REINFORCEMENT LEARNING: FROM TEMPORAL TO SPATIAL VALUE DECOMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Value function, lying in the heart of reinforcement learning (RL), is defined as expected total reward to capture the long-term dependency between current state/action and delayed reward. Following this definition, it’s natural to derive a temporal decomposition for value function as a sum of discounted reward on different time steps. Most existing RL algorithms estimate value function based on this temporal decomposition.

Apart of traditional temporal decomposition, in this paper, we explore a novel spatial decomposition for value function. In our formulation, value function \( Q(s, a) \) is a weighted sum of reward \( R(s') \), and the corresponding weight is defined as the discounted visiting frequency \( d(s, a, s') \). Here, \( d(s, a, s') \) is the key part that captures long-term dependency between current state/action \((s/a)\) with future state \(s'\). This formulation inspires us to study \( d(s, a, s') \) in depth, with function approximation as usual. Under the function approximation setting, the estimation of \( d(s, a, s') \) can benefit from enhanced generalization between similar future state \(s'\), which thus results in a better estimation for \( Q(s, a) \). Accordingly, we propose spatial monte-carlo (SMC) method, and apply the idea of SMC to actor-critic algorithms with \( m \)-step advantage estimation and GAE. Extensive experiments on a couple of public environment have demonstrated that our algorithms yields satisfactory performance empirically.

1 INTRODUCTION

Model-free reinforcement learning (RL) has risen as a promising approach for solving arbitrary goal-directed sequential decision-making problems with only high-level reward signals but no explicit supervision. Since its recent extension to utilize large scale and deep neural networks to approximate policies and value functions, it has demonstrated a great success in solving a variety of difficult problems (Mnih et al., 2015; Schulman et al., 2015a; Silver et al., 2016; Lillicrap et al., 2015). In terms of specific learning algorithms, all model-free reinforcement learning methods can be categorized into three major classes, including value-based (Watkins & Dayan, 1992), policy-based (Williams, 1992; Kakade, 2002; Schulman et al., 2015a), and actor-critic methods (Konda & Tsitsiklis, 2000; Peters & Schaal, 2008; Mnih et al., 2016). Among all these various methods, one of the essential parts is to estimate the \( Q \)-value, which represents the expected total reward. In practical value-based (e.g., DQN (Mnih et al., 2015), A3C (Mnih et al., 2016)) and actor-critic methods (e.g., DDPG (Lillicrap et al., 2015), TRPO (Schulman et al., 2015a), PPO (Schulman et al., 2017)), \( Q \)-value is often estimated by temporal different (TD) technique, which is however biased. In comparison, policy gradient methods (e.g., REINFORCE) compute an unbiased \( Q \)-value by using monte carlo (MC) methods.

Despite of diverse estimation methods, various RL algorithms formulate the value function based on the same idea of temporal decomposition, which follows the natural definition of value function and define the value function as a sum of reward over different time steps. For instance, TD methods formulate value function as a sum of expected reward in current time step and expected discount value function in next time step, and MC methods define value function as a sum of expected discount reward on all time steps. In this way, the reward is associated with time steps and shares the similar temporal decomposition of value over all RL algorithms.
Apart of existing temporal decomposition, we explore a new direction and introduce the spatial decomposition for value function in this paper. From the perspective of spatial decomposition, the value function $Q(s, a)$ can be defined as a weighted sum of reward $R(s')$ over all state $s'$ in the state space, where the weight is specified by the discounted visiting frequency $d^\pi(s, a, s')$. Essentially, $d^\pi(s, a, s')$ is determined by the interaction between the policy and the dynamic environment and yields the capability of capturing long-term dependency between the action and delayed reward/state. This formulation inspires us to take deep understanding on spatial decomposition as well as obtain effective function approximation on $d^\pi(s, a, s')$. An effective function approximation on $d^\pi(s, a, s')$ is quite crucial since it can enhance the generalization of the learned $d^\pi(s, a, s')$ when facing rarely-observed states.

However, directly learning $d^\pi(s, a, s')$ will lead to an intractable algorithm, since we need to sum over the whole state space. In order to derive an tractable algorithm, we convert the $Q$-value estimation to depend on sampled states $s'$ by introducing Gamma function $\Gamma(s, a, s')$, which equals the average discounted visiting frequency of state $s'$ over all the trajectories in which $s'$ ever appears. Accordingly, the $Q$-value can be estimated by summing over all the states that appear in sampled trajectories based on $\Gamma$ function. Following the above principles, we introduce spatial MC and apply the idea of spatial MC to actor-critic algorithms with $m$-step advantage estimation and GAE. Extensive experiments have demonstrated that such spatial decomposition is quite promising for $Q$-value estimation.

