 \) denotes the initial state along a trajectory
\( \{s_t, a_t, r(s_t, a_t)\}_{t=0}^{\infty} \). Similar to fixing the initial state \( s_0 \),
if we fix initial action as well \( a_0 = a \), the we can write the
action-value function as

\[
Q^{\pi}(s,a) = \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a, a_t \sim \pi(\cdot|s_t) \right].
\]  

(2)

We note that the expectation in (1) is with respect to the
product measure of policy \( a_t \sim \pi(\cdot|s_t) \) and state transition
density \( s_{t+1} \sim P(\cdot|s_t, a_t) \). The selection of action \( a_t \) would
control the possibility of visiting different state in the state
space \( S \), and hence also responsible for exploring the state
space. This also becomes important because in this work,
we are specifically interested in environments where the rewards
are sparse (cf. Sec. IV). By sparse rewards we mean that
they are available once in a while (see Fig. 2(a)) or there are
high reward states available (see Fig. 2(b)) but too far in the
state space. Learning a good policy in such environments is
difficult task and that is the focus of this work. Hence, the
goal here is to find a policy \( \pi \) such that

\[
\max_{\pi} V(s_0),
\]  

(3)

with \( s_0 \sim \rho_0(s) \) and \( \rho_0(s) \) being an arbitrary initial state
distribution. Since, \( \pi \) here is a policy distribution, it becomes
intractable to solve the problem (3) in it general form and we
keep our focus to a search over parameterized class of
policies denoted by \( \pi_{\theta}(\cdot|s) \) where \( \theta \) is the parameter
which defined the policy distribution completely. So now, our search
over distributions boil down to search over set of parameters
\( \theta \) [38] given by

\[
\max_{\theta} J(\theta) := V^{\pi_{\theta}}(s_0),
\]  

(4)

with \( s_0 \sim \rho_0(s) \). We note that the problem in (4) is non-
convex with respect to optimization variable \( \theta \). Next, we
derive the standard policy gradient algorithm to solve the
problem in (4) and discuss challenges in the sparse reward
settings.

A. Policy Gradient Algorithm

The policy gradient (PG) algorithm is a well known
technique to perform search for optimal parameters \( \theta \) in
parameter space \( \theta \in \mathbb{R}^d \). The key result which enables us
to write policy gradient for the complicated objective in (4)
is the Policy Gradient Theorem [38], which states that the
gradient of \( J(\theta) \) with respect to \( \theta \) can be written as

\[
\nabla J(\theta) = \int_{S \times A} \gamma^t \cdot P(s_k = s \mid s_0, \rho_\theta) \times
\nabla \pi_{\theta}(a \mid s) \cdot Q_{\pi_{\theta}}(s, a) \cdot dsda
\]

(5)

\[
= \frac{1}{1 - \gamma} \int_{S \times A} (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \cdot P(s_k = s \mid s_0, \rho_\theta) \times
\nabla \pi_{\theta}(a \mid s) \cdot Q_{\pi_{\theta}}(s, a) \cdot dsda
\]

(6)

and the expectation in (6) is over \( (s, a) \sim \rho_\theta(\cdot, \cdot) \) where
\( \rho_\theta(s,a)=\rho_{\pi_{\theta}}(s) \cdot \pi_{\theta}(a \mid s) \) now denotes a valid probability
distribution function also called as discounted state-action
occupancy measure over continuous state and action spaces.
From the expressions of \( \rho_\theta(s,a) \) note that the selection
of policy class has a significant affect on the eventually
occupancy measure induced. In tabular MDP settings, to
make sure the convergence to global optimal, an assumption
of persistent exploration is needed [22], which is satisfied
by making sure that \( \pi(a \mid s) > 0 \) for all \( s \) and \( a \). We
remark that satisfying such assumption automatically takes
care of the fact that we explore almost all parts of the
state space because the probability of reaching any other state \( s' \) is not zero because of \( \pi(a \mid s) > 0 \). Therefore, in tabular MDP, things work well even in the sparse reward settings. In contrast, in continuous action spaces, imposing such assumption \( \pi(a \mid s) > 0 \) on the policy distribution \( \pi(a \mid s) > 0 \) would violate the integrable assumption of probability distributions, and hence is not a valid assumption. So the induced exploration in the state space is mostly controlled by the policy distribution class we choose for parametrization. The standard parametrization class which is widely used in the literature is Gaussian [7], [8], [39], [40], and given as follows.

