A randomized block policy gradient algorithm with differential privacy in Content Centric Networks

Lin Wang¹, Xingang Xu¹, Xuhui Zhao¹, Baozhu Li², Ruijuan Zheng¹ and Qingtao Wu¹

Abstract
Policy gradient methods are effective means to solve the problems of mobile multimedia data transmission in Content Centric Networks. Current policy gradient algorithms impose high computational cost in processing high-dimensional data. Meanwhile, the issue of privacy disclosure has not been taken into account. However, privacy protection is important in data training. Therefore, we propose a randomized block policy gradient algorithm with differential privacy. In order to reduce computational complexity when processing high-dimensional data, we randomly select a block coordinate to update the gradients at each round. To solve the privacy protection problem, we add a differential privacy protection mechanism to the algorithm, and we prove that it preserves the ε-privacy level. We conduct extensive simulations in four environments, which are CartPole, Walker, HalfCheetah, and Hopper. Compared with the methods such as important-sampling momentum-based policy gradient, Hessian-Aided momentum-based policy gradient, REINFORCE, the experimental results of our algorithm show a faster convergence rate than others in the same environment.

Keywords
Content Centric Networks, differential privacy protection, randomized block coordinate

Date received: 12 July 2021; accepted: 7 October 2021
Handling Editor: Yanjiao Chen

Introduction
Content Centric Network (CCN) shows great potential in the future Internet development; many multimedia applications in CCN are policy decision problems in essence, for example, traffic prediction and resource allocation. Reinforcement learning¹–³ is an effective means to deal with optimization decision problems; it is applied in many network fields, such as congestion control,⁴ traffic scheduling,⁵ network security,⁶ and load balancing.⁷ The principle of reinforcement learning is an iteration decision process in which an agent constantly interacts with the environment to strengthen its decision-making ability, which mainly solves the sequence decision problem. At each iteration, an agent changes the state of the environment by performing an action, and the environment feeds it back a reward based on a reward function. The objective of reinforcement learning is to obtain a policy that maximizes cumulative reward. Therefore, how to design a better policy to achieve this goal is very important.

As an important branch of reinforcement learning, policy gradient methods are widely used in many fields,
such as video game, AlphaGo, news recommendation, and network resource. Policy gradient methods have a better performance in continuous action space (or higher dimensional space), and they can implement a randomized strategy. Policy gradient methods directly parameterize the policy and learn based on the strategy, which defines a policy \( \pi \) as a function with parameter \( \varphi \); the policy \( \pi \) is a probability distribution, indicating the probability distribution of choosing different actions in a certain state. After the policy is represented as a continuous function, we can use the continuous function optimization method to update the parameters; policy gradient optimization algorithms are becoming the major research areas. The most classic optimization algorithms are REINFORCE\(^{13}\), Policy Gradient Theorem (PGT)\(^{14}\), and Gradient of a Partially Observable Markov Decision Process (GPOMDP).\(^{15}\) These methods use the policy \( \pi_\varphi(a|s) \) with the parameter \( \varphi \) to sample, and then they estimate the gradient and update the parameter \( \varphi \). However, these algorithms have a poor convergence rate because of the large variance in gradient estimation and sampling.

In order to reduce the variance in gradient estimation, Stochastic Variance Reduced Gradient Descent (SVRG)\(^{16}\) uses a batch sample to estimate and calculate the gradient of all the samples every once in a while; although SVRG is very successful in supervised learning, it is difficult to apply it to policy gradient. To address the problem, Stochastic Variance Reduced Policy Gradient (SVRPG)\(^{17}\) is a random variance reduction algorithm of the policy gradient used to solve the Markov Decision Process (MDP). SVRPG uses the importance sampling weight to retain the unbiased gradient estimation, which can ensure convergence under the standard assumption of MDP. But the above algorithms have a high sample complexity and large training batches; then important-sampling momentum-based policy gradient (IS-MBPG) and Hessian-Aided Momentum-based Policy Gradient (HA-MBPG)\(^{18}\) combine momentum methods\(^{19–21}\) with the important sampling and Hessian-aided methods, which achieves a faster convergence rate with an adaptive learning rate, and IS-MBPG reduces the sample complexity of \( \varepsilon \) approximate point of stability to \( O(1/\varepsilon^3) \).

However, the algorithms mentioned above calculate the full gradient for all dimensions of data in each iteration, which causes a large amount of calculation. And these algorithms use centralized processing methods in the data training, which upload data to the central node, reducing communication resources. In addition, the data training may involve sensitive data, which causes the privacy breach.

In order to solve the above problems, we propose a randomized block policy gradient algorithm with differential privacy (DP-RBPG) in CCN, which combines the random block-coordinate methods and differential privacy to meet the demand of high-dimensional data processing and privacy protection. At each iteration, we randomly select one block coordinate to update the gradients, which decreases the variance of gradient estimator and accelerates the convergence rate. And if we use all block coordinates in gradient updating, our algorithm will reduce to IS-MBPG. DP-RBPG can also reduce the sample complexity of \( \varepsilon \) approximate point of stability to \( O(1/\varepsilon^3) \); the whole training process does not need large batches, and the comparison on sample complexity with other algorithms is shown in Table 1.