To summarize, our contributions are as follows:

1. We introduce a new perspective of value function estimation through spatial decomposition, other than existing temporal decomposition. In this way, the $Q$-value function is computed by summing over the state space instead of the time step. To our best knowledge, this work is the first attempt to investigate spatial value decomposition for reinforcement learning.

2. We identify a key role of such novel spatial view, i.e., $\Gamma(s, a, s')$, which reflects the discounted state visiting frequency based on interaction between the environment and the agent. Moreover, an effective function approximation on $\Gamma(s, a, s')$ can enable generalization of the learned $\Gamma$ function.

3. To verify our idea, we demonstrate its application to policy gradient methods. We conduct experiments on both MuJoCo and Atari domain and show that our idea is quite promising.

## 2 Q-Function: Temporal v.s. Spatial Decomposition

### 2.1 Preliminaries

One of the key problem of reinforcement learning is to learn the policy towards delayed reward. That is, the action we take now may have some delayed consequence, so we cannot take action greedily according to current reward. To represent delay reward, value function is introduced to capture the long-term dependency between reward and action, by backing up future reward into current state(or state-action pair).

Specifically, value function is defined as the expected future reward starting from current state $s$ or state-action pair $(s, a)$ and then following policy $\pi$. By definition, there are several formulation for value function.

$$V^\pi(s) = \mathbb{E}_\tau \left[ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \bigg| s_t = s, \pi \right],$$

$$Q^\pi(s, a) = \mathbb{E}_\tau \left[ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \bigg| s_t = s, a_t = a, \pi \right],$$

where $\gamma \in [0, 1]$ is a discount rate ($\gamma = 1$ is allowed only in episodic tasks). When applying value function in actor-critic based methods, estimating $Q$-function over limited number of sampled trajectories may introduce high variance. One way to reduce the variance is replacing original $Q$-function with an advantage function, which is defined as

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

$$\text{(3)}$$
Besides value function, another less used notation in RL is \( d^\pi(s) \) (Sutton et al., 2000), which represents the discounted weighting of states encountered starting at \( s_0 \) and then following \( \pi \):
\[
d^\pi(s) = \sum_{k=0}^{\infty} \gamma^k P_t\{s_k = s|s_0, \pi\}.
\]
(4)

Here \( d^\pi(s) \) characterizes the statistics of encountered states in discounted setting under policy \( \pi \).

2.2 Temporal Expansion for Q-function

Following the definition as shown in Eq. 2, Q-function can be naturally expanded in a temporal way. The roll-out in time step is common in all RL algorithms, as shown in Figure 1. For instance, Monte-Carlo (MC) methods count all time step in each episode, while temporal different (TD) methods count one step and then perform estimation on the next state. The formulation of MC and TD methods are shown in Eq. 5 and Eq. 6 respectively:
\[
Q^\pi(s, a) = \mathbb{E}_\tau \left[ r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi \right],
\]
(5)
\[
Q^\pi(s, a) = \mathbb{E}_\tau \left[ r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a, \pi \right].
\]
(6)

Note that MC is an unbiased estimation and suffers from high variance, while TD is biased with lower variance. Both MC and TD leverage the temporal structure of sampled trajectories \( \tau \) with using different discount value \( \gamma_t \) on each time step \( (t = 0, 1, 2, \ldots) \), where the discount factor \( (\gamma < 1) \) is used to ignore delayed effect of reward in the temporal structure. Since each reward is associated with time step in such temporal decomposition, the Q-function estimation is computed by a sum of discounted reward in time line.