**Example 1 (Gaussian Parametrization):** We assume that the policy \( \pi_\theta(a \mid s) \) is a Gaussian distribution given by

\[
\pi_\theta(a \mid s) = \mathcal{N}(\varphi(s)^\top \theta, \sigma^2),
\]

(7)

where \( \varphi \) controls the mean of the Gaussian, \( \varphi(s) \) denotes the states feature representation \( \varphi : S \rightarrow \mathbb{R}^d \) with \( d \ll q \), and \( \sigma^2 \) is fixed variance. We can make \( \sigma \) as a parameter as well we avoid that for the sake of explanation simplicity.

Now, specifically for sparse reward settings, one major drawback of Gaussian parametrization for policy is its tendency to take actions close to mean value. This feature would restrict the model transition to a state \( s' \) which is farther from current state \( s \) due to action selection \( a \sim \mathcal{N}(\varphi(s)^\top \theta, \sigma^2) \) close to mean value. This induces a limited exploration for the algorithm, and it fails to learn in sparse reward environments. To deal with this issue, different techniques such as information maximization [20] and learning from demonstrations [7] are proposed. But the main disadvantages of such techniques are that entropy regularization required the estimation of the density function of occupancy which is quite expensive, and prior demonstrations could be quite bad and lead to completely irrelevant policies. Hence, to deal with such issues, instead of proposing any augmentation to existing techniques to handle sparse rewards, we resort to a completely novel approach and proposed to utilize heavy-tailed distributions to parameterize the policy \( \pi_\theta \). We explain this idea in detail in the next section.

### III. Proposed Heavy-Tailed Stochastic Policy Gradient for Sparse Rewards

In this section, we present the main idea of this work and develop a stable heavy-tailed stochastic policy gradient descent algorithm to deal with sparse reward settings.

#### A. Heavy-Tailed Policy Parametrization

As a first step towards developing such an algorithm, we propose to parameterize the policy by a class of heavy-tailed distributions. An example of heavy such parametrization is Cauchy distribution which is given by

\[
\pi_\theta(a \mid s) = \frac{1}{\sigma \pi(1 + ((a - \varphi(s)^\top \theta)/\sigma)^2)},
\]

(8)

where \( \sigma \) is the fixed scaling parameter. Other heavy-tailed distributions include the Extreme value distribution, Weibull distribution, log-normal distribution, Student’s t distribution, Generalized Gaussian distribution, etc. The Laplace distribution has also fatter tails than the Gaussian distribution. In the financial literature, such distributions have been associated with the phenomenon of “black swan” events [26], [27].

With the policy parametrization specified, next goal is to compute the policy gradient mentioned in (6). But the challenge is the transition model dynamics are assumed to me unknown so it is not possible to evaluate \( \nabla J(\theta) \) in closed form. So we take stochastic approximation approach and evaluate the stochastic gradient estimate. To write that, consider a randomized horizon \( T_k \sim \text{Geom}(1 - \gamma^{1/2}) \) with trajectory sample \( \{(s_0, a_0) \cdots (s_{T_k}, a_{T_k})\} =: \xi_k(\theta_k) \), then stochastic gradient can be written as

\[
\nabla J(\theta_k, \xi_k(\theta_k)) = \sum_{t=0}^{T_k} \gamma^{t/2} \nabla \log \pi_{\theta_k}(a_t \mid s_t),
\]