![Table 1: The comparison on sample complexity to attain \( \varepsilon \) approximate point of stability.](image-url)

| Algorithm        | Sample complexity   | Privacy protection |
|------------------|---------------------|--------------------|
| REINFORCE\(^{13}\) | \( O(1/\varepsilon^2) \) | \( \times \)        |
| PGT\(^{14}\)     | \( O(1/\varepsilon^2) \) | \( \times \)        |
| GPOMDP\(^{15}\)  | \( O(1/\varepsilon^2) \) | \( \times \)        |
| SVRPG\(^{4}\)    | \( O(1/\varepsilon^{5/2}) \) | \( \times \)        |
| HAPG\(^{22}\)    | \( O(1/\varepsilon) \) | \( \times \)        |
| IS-MBPG\(^{18}\) | \( O(1/\varepsilon^3) \) | \( \times \)        |
| DP-RBPG (this study) | \( O(1/\varepsilon^3) \) | \( \checkmark \) |

DP-RBPG: randomized block policy gradient algorithm with differential privacy; GPOMDP: gradient of a partially observable markov decision process; HA-MBPG: Hessian-aided momentum-based policy gradient; HAPG: Hessian aided policy gradient; IS-MBPG: important-sampling momentum-based policy gradient; PGT: Policy Gradient Theorem; SVRPG: Stochastic Variance Reduced Policy Gradient.

The main contributions are listed as follows.

- We propose a DP-RBPG based on momentum and important sampling methods. DP-RBPG accelerates gradient descent and decreases the variance of gradient estimator without sacrificing the performance, and we introduce a differential privacy protection mechanism during data processing.
- We add the noise interference into updating gradient, which follows the Laplace distribution. Based on the property of Laplace distribution, we prove that it can preserve the \( \varepsilon \)-privacy level.
- We implement DP-RBPG in MuJoCo environment, which contains CartPole, Walker, Hopper, and HalfCheetah. The experimental results show that DP-RBPG improves the convergence rate, and as the data dimension goes up, the algorithm has a better performance. Meanwhile, as the
level of ε-privacy increases, our algorithms will decrease slightly in experiments.

The article is organized as follows. We review the related works in the “Related work” section. The preliminaries of policy gradient and reinforcement learning are described in the “Preliminaries” section. Our algorithm is proposed in the “Differential privacy randomized block policy gradient” section. Environment deployment and results of the contrast experiments are described in the “Experiments” section. Finally, we give a conclusion about the article in the “Conclusion” section.

Related work

Policy gradient

Recently, decreasing the variance of gradient estimator is the main way to study policy gradient. Le Roux et al.23 proposed stochastic average gradient (SAG) algorithm, which used two gradients: one is the gradient of the previous iteration and the other is the new gradient—both of these gradients selected a sample randomly to calculate. The convergence rate of SAG was faster than the stochastic gradient descent (SGD) of the gradient estimated from a single sample. Defazio et al.24 proposed stochastic average gradient descent (SAGA), an accelerated version of the SAG algorithm, which used an unbiased estimator to update gradient and reduced the impact of noise. However, the problems of SAG and SAGA were that they required memory to maintain every old gradient. Xu et al. proposed SVRG, which used a batch sample to estimate the gradient and estimated gradient with the batch samples over a period of time; after then, it reselected a batch sample to estimate the gradient. SVRG attained a faster convergence rate than SAG, but it could not guarantee the unbiased estimation of gradient. Allen-Zhu25 proposed a new algorithm named for nonconvex stochastic optimization problems, which divided the internal cycle of SVRG into n sub-epochs and used the structural information of strong nonconvex functions effectively, so that it is more efficient than SVRG in stochastic optimization. Fang et al.26 put forward stochastic path-integrated differential estimator (SPIIDER) algorithm for finding first- and second-order stable points of nonconvex stochastic optimization, and they proved that the SPIIDER algorithm was optimal in solving nonconvex random optimization algorithms. Nguyen et al.27 proposed a new random recursive gradient algorithm StochAstic Recursive grAdient algoritHm (SARAH), which used a recursive framework to update the gradient. Different from SAG and SVRG algorithms, it did not need the pass gradients to update the gradient estimator, which saved memory space, so that the convergence rate of SARAH is faster than others. However, all of these algorithms only did well in oblivious supervised learning, not reinforcement learning.

More recently, Papini et al.17 came up with a new reinforcement learning algorithm named SVRPG, which was applied to policy gradient. This method decreased the sample complexity and converged faster. Xu et al. proposed a better convergence analysis method than SVRPG; the sample complexity of ε approximate point of stability was reduced to $O(1/\epsilon^{5/3})$. Shen et al.22 proposed HAPG, which combined a Hessian aided with policy gradient, and HAPG reduced the sample complexity of ε approximate point of stability to $O(1/\epsilon^3)$; meanwhile, the method could apply in existing reinforcement learning techniques, and the performance of Hessian aided policy gradient (HAPG) was better than SVRPG. Xu et al.28 proposed the stochastic recursive variance reduced policy gradient (SVR-PG), which reduced the sample complexity of ε approximate point of stability to $O(1/\epsilon^{3/2})$. Huang et al. put forward IS-MBPG and HA-MBPG, which were based on important sampling and momentum methods. IS-MBPG and HA-MBPG improved the sample complexity of ε approximate point of stability to $O(1/\epsilon^3)$ with small sample batches, and they achieved adaptive learning rate by adjusting step size in these algorithms. But the above algorithms are poor convergence in high-dimensional data.