2.3 Spatial Expansion for Q-function

In this paper, we highlight another expansion of Q-function from a spatial perspective. Inspired by \( d^\pi(s) \) in Eq. 4, we introduce a new notation \( d^\pi(s, a, s') \), which represents the discounted visiting frequency of state \( s' \) starting from \( s \), taking action \( a \) and then following policy \( \pi \).
\[
d^\pi(s, a, s') = \mathbb{E}_\tau \left[ \sum_{k=1}^{\infty} \gamma^{k-1} I\{s_{t+k} = s'|s_t = s, a_t = a, \pi\}\right],
\]
(7)

Based on this definition, we can rewrite the expectation of Q-value from spatial view:
\[
Q^\pi(s, a) = \sum_{s' \in S} d^\pi(s, a, s') R(s'),
\]
(8)

where \( R(s') \) represents a scalar reward at state \( s' \). From Eq. 8 we can see that the sum is over the whole state space, i.e., we are counting in state space instead of temporal space. In this spatial...
formulation, the reward is associated with state. As shown in Figure 2, the Q-function is a weighted sum over rewards with respect to various states, where the weight is \(d_\pi(s, a, s')\). In this new formulation, we care about which state leads to reward, instead of which time step leads to reward.

By using spatial decomposition, the value function is apart into the reward \(R(s')\) and the state discounted visiting frequency \(d_\pi(s, a, s')\). And, \(d_\pi(s, a, s')\) can capture the long-term dependency between reward and action in RL, it become a critical problem to make function approximation of \(d_\pi(s, a, s')\) to increase the generalization of learned policy.

![Figure 2: Spatial value decomposition.](image)

2.4 Function Approximation in Spatial Decomposition

When learning a function approximation of \(d_\pi(s, a, s')\), i.e., \(d_\theta\), it remains challenging to derive Q-function directly from \(d_\theta\), since the Q-function, defined by summing over the whole state space (even unseen state \(s'\) may also have a non-zero \(d_\pi(s, a, s')\) and \(R(s')\)), is not tractable.

To solve this problem, it is natural to sum over all sampled state. In order to do so, we introduce Gamma function \(\Gamma_\pi(s, a, s')\), which is defined as follows:

\[
\Gamma_\pi(s, a, s') = \mathbb{E}_{s' \in \tau} \left[ \sum_{k=1}^{\infty} \gamma^{k-1} \mathbb{1}\{s_{t+k} = s' | s_t = s, a_t = a, \pi, \tau\} \right]. \tag{9}
\]

\(\Gamma_\pi(s, a, s')\) represents the estimation of discounted visiting frequency of states that appears in sampled trajectories. By do so, we do not need to traverse in the whole state space and we can estimate the state visiting frequency using sampled trajectories. We show in Appendix that the relation between \(d_\pi(s, a, s')\) and \(\Gamma_\pi(s, a, s')\) is

\[
d_\pi(s, a, s') = \mathbb{E}_\tau \left[ \mathbb{1}\{s' \in \tau\} \right] \Gamma_\pi(s, a, s'). \tag{10}
\]

Then we can use \(\Gamma_\pi(s, a, s')\) for the estimation of Q-value, by converting Eq. (8) to the following:

\[
Q_\pi(s, a) = \mathbb{E}_\tau \left[ \sum_{s' \in \tau} \Gamma_\pi(s, a, s') R(s') \right]. \tag{11}
\]

The full derivation is shown in Appendix. As shown in Figure 3, \(\Gamma(s, a, s')\) can be estimated using sampled trajectories between state \(s\) and \(s'\).

![Figure 3: Estimation of \(\Gamma(s, a, s')\) on sampled trajectories.](image)
3 Spatial Monte-Carlo Methods with Function Approximation

Based on the definition of spatial decomposition, we introduce a new spatial monte-carlo (SMC) methods. We denote the parameter of Gamma function by $\rho$, then the spatial monte-carlo value can be estimated by sampling $n$ trajectories $\{\tau_1, \tau_2, ..., \tau_n\}$:

$$
\hat{Q}_\pi(s, a) = \frac{1}{n} \sum_{i=1}^{n} \sum_{s' \in \tau_i} \Gamma_s(s, a, s'; \rho) \hat{R}(s').
$$

(12)

Here $\Gamma_s(s, a, s'; \rho)$ is trainable function which is learned using on-policy data.