(9)

where \( \nabla J(\theta_k, \xi_k(\theta_k)) \) denotes the unbiased estimator of gradient \( \nabla J(\theta_k) \) at \( \theta_k \) (see [23, Lemma 1] for proofs) and \( \xi_k(\theta_k) \) denotes the randomness in the estimate at \( k \). Note the variable horizon length of the trajectories in [9] which is important to obtain an unbiased estimator. Otherwise, a fixed horizon length estimators where \( T_k = H \) for all \( k \) (as in [40], [42]), results in a bias-variance tradeoff for gradient estimate [43]. Further, note the summation over two indexes in [9] \( t \) corresponds to the rollout trajectory, and \( \tau \) collects score function till \( t \) from the starting. With the stochastic gradient defined in [9], the heavy tailed stochastic policy gradient iterate is given by

\[
\theta_{k+1} = \theta_k + \eta \nabla J(\theta_k, \xi_k(\theta_k)),
\]

(10)

where \( \eta > 0 \) denotes the step size. Note that the score function to evaluate the stochastic policy gradient in [10] is parameterized by a heavy-tailed policy due to the sparse rewards settings considered in this work. While this selection...
Algorithm 1 Heavy-Tailed Stochastic Policy Gradient (HT-SPG)

1: Initialize: Initial parameter \( \theta_0 \), momentum parameter \( \beta \), discount factor \( \gamma \), step-size \( \eta \), and gradient estimate \( g_0 = 0 \)
2: Repeat for \( k = 1, \ldots \)
3: Sample two trajectories \( \xi_k(\theta_k) \) and \( \xi_k(\theta_{k-1}) \) of length \( T_k \sim \text{Geom}(1 - \gamma^{1/2}) \) using policies \( \pi_{\theta_k} \) and \( \pi_{\theta_{k-1}} \), respectively
4: Estimate \( \nabla J(\theta_k, \xi_k(\theta_k)) \), \( \nabla J(\theta_{k-1}, \xi_k(\theta_{k-1})) \) via \( g_k \)
5: Update \( \theta_{k+1} = \theta_k + \eta g_k \)
6: \( k \leftarrow k + 1 \)
7: Until Convergence

Return: \( \theta_k \)

of heavy tailed parametrization serves the purpose of selecting actions far from mean and induce sufficient exploration into the algorithm behavior, this exhibits a downside as well. The resulting algorithm tends to be unstable to heavy tails and probability of taking extreme actions. We mitigate this issue by introducing a momentum based gradient tracking to the proposed algorithm which is the focus of next subsection.

B. Stable Heavy-Tailed Stochastic Policy Gradient Algorithm

The direct replacement of Gaussian policy parametrization with heavy-tailed policy parametrization results in an unstable behavior for the algorithm because the stochastic gradient estimates exhibit high variations from one sample to other. To deal with this issue, we need to invoke the idea of introducing momentum to stochastic gradient (SG) updates which has been successfully used in other machine learning approaches [44]. Hence, we replace the update in (10) as follows

\[
g_k = (1 - \beta)g_{k-1} + \beta \nabla J(\theta_k, \xi_k(\theta_k)), \quad (11)
\]

\[
\theta_{k+1} = \theta_k + \eta g_k, \quad (12)
\]

where \( \beta \) is the tuning parameter and update in (11) is called the momentum update. Note that for a small \( \beta \) (say \( \beta = 0.2 \)) would results in utilizing the exponential average of past gradients rather than just considering the current stochastic gradient \( \nabla J(\theta_k, \xi_k(\theta_k)) \). This update is popular in the SG descent literature and achieves significant improvement empirically as compared to special case of \( \beta = 1 \) [13 from [30] but does not result in theoretical gain. To address this issue, the authors in [30] have proposed a modified momentum based gradient tracking which result in provable variance reduction. With motivation from results in [30], we propose a novel gradient tracking scheme presented next for stochastic policy gradients with heavy-tailed policy parametrization as

\[
g_k = (1 - \beta)g_{k-1} + \beta \nabla J(\theta_k, \xi_k(\theta_k)) + (1 - \beta)(\nabla J(\theta_k, \xi_k(\theta_{k-1})) - \nabla J(\theta_{k-1}, \xi_k(\theta_{k-1}))), \quad (13)
\]

\[
\theta_{k+1} = \theta_k + \eta g_k, \quad (14)
\]

where \( \nabla J(\theta_{k-1}, \xi_k(\theta_{k-1})) \) denotes the another stochastic gradient evaluated at instant \( k \) with policy parameter \( \theta_{k-1} \).