Randomized block coordinate

Randomized block methods selected one coordinate to update the gradient in each iteration, which could reduce iteration costs and memory requirements and speed up convergence rate.29 In order to handle large training tasks, Diakonikolas and Orecchia30 proposed an algorithm named accelerated alternating randomized block coordinate descent (AAR-BCD), which optimized the method of random block selection. Zhao et al.31 proposed a mini-batch randomized block coordinate descent (MRBCD) algorithm, which updated the gradient with mini-batch samples in each round, and the variance of gradient estimator in MRBCD was reduced and the convergence rate was accelerated. Lacoste-Julien et al.32 proposed a randomized block algorithm to solve the problem of block-separable constraints, which was mainly used in support vector machines (SVMs). Singh et al.33 improved the Nesterov method with gradient projection methods, which accelerated the convergence rate. Lin et al.34 put forward a new method to analyzing the problem of asynchronous distributed optimization. Based on the above observation, we select randomized block-coordinate method to update the gradients, which can decrease the variance of gradient estimator and accelerate the convergence rate.
**Differential privacy**

Differential privacy methods resist differential attacks by adding a random mechanism to achieve the purpose of privacy protection. Gao and Ma proposed an algorithm which combines reinforcement with differential privacy in processing dynamic data. Ding et al. put forward a alternating direction method based on differential privacy and reinforcement. Cheng et al. proposed a novel stochastic gradient descent algorithm with deep learning and differential privacy. Dai et al. utilized reinforcement learning to solve the problem of network security. Whereas in processing high-dimensional data, these algorithms are lacking.

**Preliminaries**

In this section, we will introduce some preliminary knowledge about reinforcement learning and policy gradient methods.

**Reinforcement learning**

The most basic model of reinforcement learning is MDP, which consists of a set of environmental states $S$, a set of actions $A$, a set of rewards $R$, a discount factor $\gamma \in [0, 1]$, and the transition probability between states $P$, which takes the form

$$P_{st} = P[S_{t+1} = s'|S_t = s, A_t = a]$$ (1)

The MDP can be described as follows: an agent whose initial state is $s_0$ selects an action $a_0$ from $A$ to execute, the agent randomly transfers to the next state $s_1$ with a probability $P$. We define $\pi$ as the set of action policy; $\pi(a|s)$ is the probability of taking the possible action $a$ for state $s$ in the process, which is expressed as

$$\pi(a|s) = P(A_t = a|S_t = s)$$ (2)

The goal of the MDP is to find an optimal policy $\pi$, which is a mapping function from the state to action, to get the maximum reward $R^\pi = \sum_{s \in A} \pi(a|s)R^a$ from the environment.

**Policy gradient**

Different from the value-based methods, policy gradient methods are based on a policy to learn, which outputs the action or the probability of the action directly based on the state. Comparing with the parameterized representation in the value function, the policy parameterization is simpler and has better convergence. Analyzing from the perspective of importance sampling, the objective of policy gradient is maximizing cumulative return. $\sigma = \{s_0, a_0, \ldots, s_N, a_N\}$ represents a set of state-behavior trajectory sequences, where the probability of emergence of trajectory $\sigma$ is $p(\sigma|\phi)$, the trajectory reward is $R(\sigma) = \sum_{i=0}^{N} \gamma^i R(s_i, a_i)$, and the expected cumulative reward of a parametric policy is defined as

$$f(\phi) = \mathbb{E}\left(\sum_{i=0}^{N} R(s_i, a_i); \pi_\phi\right) = \sum_{\sigma} p(\sigma|\phi)R(\sigma)$$ (3)

During the training, the objective of policy gradient is to find optimal parameter $\phi$, which can be described as

$$\max_{\phi} f(\phi) = \max_{\phi} \sum_{\sigma} p(\sigma|\phi)R(\sigma)$$ (4)

Based on formula (4), we turn the policy search approach into an optimization problem. The methods solving this problem include the policy gradient, Newton’s method and Quasi-Newton Methods, and Interior Point Method. The policy gradient is the simplest and most commonly used, which uses $\phi' = \phi + \alpha \nabla f(\phi)$ to update the parameter $\phi$; we first take the derivative of the target function $f$

$$\nabla f(\phi) = \nabla_{\phi} \sum_{\sigma} p(\sigma|\phi)R(\sigma) = \sum_{\sigma} \nabla_{\phi} p(\sigma|\phi)R(\sigma)$$

When calculating the policy gradient, the data used are sampled under the new strategy, which requires that all the samples should be resampled according to the new strategy after each gradient update. The data utilization of this approach is very low, which makes the lower speed of convergence. Therefore, we bring in the concept of importance sampling, using the old parameter $\phi$ to compute the expected return of the parameter $\phi'$

$$f(\phi) = \sum_{\sigma} p(\sigma|\phi)p(\sigma|\phi')R(\sigma)$$

Taking the derivative of this formula, we can get the same result as equation (5). But the trajectory probability $p(\sigma|\phi)$ is not known, and the result of equation (6) is difficult to get. Combining stochastic gradient descent method, we select a batch of trajectories $M = \{\sigma_i\}_{i=1}^M$ from the trajectory distribution; meanwhile, we define $\nabla f(\phi)$ as an estimated value of the $\nabla f(\phi)$, which takes the form

$$\nabla f(\phi) = \frac{1}{|M|} \sum_{i \in M} \nabla \log p(\sigma_i|\phi)R(\sigma_i)$$ (7)