3.1 Generalization via Function Approximation

In this subsection, we want to discuss why we can benefit from the function approximation for $\Gamma(s, a, s'; \rho)$. In fact, without function approximation, SMC will achieve exactly the same estimation results with MC, as shown in Appendix B, since the difference between SMC and MC is the sum order of sampled rewards, which means SMC organizes the sampled rewards in a different view. On the other hand, with function approximation, the estimation result could be quite different. Particularly, assume that we have two similar states $s'$ and $s''$ with quite different number of observations in the sampled trajectories, e.g., $s'$ appears 100 times while $s''$ appears only 1 time. Under tabular setting, we can have a more accurate estimation $\Gamma(s, a, s')$ for state $s'$ since we have more samples for $s'$, while the estimation $\Gamma(s, a, s'')$ will be less accurate for state $s''$ since we have less samples for $s''$. But, with function approximation, we can learn the spatial structure between similar states, since the general assumption behind function approximation is that, similar state $s'$ may have similar estimation of $\Gamma(s, a, s')$. If this assumption holds well in our problem, then the function approximation enables generalization across similar $s'$. In our example, we can generalize the more accurate $\Gamma(s, a, s')$ estimation of state $s'$ to state $s''$, and resulting in a better $\Gamma(s, a, s'')$ estimation for state $s''$. Such generalization property of $\Gamma(s, a, s')$ can help reduce variance for $Q$-value estimation, thus can help estimate $Q$-function better.

However, function approximation might introduce some bias. If the assumption above doesn’t hold well, then we may have a bad estimation for the rarely observed state $s''$. When SMC is combined with $m$-step TD method, we can trade-off the variance and bias by tuning the value of $m$. We demonstrate the effectiveness of our model by empirical experiments in section 4.

3.2 Algorithms

While MC control so far has not been the state-of-the-art value-based algorithm, MC estimation is still widely used in actor-critic algorithms, especially when combined with $m$-step advantage estimation (Mnih et al. [2016]) or GAE (Schulman et al. [2015b]). Most of results in previous works (Schulman et al. [2015b, 2017]; Wu et al. [2017]) suggest that $m$-step TD methods are superior to direct MC estimation. The reason is that TD estimation suffers from much lower variance than MC estimation with the cost of introducing bias. To compare with the state-of-the-art actor-critic algorithms such as PPO (Schulman et al. [2017]) which often adopts advantage estimator, we propose two actor-critic based algorithms which combine SMC methods with $m$-step advantage estimator and GAE, respectively. Our algorithms are built based on PPO. The pseudo codes of the two proposed algorithms are shown in Appendix.

SMC-based $m$-step Advantage Estimation The formulation of $m$-step advantage estimation is:

$$
A^{(m)}_{MC}(s, a) = \mathbb{E}_\tau \left[ \sum_{k=0}^{m-1} \gamma^k r_{t+k} + \gamma^m V(s_{t+m}) - V(s) \right| s_t = s, a_t = a, \pi].
$$

(13)

In this formulation, MC estimation is used in the first $m$ steps. By substituting MC estimation with SMC estimation, we get the combined formulation:

$$
A^{(m)}_{SMC}(s, a) = \mathbb{E}_\tau \left[ \sum_{s' \in \{s_{t+1}, ..., s_{t+m}\}} \Gamma_s(s, a, s'; \rho) \hat{R}(s') + \gamma^m V(s_{t+m}) - V(s) \right| s_t = s, a_t = a, \pi].
$$

(14)
In this new formulation, \( m \) controls the scale of spatial expansion and \( V(s) \) represents the state-value function.

**SMC-based GAE** GAE is defined as an exponentially-weighted average of all \( m \)-step advantage estimators. In this subsection, we apply SMC to GAE. For notation simplicity, we denote \( A_{MC}^{(m)}(s_t, a_t) \) by \( A_{MC}^{(m)} \) and generalized advantage estimator \( G_{MC}(s_t, a_t) \) by \( G_{MC} \). Then the formulation of GAE is:

\[
\hat{G}_{MC} = (1 - \lambda)(\hat{A}_{MC}^{(1)} + \lambda \hat{A}_{MC}^{(2)} + \lambda^2 \hat{A}_{MC}^{(3)} + ...).
\] (15)

By replacing \( A_{MC}^{(m)} \) with \( A_{SMC}^{(m)} \), we get the formulation of SMC-based GAE.

\[
\hat{G}_{SMC} = (1 - \lambda)(\hat{A}_{SMC}^{(1)} + \lambda \hat{A}_{SMC}^{(2)} + \lambda^2 \hat{A}_{SMC}^{(3)} + ...)
\] (16)

4 **EXPERIMENT**

In this section, we verify our proposed algorithms on a various of environments. We test the effectiveness of SMC-based \( m \)-step advantage estimator and SMC-based GAE. For the comparison between MC-based and SMC-based \( m \)-step advantage estimation, we conduct experiments on Cart-Pole, Mujoco and Atari Pong game. For the comparison between MC-based and SMC-based GAE, we conduct experiments on Mujoco. We also conduct analysis for the value of \( m \). Due to the limit of space, we show the results of Atari Pong game in Appendix.