The explicit expression is given by

\[
\nabla J(\theta_{k-1}, \xi_k(\theta_{k-1})) = \sum_{i=0}^{T_k} \gamma^{i/2}r(s_i, a_i) \left( \sum_{\tau=0}^{t} \nabla \log \pi_{\theta_{k-1}}(a_t' \mid s_t') \right),
\]

where \( \xi_k(\theta_{k-1}) := \{s_i', a_i' \} \sum_{i=0}^{T_k} \) denotes the trajectory generated bu using policy parameter \( \theta_{k-1} \) but at instance \( k \).

Note that there will be two Monte Carlo trajectories required to perform the update in (13). We remark an important difference of update in (13) to the gradient tracking proposed in [30]. The momentum step in [30, Eq. (2)] would require the use of \( \nabla J(\theta_{k-1}, \xi_k(\theta_{k-1})) \) (to keep the stochastic quantity same) instead of \( \nabla J(\theta_{k-1}, \xi_k(\theta_{k-1})) \) which we propose to use in this work. The use of term \( \nabla \log \pi(\theta_{k-1}, \xi_k(\theta_{k-1})) \) has been proposed in the literature for reinforcement learning settings in [40] along with importance sampling weight adjustments to take care of the distributional shift which occurs due to the dependence of stochastic trajectory \( \xi_k(\theta_{k}) \) on \( \theta_k \). Next, we intuitively explain why it makes sense to use the update in (13), and it helps to reduce the variance of stochastic gradients, and hence results in a stable algorithm.

To understand it, let us consider the stochastic error introduced to the original gradient due to (13) as \( \epsilon_k = g_k - \nabla J(\theta_k) \). We note that \( \epsilon_k \) defines the stochastic error in the gradient direction to perform the ascent update, and if we show that \( \mathbb{E}[\|\epsilon_k\|^2] \) has a decreasing behavior with respect to \( k \), this implies that the proposed momentum based update has resulted in variance reduction. Let us look at the explicit expression of \( \epsilon_k \) as

\[
\epsilon_k = (1 - \beta)\epsilon_{k-1} + \beta(\nabla J(\theta_k, \xi_k(\theta_k)) - \nabla J(\theta_k)) + (1 - \beta)(\nabla J(\theta_{k-1}, \xi_k(\theta_{k-1})) - \nabla J(\theta_{k-1}, \xi_k(\theta_{k-1}))) + (1 - \beta)(\nabla J(\theta_k) - \nabla J(\theta_{k-1})). \quad (16)
\]

Next, note that it is the second, third, and fourth term on the right hand side of (16) which we need to control. We can easily control the second term on the right hand side of (16) by keeping \( \beta \) small. From the smoothness of \( J \), we know that \( \|\nabla J(\theta_k) - \nabla J(\theta_{k-1})\| \approx \mathcal{O} (\eta \|\theta_k - \theta_{k-1}\|) \) which can be controlled by step size \( \eta \). The only remaining term is \( \|\nabla J(\theta_k, \xi_k(\theta_k)) - \nabla J(\theta_{k-1}, \xi_k(\theta_{k-1}))\| \) which can also be assumed \( \approx \mathcal{O} (\eta \|\theta_k - \theta_{k-1}\|) \) when \( \theta_k \) and \( \theta_{k-1} \) are close to each other. This is possible because of the dependence of trajectories \( \xi_k(\cdot) \) on \( \theta \) which is not the case in [30]. Therefore, by controlling \( \beta \) and \( \eta \), it is possible to develop a stable algorithm with heavy-tailed policy parametrizations. We summarize the algorithm steps in Algorithm 1.

Further, we extensively test the empirical performance of the proposed algorithm on different sparse environments in
next section and show the performance benefits achieved in practice. We defer the theoretical analysis of the proposed algorithm to future scope of this work.