Based on equation (7), we add the learning rate $\xi_t$ to update the parameter $\phi$
where the learning rate is greater than zero and \( \varphi \in \mathbb{R}^d \), \( \mathbb{R}^d \) means that the \( d \) dimensions positive number set. At the beginning, the algorithm adopts a large learning rate; when the error curve enters the plateau stage, the learning rate is reduced to make more precise adjustment. We adopt an unbiased estimator \( v(\sigma, \varphi) \) based on the trajectory \( \sigma_t \), which meets the needs of \( \mathbb{E}[v(\sigma_t, \varphi)] = \nabla f(\varphi) \); based on equations (5) and (7), we can get

\[
\log p(\sigma|\varphi) = \log p(s_1) + \sum_{n=0}^{N} \log \pi_{\sigma}(a_{n}|s_{n}) + \sum_{n=0}^{N} \log \pi_{\sigma}(a_{n}|s_{n}, a_{n})
\]

\[
= \sum_{n=0}^{N} \gamma^n R(d_n, s_n)
\]

\[
\nabla f(\varphi) = \frac{1}{|M|} \sum_{\sigma \in M} v(\sigma, \varphi)
\]

so that \( \nabla \log p(\sigma|\varphi) = \sum_{n=0}^{N} \log \pi_{\sigma}(a_{n}|s_{n}) \), and the \( \nabla f(\varphi) \) can be rewritten as the form

\[
\nabla f(\varphi) = \frac{1}{|M|} \sum_{\sigma \in M} \left( \sum_{n=0}^{N} \nabla \log \pi_{\sigma}(a_{n}|s_{n}) \right) + \sum_{n=0}^{N} \gamma^n R(d_n, s_n)
\]

However, there are some problems with equation (9); the larger cumulative return value will make the larger parameter update, so that the model will fluctuate greatly, which may affect the final model effect. Although the policy is an unbiased estimator of expectation, the variance is very large due to over-dependence on each sampling trajectory, so we bring in the baseline \( c \) to reduce the variance as

\[
\nabla f(\varphi) = \frac{1}{|M|} \sum_{\sigma \in M} \left( \sum_{n=0}^{N} \nabla \log \pi_{\sigma}(a_{n}|s_{n}) \right) - c
\]

We can prove that \( \mathbb{E}[\nabla \log p(\sigma|\varphi)c] = 0 \) as

\[
\mathbb{E}[\nabla \log p(\sigma|\varphi)c] = \sum_{\sigma} p(\sigma|\varphi) \nabla \log p(\sigma|\varphi)c
\]

\[
= \sum_{\sigma} p(\sigma|\varphi) \nabla \log p(\sigma|\varphi)c/
\]

\[
= \sum_{\sigma} \nabla \log p(\sigma|\varphi)c
\]

\[
= \nabla \log p(\sigma|\varphi)c
\]

\[
= \nabla \log p(\sigma|\varphi)c = 0
\]

So that the baseline \( c \) is to reduce the variance without changing the expectation.

The current strategy gradient algorithm has high computational cost when processing high-dimensional data, and privacy leakage has not been considered, but privacy protection is very important in data training. Therefore, we propose the differential privacy randomized block policy gradient algorithm. In order to reduce the computational complexity of processing high-dimensional data, a random block coordinate is selected randomly to update the gradient of each round. In terms of privacy protection, a differential privacy protection mechanism is added to the algorithm, and it is proved that it maintains \( \varepsilon \)-privacy level.

### Differential privacy randomized block policy gradient

In this part, we combine the randomized coordinate method and differential privacy with important sampling momentum-based policy gradient method (DP-RBPG). The randomized coordinate way does well in dealing with optimization problems, especially in higher dimensions. The implementation of the method is shown in Algorithm 1. Due to the trajectory probability, \( v(\sigma|\varphi) \) in equation (5) unable to calculate directly; we bring in the randomized coordinate method and important sampling to reduce the variance. At each round, we randomly select a coordinate block to update the gradients. Meanwhile, we bring differential privacy in our algorithm, which adds a noisy that obeys Laplace distribution, and we prove that it preserves the \( \varepsilon \)-privacy level. Now we introduce the definitions related to differential privacy as follows.

#### Algorithm 1: DP-RBPG

**Input:** Total iteration number \( T \), constant parameters \( \{h, n, a\} \) and initial input \( \varphi_1 \).
1. **for** \( t = 1, 2, 3, \ldots, T \) **do**
2. **Randomly select the kth coordinate with probability** \( p_k \)
3. **if** \( t = 1 \) **then**
4. **Sample a trajectory** \( \sigma_1 \) **from** \( p(\sigma|\varphi_1) \), **and compute**
5. **else**
6. **Sample a trajectory** \( \sigma_t \) **from** \( p(\sigma|\varphi_t) \), **and compute**
7. **end if**
8. **Compute** \( H_t = \|\nu(\sigma_{\Lambda_1})\|\)
9. **Compute** \( \xi_1 = \frac{1}{(a + \sum_{j=1}^{H_1})^{\alpha}}\)
10. **Update** \( \varphi_{t+1} = \varphi_t + \xi_1 w_t \)
11. **Update** \( a_{t+1} = a_{t} e^{\frac{\alpha}{\xi_1}}\)
12. **end for**
13. **Output:** \( \varphi \) chosen uniformly random from \( \{\varphi_t\}_{t=1}^T \)

DP-RBPG: randomized block policy gradient algorithm with differential privacy.
Definition 1. We first define the concept of adjacent relation, we introduce the adjacent data sets $M = \{x_1, \ldots, x_n\}$ and $M' = \{x'_1, \ldots, x'_n\}$. These two data sets have one and only one data set that is different, which means that there is only one $i$ such that $x_i \neq x'_i$, and other $i' \in \{1, \ldots, n\}$ such that $x_{i'} = x'_{i'}$.