Our algorithms are built based on PPO [Schulman et al., 2017]. We call our first algorithm PPO-SMC, which represents PPO with our proposed SMC estimation, and the second algorithm PPO-GAE-SMC, which represents PPO with SMC-based GAE. We build policy network and value network following common practices [Schulman et al., 2015a, 2017]. For CartPole and MuJoCo tasks, Gamma function is represented by two MLP hidden layers of 64 units and relu nonlinearities. For Atari Pong task, Gamma function shares all the parameters with policy network except the last layer. Our code is implemented based on OpenAI baselines [Dhariwal et al., 2017].

4.1 **CartPole**

We conduct experiment in CartPole-v0 environment. In this environment, the dimension of observation and action are 4 and 2 respectively. The agent gets reward 1 for every step taken. The episode terminates when one of the condition meets: 1) pole angle is more than ±12 degree; 2) cart position is more than ±2.4; 3) episode length is greater than 200. Therefore, the maximum total reward is 200. We set \( m = 200 \) in this task, which means the agent leverages spatial structure in all sampled states. We use learning rate 0.01 for the Gamma-function learning. At each policy training iteration, we update the Gamma-function for 5 epochs.

The comparative results between PPO and PPO-SMC are shown in Figure 4. We find that PPO-SMC converges much faster and has much lower variance than PPO. In the contrast, PPO is unstable and suffer from high variance. This example demonstrate that the SMC estimation is much better than MC estimation.

![CartPole](image)

Figure 4: The results are averaged across 10 experiments with different random seeds.
4.2 MuJoCo

We also conduct the robotic locomotion experiments using the MuJoCo simulator (Todorov et al., 2012). We set \( m = 32 \) for both PPO and PPO-SMC. For each policy update, we train the Gamma function with a learning rate 0.1 for 5 epochs.

The results are shown in Figure 5. In the Swimmer and HalfCheetah environments, PPO-SMC outperforms PPO, which illustrates that PPO-SMC can benefit from the generalization from Gamma function approximation and reduce variance for the value estimation. In the Walker2d environment, PPO-SMC does not demonstrate advantage over PPO. One possible reason is that the Gamma function learns not so well that introduces high bias in the value estimation.

![Figure 5: Performance comparisons on three standard MuJoCo environments trained for 1 million time steps. The shaded region denotes the standard deviation over 5 random seeds.](image)

GAE is one of the state-of-the-art algorithms that used to trade off variance and bias of policy gradient. Here we compare the performance between MC-based and SMC-based GAE. Since GAE uses exponentially-weighted averaging for all \( m \)-step advantage estimators, it pays less attention to the advantage estimators with larger \( m \), which means that the benefit of spatial MC will be decayed when \( m \) becomes larger. Therefore, we use \( m = 16 \) in these experiments. We set the hyper-parameter \( \lambda \) in GAE to 0.95 which is well tuned in Schulman et al. (2015b). The results are shown in Figure 6. We find that PPO-GAE-SMC ourperforms PPO-GAE in all environments that we test. Even though PPO-SMC shows similar performance to PPO in Walker2d environment as shown in Figure 5, PPO-GAE-SMC can benefit a lot from SMC. This is because the Gamma function can be learned well and introduce less bias when the value of \( m \) becomes smaller.

![Figure 6](image)

4.3 Effects of Hyper-parameter

In this section, we conduct analysis for the effect of hypar-parameter \( m \). The value of \( m \) controls the scale of SMC: when \( m \) becomes larger, the algorithm can get more benefits from SMC to reduce variance for value estimation with the cost of introducing bias from Gamma function. The result is shown in Figure 7. In the Hopper environment, we get better performance when increasing the value of \( m \). While in the HalfCheetah and Walker2d environments, larger \( m \) will introduce much bias from Gamma function for value estimation, thus the performances drop when the value of \( m \) becomes larger. We find that the algorithm can perform consistently good when we set \( m = 16 \).
In this paper, we propose a novel spatial value decomposition for reinforcement learning, which splits the value function \( Q(s, a) \) into two parts: \( R(s') \) that represents reward and \( d(s, a, s') \) that captures discounted state visiting frequency. Such spatial decomposition further inspires us to study the \( d(s, a, s') \) with function approximation. In particular, we introduce a novel RL algorithm via learning \( d(s, a, s') \), which introduces bias but reduces variance. Extensive experiments demonstrate that we can benefit from learning \( d(s, a, s') \). Further analysis implies that, the assumption behind function approximation, i.e. similar input \((s, a, s')\) pair has similar output \(d(s, a, s')\), does hold well in our problem.