IV. EXPERIMENTS

In this section, we proceed to perform extensive experimental validation of the proposed ideas in this work. First, we perform a detailed analysis and performance comparison of proposed stable Heavy-Tailed Stochastic Policy Gradient Algorithm (HT-SPG) in classic continuous control reinforcement learning environments with sparse and complex rewards such as 1D Mario [41], Pathological Mountain Car (cf. Fig. 2(b)), and Sparse Pendulum [11]. Second, to test the performance on complicated continuous environments, we consider the Sparse MuJoCo environments namely Hopper-v2 as done in [8]. Finally, we compare the performance of HT-SPG against state-of-the-art LOGO algorithm [8] and show consistent performance improvements under complex and challenging settings.

Importance of Policy Parametrization: Before discussing the main experimental results, we start by demonstrating (see Fig. 3) the limitations of light-tail policy parametrization and emphasize the importance of using heavy-tail distributions such as Cauchy for policy parametrization. Fig. 3 shows the average cumulative reward return for different policy parametrizations in a 1D Mario environment. We demonstrate that Cauchy distribution-based policy is able to achieve the highest reward return in the most sample-efficient manner. This is mainly due to the better exploratory behavior achieved by the Cauchy-based policy as compared to other policies. Hence, we will be using Cauchy policy parametrizations for the rest of the experiments. We detail the different environment settings as follows.

A. Learning Without Demonstrations

In this subsection, we run experiments in sparse reward environments and compare against other state-of-the-art algorithms which operate without any access to expert’s demonstrations. The details of environments are as follows.

- 1D Mario Environment: This is a one-dimensional, discrete-time, continuous state and action space environment (cf. Fig. 2(b)). The state space is \( s \in [0, 1] \) and action space is \( a \in [-0.1, 0.1] \). The goal is to collect the coin place at \( s = 0 \) and agent can move in right or left by any amount between \([-0.1, 0.1]\). The reward is defined as \( r(s_t, a_t) = \begin{cases} 1, & s_t + a_t < 0 \\ 0, & \text{otherwise} \end{cases} \) and transition model as \( s_{t+1} = \min\{1, \max\{0, s_t + a_t\}\} \). We note that reward is sparse because it is 1 only at the goal, otherwise it’s zero in the full state space. Each of the episodes are initialized at \( s_0 = 0.9 \).

- Pathological Mountain Car: This is a continuous state action space environment with misaligned goal (see Fig. 2(b)). The reward is distributed widely over the state space with a low reward state and a bonanza top a higher hill. The low reward state is at \( s = 2.667 \) with a reward of 10 and a high reward state farther apart at \( s = -4.0 \) with 500 units of reward. For PMC, we consider a reward structure in which the amount of energy expenditure, i.e., the action squared, at each time-step is negatively penalized, as given by

\[
 r(s, a) = -a_t^2 I_{\{-4.0 < s < 3.709, s \neq 2.667\}} + (500 - a_t^2) I_{\{s = -4.0\}} + (10 - a_t^2) I_{\{s = 2.667\}}. \tag{17}
\]

Here, the action is denoted as \( a \) and is a one-dimensional scalar which represents the speed of the vehicle \( \dot{s} \).

- Sparse Inverted Pendulum: This is an unstable inverted pendulum (pole) attached to a cart (see Fig. 1), and the goal is to keep the pole upright [11]. An agent can move the cart to the left or right via applying a discrete force of \( \pm 1 \) along the horizontal axis of the cart. It is exactly like Open AI gym’s Pendulum-v0, but with sparse rewards.