Definition 2. Then we bring in the definition of differential privacy. For a randomized algorithm $B$, which means that the output $\Sigma \subseteq \text{Range}(B)$ of the algorithm is not fixed but obeys a certain distribution, we define the differential privacy as

$$P[B(M)] \leq e^{-\varepsilon} \cdot P[B(M')]$$

where $\varepsilon$ is a smaller positive number, $M, M'$ means different data sets, and $\text{Range}(B)$ is the output range of randomized algorithm $B$. The objective is to prove that the randomized algorithm $B$ preserves the $\varepsilon$-privacy level, and the smaller the $\varepsilon$, the higher the degree of privacy protection.

Definition 3. We introduce the definition of the randomized algorithm $B$ global sensitivity as

$$\Delta(i) = \sup_{M_i, M'_i : \Delta(M_i, M'_i)} \| B(M_i) - B(M'_i) \|$$

where $\| B(M_i) - B(M'_i) \|$ denotes the Manhattan distance between $B(M_i)$ and $B(M'_i)$ at time $t$, and the global sensitivity reflects the maximum range of variation of a randomized algorithm in a pair of adjacent data sets.

Formally, for $t = 1$, the policy gradient takes the form

$$w_1 = v_k(\sigma_1|\phi_1) + \eta_k(t)$$

where $\eta_k(t)$ obeys the Laplace distribution $\text{Lap}(\delta(t))$, it denotes that the $k$th dimension number in the noisy, $\eta_k(t)$, has the same data dimension as the gradient $w$. More generally, we define the momentum-based gradient as

$$w_{t,k} = \alpha_i v_k(\sigma_i|\phi_i) + (1 - \alpha_i) [w_{t-1,k} + v_k(\sigma_i|\phi_i) - \alpha_i u(\sigma_i|\phi_{i-1}, \phi_i) v_k(\sigma_i|\phi_{i-1})] + \eta_k(t)$$

where we select $k$th coordinate of decision to update the policy gradient, and the coordinate variable $k$ is independent of $t, \phi, \sigma$. It generates the gradient $w_{t,k} = \{w_{t,1}, w_{t,2}, \ldots, w_{t,n}\}$. And we also use $k$th number of noisy as the distributions. $u(\sigma_i|\phi', \phi_i)$ is the important sampling weight; based on the fundamental apply of important sampling in policy gradient in equation (6), we use a known probability distribution to get it and we add the important sampling weight as

$$u(\sigma_i|\phi', \phi_i) = \frac{u(\sigma_i|\phi_i)}{p(\sigma_i|\phi_i)} = \prod_{n=0}^{N} \frac{\pi_f(\sigma_i|\phi_i)}{\pi_f(\sigma_i|\phi_i)}$$

By equation (16), we can achieve that $\mathbb{E}_{\sigma(\phi)}[\nabla \sigma(\phi)] - u(\sigma(\phi), \phi) = \nabla f(\phi) - f(\phi')$, which decreases the variance on the basis of constant expectation in the process of gradient calculation. As showed in Algorithm 1, we define the norm of the gradient $\nabla V(\phi)$ as $G_t$, and then the learning rate is $\xi_t$, which implements adaptive adjustment.

According to the proof method in IS-MBPG, we can also get the similar conclusion that the $\varepsilon$ approximate point of stability with our algorithm is $\mathcal{O}(1/\varepsilon^3)$.

Then we will prove that DP-RBPG preserves the $\varepsilon$-differential privacy level. We define the $b_{t,k}$ as

$$b_{t,k} = w_{t,k} - \eta_k(t)$$

For the $k$th coordinate, $w_{t,k}$ means that the coordinate $i$ of $w_{t,k}$ and the gradients $w_{t,k}$ and $b_{t,k}$ have $n_k$ dimensions, which means that $m_1 + \cdots + m_m = n_k$.

Based on Definition 3, we can get the conclusion that

$$\| b_{t,k} - b_{t,k} \| \leq \Delta(t)$$

Next, we will analyze the privacy property of our algorithm.

Theorem 1. Based on Algorithm 1, $w_{t,k}$ are independent and identically distributed variables drawn from the Laplace distribution with parameter $\delta(t)$. If the parameter $\delta(t)$ makes $\delta(t) = \Delta(t)/\varepsilon$ for all $t \in \{1, \ldots, T\}$, we can prove that our algorithm guarantees the $\varepsilon$-privacy level.

Proof: \begin{align} n \prod_{i=1}^{n} \frac{P[w_{t,k} - b_{t,k}]}{P[w_{t,k} - b_{t,k}]} & \leq \prod_{i=1}^{n} \exp \left( \frac{-\| w_{t,k} - b_{t,k} \|}{\delta(t)} \right) \leq \prod_{i=1}^{n} \exp \left( \frac{\| b_{t,k} - b_{t,k} \|}{\delta(t)} \right) = \exp \left( \frac{\Delta(t)}{\delta(t)} \right) \end{align}

The first inequality above uses the triangle inequality, and the second inequality uses equation (19) to prove; when the parameter $\delta(t)$ makes $\Delta(t)/\varepsilon = \varepsilon$, we can prove that it preserves the $\varepsilon$-privacy level. Therefore, we finish the proof of Theorem 1.
Experiments

In this part, considering about the multimedia data transmission mechanism in CCN, we train our algorithm offline in a lot of epochs. Meanwhile, we mainly introduce the experimental results with our algorithm in four simulation environments, which are CartPole, Walker2D, HalfCheetah, and Hopper. We compare our algorithm with IS-MBPG, REINFORCE, and HA-MBPG in these simulation environments. The detailed environmental setup and the analysis of experimental results are shown below.