We believe spatial value decomposition is a quite interesting direction. In the future, there is still much more stuff to explore. For example, by splitting \( R(s') \) that represents reward, and \( d(s, a, s') \) that capture discounted state visiting frequency, we can study how to learn smooth \( d(s, a, s') \) even when reward is highly non-smooth. Another interesting direction is how to use the spatial value decomposition into the RL problem with extremely sparse reward. While existing RL can learn nothing from zero-reward trajectories, our new algorithm enables us to learn \( d(s, a, s') \) through zero-reward trajectories, which will further improve sample efficiency.
ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES


A DERIVATION FOR SPATIAL VALUE DECOMPOSITION

Expectation of $Q$-value in temporal view:

$$Q^\pi(s, a) = E_r[\sum_{k=1}^\infty \gamma^{k-1}r_{t+k}|s_t = s, a_t = a, \pi]$$ (17)

To get the expectation of $Q$-value from spatial view, firstly, we use $d^\pi(s, a, s')$ to represent the discounted weighting of states encountered starting at $s_t = s, a_t = a$ and then following policy $\pi$.

$$d^\pi(s, a, s') = E_r[\sum_{k=1}^\infty \gamma^{k-1}1\{s_{t+k} = s'|s_t = s, a_t = a, \pi\}]$$ (18)

Based on this definition, we can rewrite the expectation of $Q$-value in spatial view:

$$Q^\pi(s, a) = \sum_{s' \in S} d^\pi(s, a, s')R(s')$$ (19)

In this spatial formulation, we need to traverse all states, which is intractable in actual implementation. Here we introduce Gamma function $\Gamma^\pi(s, a, s')$, which represents the expectation of the discounted weighting of states $s'$ under the distribution of sampled trajectories. The expectation of Gamma function is:

$$\Gamma^\pi(s, a, s') = E_{s' \in \tau} \left[ \sum_{k=1}^\infty \gamma^{k-1}1\{s_{t+k} = s'|s_t = s, a_t = a, \pi, \tau\} \right].$$ (20)

Based on the Gamma function, we get:

$$d^\pi(s, a, s') = E_r[\sum_{k=1}^\infty \gamma^{k-1}1\{s_{t+k} = s'|s_t = s, a_t = a, \pi\}]$$

$$= E_r \left[ \sum_{k=1}^\infty 1\{s' \in \tau\}\gamma^{k-1}1\{s_{t+k} = s'|s_t = s, a_t = a, \pi, \tau\} \right]$$

$$+ E_r \left[ \sum_{k=1}^\infty 1\{s' \notin \tau\}\gamma^{k-1}1\{s_{t+k} = s'|s_t = s, a_t = a, \pi, \tau\} \right]$$

$$= E_r \left[ \sum_{k=1}^\infty 1\{s' \in \tau\}\gamma^{k-1}1\{s_{t+k} = s'|s_t = s, a_t = a, \pi, \tau\} \right]$$

$$= E_r \left[ 1\{s' \in \tau\} \right] E_{s' \in \tau} \left[ \sum_{k=1}^\infty 1\{s' \in \tau\}\gamma^{k-1}1\{s_{t+k} = s'|s_t = s, a_t = a, \pi, \tau\} \right]$$

$$= E_r \left[ 1\{s' \in \tau\} \right] E_{s' \in \tau} \left[ \sum_{k=1}^\infty \gamma^{k-1}1\{s_{t+k} = s'|s_t = s, a_t = a, \pi, \tau\} \right]$$

$$= E_r \left[ 1\{s' \in \tau\} \right] \Gamma^\pi(s, a, s')$$

By substituting $d^\pi(s, a, s')$ using Eq. (21) we get:

$$Q^\pi(s, a) = \sum_{s' \in S} d^\pi(s, a, s')R(s')$$

$$= \sum_{s' \in S} E_r \left[ 1\{s' \in \tau\} \right] \Gamma^\pi(s, a, s')R(s')$$

$$= E_r \left[ \sum_{s' \in S} 1\{s' \in \tau\} \Gamma^\pi(s, a, s')R(s') \right]$$

$$= E_r \left[ \sum_{s' \in \tau} \Gamma^\pi(s, a, s')R(s') \right]$$ (22)