We run the proposed algorithm HT-SPG in the above-mentioned environments and compare with other state-of-the-art existing algorithms with light-tailed policy parametrization (Gaussian) such as RPG [39], and STORM-PG [40]. There is a state-of-the-art algorithm to solve continuous control problems without any demonstrations. We present the results in Fig. 4 where RPG (Cauchy) denotes RPG algorithm with Cauchy policy parametrization. It is included to show that just replacing Gaussian (RPG (Gaussian)) with Cauchy is not sufficient to achieve the desired performance, and it results in unstable behavior which exhibits high variance in the reward returns. This issue is corrected by using the momentum-based tracking in HT-SPG. In all
In this section, we consider the complex sparse MuJoCo environments of Hopper (see Fig. 1) and test the performance of the proposed HT-SPG algorithm. We compare it with the state-of-the-art LOGO algorithm [8]. The state and action spaces for these environments are no longer scalar anymore and require us to deal with multi-variate distributions for the policy parametrizations. State-space is 12-dimensional, action space is 3-dimensional linear reward for forward progress and a quadratic penalty on a joint effort to produce the reward with a bonus of +1 for being in a non-terminal state. The episodes when the hopper fell over, which was defined by thresholds on the torso height and angle. The sparsity in reward structure is obtained by reducing the events at which reward feedback is provided. Specifically, we provide a reward of +1 only after the agent moves forward over 2 units from its initial position. We present the performance of HT-SPG as compared to the LOGO algorithm in Fig. 5(a). Since the performance of LOGO was optimized to operate with demonstrations, we considered the same learning environment with demonstrations for the proposed HT-SPG algorithm as well. We note that the proposed algorithm is able to outperform LOGO by a significant margin and exhibit better sample efficiency. In Fig. 6(a), we present the snapshots of the behavior of final policy learned by the proposed algorithm.

V. CONCLUSION, LIMITATIONS, AND FUTURE WORK

In this work, we proposed a novel approach to deal with sparse reward in continuous control robotics task. Instead of relying on reward shaping or seeking information from expert’s demonstrations, we utilize heavy-tailed policy parametrizations along with momentum based gradient tracking to learn in sparse robotics environments. We prove the efficacy of the proposed ideas on various robotics tasks including inverted pendulum of OpenAI Gym and Hopper-v2 of MuJoCo environments. The main limitation of the current approach is that we cannot prove any theoretical convergence guarantees for the proposed approach. For future work, it would be interesting to look at the sample complexity analysis of the proposed HT-PSG.
Fig. 6: We note that the proposed heavy-tailed policy (top row) was able to better learn and hop the robot for 500 number of steps while the Gaussian policy (bottom row) could only do it for 50 steps. Note: Model trained on the 400 iteration has been used for evaluation.

**REFERENCES**

[1] O. Kilinc and G. Montana, “Reinforcement learning for robotic manipulation using simulated locomotion demonstrations,” *Machine Learning*, pp. 1–22, 2021.

[2] M. Everett, Y. F. Chen, and J. P. How, “Motion planning among dynamic, decision-making agents with deep reinforcement learning,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[3] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and J. Luo, “Robot navigation in crowded environments using deep reinforcement learning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5671–5677.

[4] A. Amini, I. Gilitschenski, J. Phillips, J. Moseykro, R. Banerjee, S. Karaman, and D. Rus, “Learning robust control policies for end-to-end autonomous driving from data-driven simulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.

[5] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, “End-to-end robotic reinforcement learning without reward engineering,” *arXiv preprint arXiv:1904.07854*, 2019.

[6] G. Schoettler, A. Nair, J. Luo, S. Bahl, I. Aparicio Ojea, E. Solowjow, and S. Levine, “Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5548–5555.

[7] B. Kang, Z. Jie, and J. Feng, “Policy optimization with demonstrations,” in *International conference on machine learning*. PMLR, 2018, pp. 2469–2478.

[8] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakhattei, “Reinforcement learning with sparse rewards using guidance from offline demonstration,” *arXiv preprint arXiv:2202.04628*, 2022.

[9] P. Rauber, A. Ummadisingu, F. Mutz, and J. Schmidhuber, “Reinforcement learning in sparse-reward environments with hindsight policy gradients.” *Neural Computation*, vol. 33, no. 6, pp. 1498–1553, 2021.

[10] H. Van Hasselt and M. A. Wiering, “Reinforcement learning in continuous action spaces,” in *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. IEEE, 2007, pp. 272–279.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

[12] N. Botteghi, B. Sirmacek, K. A. Mustafa, M. Poel, and S. Stramigioli, “On reward shaping for mobile robot navigation: A reinforcement learning and slam based approach,” *arXiv preprint arXiv:2002.04109*, 2020.

[13] M. J. Matarić, “Reward functions for accelerated learning.” in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 181–189.

[14] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Overcoming exploration in reinforcement learning with demonstrations,” in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[15] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rotherhörl, T. Lampe, and M. Riedmiller, “Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards,” *arXiv preprint arXiv:1707.08817*, 2017.