Experimental setup

The experiments are implemented on VMware Workstation 15 Pro 15.5.0 with a Linux environment with Ubuntu-18.04.3 (Linux kernel 4.8.4); the whole process is in a computer with Intel® Xeon® Sliver 4114 CPU and 64 GB RAM. To ensure the fairness of the experiments, we implement these algorithms in the same environment, which is based on PyTorch 1.6.0, MuJoCo 1.50.1.68, Garage 2019.10.0, TensorFlow 1.15.3, and Gym 0.12.4. In the process, we set the same initial value for all algorithms, aiming at the problem of data randomness that may exist in the experiment; we repeated the experiment many times and selected the average result as the final result.

In particular, different from REINFORCE, we rewrite CartPole using categorical policies to calculate, which is always used in discrete action space, and other environments use Gaussian policies, which are always used in continuous action space. The parameters of the experiment are set as follows: the Neural Network expects that CartPole is $8 \times 8$, others are $64 \times 64$. In an experiment, the training horizon of CartPole is 100, Walker2D and HalfCheetah are 500, and Hopper is 1000. To be fair, the learning rate of these algorithms is set to 0.01. The number of timesteps of CartPole is set to $5 \times 10^5$, and others are $1 \times 10^7$. The batch size of CartPole and Hopper is 50, and the ones in Walker2D and HalfCheetah are 100. Some other hyperparameters such as $h, n, a$ of IS-MBPG in our algorithm are also same to the original paper, and specific numerical values can be obtained from the appendix of the original paper. For ease of reading, the specific parameter settings are shown in Table 2.

Specially, similar to HA-MBPG and IS-MBPG, we use the system probes to represent the complexity of samples, which is a better measurement standard for training. The system probes avoid the problem of returning failure caused by different sample lengths in the training process. In the experiments, we achieve a faster convergence rate than the other three algorithms by comparing running time for the same system probes; meanwhile, the average episode return is close to the latest algorithm IS-MBPG. In addition, during data training, considering about preserving data privacy, we add differential privacy protections when calculating average episode returns, which increases security during data training. We add a stochastic Laplace distribution factor $\eta(t)$ in gradient updating, which is between 0 and 1. The stochastic factor $\eta(t)$ is used to simulate the interference items during the network transmission, which can increase the security of data transfer process, and we demonstrate the influence of different $\epsilon$-privacy level on our algorithm in experiments.

Experimental results

As shown in Figure 1, we deploy four algorithms, which are DP-RBPG, IS-MBPG, HA-MBPG, and REINFORCE, in the same environment. In CartPole environment, we can see that our algorithm is close to IS-MBPG, which is obviously better than HA-MBPG and REINFORCE; to be more precise, the training results of the average training episode reward of IS-MBPG and DP-RBPG are close to 90, and HA-MBPG and REINFORCE are close to 85 and 78. As shown in Figure 2, because there is little difference in low-dimensional data of training, we record the training time of 500 epochs; the time of these algorithms is similar, and the fastest one is REINFORCE, followed by DP-RBPG. Because the algorithm has a certain degree
of randomness, we choose the best training result of each algorithm as the final result, and the running time of DP-RBPG is 99% and 96% of IS-MBPG and HA-MBPG, respectively, in CartPole; though the convergence rate of DP-RBPG is 1% slower than REINFORCE, the average episode return is much better than it.

In Walker environment, as shown in Figure 1, we can see that the average episode return of our algorithm is close to IS-MBPG, which is obviously better than HA-MBPG and REINFORCE. In particular, the training results of IS-MBPG and DP-RBPG are close to 350, and HA-MBPG and REINFORCE are close to 290 and 230. Meanwhile, different from training low-dimensional data in CartPole, as the data dimension goes up, we can see that DP-RBPG reaches a state of convergence more quickly than IS-MBPG and HA-MBPG. In Figure 2, we record the training time of 200 epochs; although REINFORCE is also the fastest one, the performance is poor. The running time of DP-RBPG is 87% and 82% of IS-MBPG and HA-MBPG, respectively, in Walker environment.

In Hopper environment, as shown in Figure 1, all of these algorithms are convergent to 1000. However, we can plainly find that our algorithm DP-RBPG converges faster than IS-MBPG, HA-MBPG, and REINFORCE from the beginning to the maximum. Due to the fluctuation of data training greatly, the training result is the average episode return, the batch size is 50,000, and we divide it into five rounds. The results show that our algorithm can effectively converge. In Figure 2, we can see that the running time of DP-RBPG is lower than IS-MBPG, HA-MBPG, and REINFORCE in 200 training epochs. The running time of DP-RBPG is 89%, 85%, and 95% of IS-MBPG, HA-MBPG, and REINFORCE, respectively, in Hopper environment.
In HalfCheetah environment, as shown in Figure 1, IS-MBPG is the best one in average episode return, which is close to 240. The average episode return of our algorithm DP-RBPG is close to 200, and the growth trend is roughly the same with IS-MBPG. The training results of HA-MBPG and REINFORCE are close to 120 and 250. In Figure 2, we can find that HA-MBPG costs the most time in 200 epochs; although REINFORCE is also the fastest one, the performance is poor in Figure 1. The running time of DP-RBPG is 91% and 85% of IS-MBPG and HA-MBPG, respectively, in HalfCheetah environment.