Afterward, we can estimate the $Q$-value using $\Gamma(s, a, s')$ based on the sampled trajectories.
B EQUIVALENCE BETWEEN TEMPORAL AND SPATIAL FORMULATION

Temporal Q-value expectation:

\[ Q^\pi(s, a) = \mathbb{E}_\tau \left[ \sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \right] = \mathbb{E}_\tau \left[ \sum_{k=1}^{\infty} \gamma^{k-1} R(s_{t+k}) \right] \tag{23} \]

Spatial Q-value expectation:

\[
Q^\pi(s, a) = \sum_{s' \in S} d^\pi(s, a, s') R(s') \\
= \mathbb{E}_\tau \left[ \sum_{s' \in \tau} \Gamma^\pi(s, a, s') \hat{R}(s') \right] \\
= \mathbb{E}_\tau \left[ \sum_{s' \in \tau} \mathbb{E}_{s' \in \tau} \left[ \sum_{k=1}^{\infty} \gamma^{k-1} \mathbb{1}\{s_{t+k} = s' \mid s_t = s, a_t = a, \pi, \tau\} R(s') \right] \right] \tag{24} \\
= \mathbb{E}_\tau \left[ \sum_{s' \in \tau} \mathbb{E}_{s' \in \tau} \left[ \sum_{k=1}^{\infty} \gamma^{k-1} R(s_{t+k}) \mathbb{1}\{s_{t+k} = s' \mid s_t = s, a_t = a, \pi, \tau\} \right] \right] \\
= \mathbb{E}_\tau \left[ \sum_{k=1}^{\infty} \gamma^{k-1} R(s_{t+k}) \right]
\]

C ADDITIONAL EXPERIMENTS

C.1 ATARI GAME

We also conduct experiments on Atari domain. Due to the limit of time and resource, we only choose the commonly used Pong environment. We show the training result in Figure 8. Empirically, we find that our spatial formulation can help to reduce variance of training process and make the training policy more robust.

![Figure 8](image-url)

Figure 8: The curves are averaged on 3 different random seeds.

C.2 EXTENSION TO OTHER RL ALGORITHMS

We extend our proposed method to other RL algorithms (e.g. TRPO) and find that TRPO can also benefit a lot from this spatial decomposition. We believe that our proposed method can be leveraged by other RL algorithms.
D ALGORITHMS

Algorithm 1 SMC-based $m$-step advantage estimation for Policy Optimization

1: Initialize policy parameter $\theta_0$, value-function parameter $\phi_0$, and Gamma-function parameter $\rho_0$
2: for $k = 0, 1, 2, \ldots$ do
3:   Simulate current policy $\pi_{\theta_k}$ until $M$ timesteps are obtained.
4:   Compute Gamma function loss: $L_\rho = \sum_{t=1}^{T-1} \sum_{s' \in \{s_{t+1}, \ldots, s_{t+m}\}} (\Gamma_{\rho_k}(s_t, a_t, s') - \hat{\Gamma}(s_t, a_t, s'))^2$
5:   Update distance function parameter using Adam optimizer: $\rho_{k+1} = \rho_k + \alpha \nabla L_\rho$
6:   Compute $\hat{A}\text{SMC}(s_t, a_t)$ at all timesteps.
7:   Compute $\theta_{k+1}$ and $\phi_{k+1}$
8: end for

Algorithm 2 SMC-based GAE for Policy Optimization

1: Initialize policy parameter $\theta_0$, value-function parameter $\phi_0$, and Gamma-function parameter $\rho_0$
2: for $k = 0, 1, 2, \ldots$ do
3:   Simulate current policy $\pi_{\theta_k}$ until $M$ timesteps are obtained.
4:   Compute Gamma function loss: $L_\rho = \sum_{t=1}^{T-1} \sum_{s' \in \{s_{t+1}, \ldots, s_{t+m}\}} (\Gamma_{\rho_k}(s_t, a_t, s') - \hat{\Gamma}(s_t, a_t, s'))^2$
5:   Update distance function parameter usingAdam optimizer: $\rho_{k+1} = \rho_k + \alpha \nabla L_\rho$
6:   Compute $\hat{G}\text{SMC}(s_t, a_t)$ at all timesteps.
7:   Compute $\theta_{k+1}$ and $\phi_{k+1}$
8: end for