[16] A. Y. Ng, S. J. Russell et al., “Algorithms for inverse reinforcement learning.” in *ICml*, vol. 1, 2000, p. 2.

[17] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey et al., “Maximum entropy inverse reinforcement learning,” in *AAai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[18] G. Libardi, Gabriele anfd De Fabritius and S. Dittert, “Guided exploration with proximal policy optimization using a single demonstration,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 6611–6620.

[19] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven exploration by self-supervised prediction,” in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[20] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, “Vime: Variational information maximizing exploration,” *Advances in neural information processing systems*, vol. 29, 2016.

[21] T. Hester, M. Vecerik, O. Pietquin, M. Lanktot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sundararaj, I. Osband et al., “Deep q-learning from demonstrations,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[22] J. Mei, C. Xiao, C. Szepesvari, and D. Schuurmans, “On the global convergence rates of softmax policy gradient methods,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 6820–6829.

[23] A. S. Bedi, A. Parajjil, J. Zhang, M. Wang, and A. Koppel, “On the sample complexity and metastability of heavy-tailed policy search in continuous control,” *arXiv preprint arXiv:2106.08414*, 2021.

[24] J. E. Hutchinson, “Fractals and self similarity,” *Indiana University Mathematics Journal*, vol. 30, no. 5, pp. 713–747, 1981.

[25] B. B. Mandelbrot and B. B. Mandelbrot, *The fractal geometry of nature*. WH freeman New York, 1982, vol. 1.

[26] N. N. Taleb, *The black swan: The impact of the highly improbable*. Random house, 2007, vol. 2.

[27] J. B. Taylor and J. C. Williams, “A black swan in the money market,” *American Economic Journal: Macroeconomics*, vol. 1, no. 1, pp. 58–83, 2009.

[28] D. Avinir, O. Biham, D. Lidar, and O. Malcai, “Is the geometry of nature fractal?” *Science*, vol. 279, no. 5347, pp. 39–40, 1998.

[29] A. Clauset, C. R. Shalizi, and M. E. Newman, “Power-law distributions in empirical data,” *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.

[30] A. Cutkosky and F. Orabona, “Momentum-based variance reduction in non-convex sgd,” *arXiv preprint arXiv:1905.10018*, 2019.

[31] M. Plaﬄert, R. Houthooft, P. Dharwail, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, “Parameter space noise for exploration,” *arXiv preprint arXiv:1706.01905*, 2017.

[32] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, “Imitation learning: A survey of learning methods,” *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[33] J. Ho and S. Ermon, “Generative adversarial imitation learning,” *Advances in neural information processing systems*, vol. 29, 2016.

[34] S. Schaal, “Learning from demonstration,” *Advances in neural information processing systems*, vol. 9, 1996.

[35] J. Chen and W. Xu, “Policy gradient from demonstration and curiosity,” *arXiv preprint arXiv:1709.10087*, 2017.

[36] A. A. J. Karaman and D. Rus, “Learning robust control policies for end-to-end robotic reinforcement learning without reward engineering,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 2, 2019.

[37] P.-W. Chou, “The beta policy for continuous control reinforcement learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 8, no. 1, pp. 2–9, 2014.

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

[39] C. E. Alippi, L. Bassi, and A. G. Barto, “On the role of curiosity in reinforcement learning,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 857–873, 2019.

[40] H. Yuan, X. Lian, J. Liu, and Y. Zhou, “Stochastic recursive momentum for policy gradient methods,” *arXiv preprint arXiv:2003.04702*, 2020.
[41] G. Matheron, N. Perrin, and O. Sigaud, “The problem with ddpg: understanding failures in deterministic environments with sparse rewards,” 2019.

[42] M. Papini, D. Binaghi, G. Canonaco, M. Pirotta, and M. Restelli, “Stochastic variance-reduced policy gradient,” in ICML, 2018, pp. 4026–4035.

[43] J. Baxter and P. L. Bartlett, “Infinite-horizon policy-gradient estimation,” Journal of Artificial Intelligence Research, vol. 15, pp. 319–350, 2001.

[44] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.