In summary, from the above experimental results, the running time of our algorithm is lower than IS-MBPG and HA-MBPG in the same environment. Although the running time of REINFORCE is lower than our algorithm, the performance is poor. The reason is that REINFORCE is an update of the turn-based system, which has a big variance in gradient estimation. We can get a conclusion that our algorithm iterates more times than others in the same time period, which accelerates gradient descent. Moreover, we can see that the average episode return is close to IS-MBPG in all environments. And we can see that the experimental results in CartPole are inconspicuous; as the dimensions go up, we can get the difference between the running time in these algorithms.

In addition, we bring in the differential privacy protection mechanism, which adds a stochastic Laplace distribution factor $\eta_t(t)$ that is between 0 and 1. In order to show the performance of DP-RBPG in different environments, we test it in CartPole and Walker2D.
environment. As shown in Figure 3(a), we test our algorithm DP-RBPG with different $\varepsilon$-privacy level ($\varepsilon = 0, 0.2, 0.5, 0.8$) in CartPole environment. Due to the randomness in data processing, we used the average result after 20 training sessions; we can see that as the $\varepsilon$-privacy level increases, the average episode return decreases gradually. The average episode return of DP-RBPG outperforms the DP-RBPG ($\varepsilon = 0.2, 0.5, 0.8$) by 3.5%, 4.9%, and 5.7%, respectively. The reason for the average episode return degradation is that we add noise to every gradient descent, which ensures the privacy of the data training. Similarly, as shown in Figure 3(b), we test our algorithm DP-RBPG with different $\varepsilon$-privacy level ($\varepsilon = 0, 0.2, 0.5, 0.8$) in Walker environment. We used the average result after 10 training sessions; we can see that DP-RBPG works best when $\varepsilon$-privacy level is 0. The average episode return of DP-RBPG leads to 2.1%, 3.3%, and 5.5% improvements over the DP-RBPG ($\varepsilon = 0.2, 0.5, 0.8$).

**Conclusion**

In this article, we proposed a DP-RBPG in CCN, which can solve the problems of mobile multimedia data transmission in CCN. In the process of data packet transmission in CCN, traffic optimization and congestion control problems can be formulated into policy decision problem based on the current network delay, packet loss rate, and maximum throughput. DP-RBPG can generate a policy to adjust network state through constantly interacting with the environment. DP-RBPG selects one randomized block coordinate in gradient updating at every round; we compared it with other algorithms. The experiments prove that the training time of DP-RBPG is lower than others in the same training epochs. Moreover, we bring in a differential privacy protection mechanism, which adds a Laplace distribution factor in gradient updating, and we prove that it preserves the $\varepsilon$-privacy level. Therefore, DP-RBPG has solved the problems of large computations caused by calculating all dimensional data and the privacy breach in data training. This method can solve the problem of mobile multimedia data transmission in the CCN, and this method also can protect user privacy, improve network performance and user experience. Meanwhile, DP-RBPG can be applied in some practical problems of the conventional network, such as traffic optimization and congestion control. These applications can be regarded as policy decision problems. We will study these applications in our future work.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the National Natural Science Foundation of China (NSFC; Grant No. 61871430), the Scientific and Technological Innovation Team of Colleges and Universities in Henan Province (Grant No. 20IRTSTHN018), the basic research projects in the University of Henan Province (Grant No. 19zx010), and the Key Scientific and Technological Projects Henan Province (Grant No. 202102210169), Key Scientific Research Projects of Colleges and Universities in Henan Province (Grant No.22A520005).

**ORCID iDs**

Lin Wang https://orcid.org/0000-0003-1704-5144  
Xingang Xu https://orcid.org/0000-0003-1844-7639  
Xuhui Zhao https://orcid.org/0000-0003-4361-8604  
Baozhu Li https://orcid.org/0000-0002-4631-0317  
Ruijuan Zheng https://orcid.org/0000-0002-0932-8788  
Qingtao Wu https://orcid.org/0000-0003-1572-5293
References

1. Alt B, Sošić A and Koeppl H. Correlation priors for reinforcement learning, https://arxiv.org/abs/1909.05106 (2019, accessed 12 October 2020).
2. Song F, Ai Z, Zhou Y, et al. Smart collaborative automation for receive buffer control in multipath industrial networks. *IEEE T Ind Inform* 2020; 16(2): 1385–1394.
3. Zhang W, Zhang Z, Zeadally S, et al. MASM: a multiple-algorithm service model for energy-delay optimization in edge artificial intelligence. *IEEE T Ind Inform* 2019; 15(7): 4216–4224.
4. Xu Z, Tang J, Yin C, et al. Experience-driven congestion control: when multi-path TCP meets deep reinforcement learning. *IEEE J Sel Area Comm* 2019; 37(6): 1325–1336.
5. Comsa IS, Zhang S, Aydin ME, et al. Towards 5G: a reinforcement learning-based scheduling solution for data traffic management. *IEEE T Net Serv Man* 2018; 15(4): 1661–1675.
6. Ni Z and Paul S. A multistage game in smart grid security: a reinforcement learning solution. *IEEE T Netw Lett* 2019; 30(9): 2684–2695.
7. Li Z, Wang C and Jiang CJ. User association for load balancing in vehicular networks: an online reinforcement learning approach. *IEEE T Intell Transp* 2017; 18(8): 2217–2228.
8. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015; 518(7540): 529–533.
9. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge. *Nature* 2017; 550(7676): 354–359.
10. Zheng G, Zhang F, Zheng Z, et al. DRN: a deep reinforcement learning framework for news recommendation. In: *Proceedings of the 2018 World Wide Web conference on World Wide Web (WWW’18)*, Lyon, 23–27 April 2018, pp.167–176. New York: ACM.
11. Quan W, Liu Y, Zhang H, et al. Enhancing crowd collaborations for software defined vehicular networks. *IEEE Commun Mag* 2017; 55(8): 80–86.
12. Zhang M, Hao B, Song F, et al. Smart collaborative video caching for energy efficiency in cognitive Content Centric Networks. *J Netw Comput Appl* 2020; 158: 102587.
13. Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach Learn* 1992; 8(3–4): 229–256.
14. Sutton RS, McAllester DA, Singh SP, et al. Policy gradient methods for reinforcement learning with function approximation. In: Sola SA, Leen TK and Müller KR (eds) *Advances in neural information processing systems 12*. Cambridge, MA: The MIT Press, 2000, pp.1057–1063.
15. Baxter J and Bartlett PL. Infinite-horizon policy-gradient estimation. *J Artif Intell Res* 2001; 15: 319–350.
16. Xu T, Liu Q and Peng J. Stochastic variance reduction for policy gradient estimation, https://arxiv.org/pdf/1710.06034.pdf (2018, accessed 12 September 2020).
17. Papini M, Binaghi D, Canonaco G, et al. Stochastic variance-reduced policy gradient, https://ui.adsabs.harvard.edu/abs/2018arXiv180605618P (2018, accessed 6 January 2020).
18. Huang F, Gao S, Pei J, et al. Momentum-based policy gradient methods. In: *Proceedings of the 37th international conference on machine learning (virtual event)*, Vienna, 13–18 July 2020, vol. 119, pp.4422–4433. New York: ACM.
19. Gitman I, Lang H, Zhang P, et al. Understanding the role of momentum in stochastic gradient methods. In: *Proceedings of the 33rd conference on neural information processing systems*, Vancouver, BC, Canada, 8–14 December 2019, pp.9630–9640. Cambridge: The MIT Press.
20. Luo ZJ, Chen SY, Qian YT, et al. Stochastic momentum method with double acceleration for regularized empirical risk minimization. *IEEE Access* 2019; 7: 166551–166563.
21. Zhang M, Zhou Y, Quan W, et al. Online learning for IoT optimization: a Frank-Wolfe Adam-based algorithm. *IEEE Internet Things* 2020; 7(9): 8228–8237.
22. Shen Z, Ribeiro A, Hassani H, et al. Hessian aided policy gradient. In: *Proceedings of the 36th international conference on machine learning*, Long Beach, CA, 9–15 June 2019, pp.5729–5738. New York: ACM.
23. Le Roux N, Schmidt M and Bach FR. A stochastic gradient method with an exponential convergence rate for finite training sets. In: *Proceedings of the 25th international conference on neural information processing systems*, Lake Tahoe, NV, 3–6 December 2012, pp.2672–2680. New York: ACM.
24. Defazio A, Bach F and Lacoste-Julien S. SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives. In: *Proceedings of the 27th international conference on neural information processing systems*, Montreal, QC, Canada, 8–13 December 2014, pp.1646–1654. New York: ACM.
25. Allen-Zhu Z. Natasha: faster non-convex stochastic optimization via strongly non-convex parameter. In: *Proceedings of the 34th international conference on machine learning*, Sydney, NSW, Australia, 6–11 August 2017, pp.89–97. New York: ACM.
26. Fang C, Li CJ, Lin Z, et al. SPIDER: near-optimal non-convex optimization via stochastic path integrated differential estimator. In: *Proceedings of the 32nd international conference on neural information processing systems*, Montreal, QC, Canada, 3–8 December 2018, pp.687–697. New York: ACM.
27. Nguyen LM, Liu J, Scheinberg K, et al. SARAH: a novel method for machine learning problems using stochastic recursive gradient. In: *Proceedings of the 34th international conference on machine learning*, Sydney, NSW, Australia, 6–11 August 2017, pp.2613–2621. New York: ACM.
28. Xu P, Gao F and Gu Q. Sample efficient policy gradient methods with recursive variance reduction. In: *Proceedings of the 8th international conference on learning representations*, Addis Ababa, 26–30 April 2020. OpenReview.net.
29. Xie T, Liu B, Xu Y, et al. A block coordinate ascent algorithm for mean-variance optimization. In: *Proceedings of the 32nd international conference on neural information processing systems*, Montreal, QC, Canada, 3–8 December 2018, pp.1073–1083. New York: ACM.
30. Diakonikolas J and Orecchia L. Alternating randomized block coordinate descent. In: *Proceedings of the 35th international conference on machine learning*, Stockholm, 10–15 July 2018, pp.1232–1240. New York: ACM.
31. Zhao T, Yu M, Wang Y, et al. Accelerated mini-batch randomized block coordinate descent method. In: Proceedings of the 28th international conference on neural information processing systems, Montreal, QC, Canada, 8–13 December 2014. Cambridge: The MIT Press.

32. Lacoste-Julien S, Jaggi M, Schmidt M, et al. Block-coordinate Frank-Wolfe optimization for structural SVMs. In: Proceedings of the 30th international conference on machine learning, Atlanta, GA, 16–21 June 2013, vol. 28, pp.53–61. New York: ACM.

33. Singh C, Nedic A and Srikant R. Random block-coordinate gradient projection algorithms. In: Proceedings of the 53rd IEEE conference on decision and control, Los Angeles, CA, 15–17 December 2014, pp.185–190. New York: IEEE.

34. Lin Y, Shames I and Nešić D. Asynchronous distributed optimization via dual decomposition and block coordinate ascent. In: Proceedings of the 2019 IEEE 58th conference on decision and control, Nice, 11–13 December 2019, pp.6380–6385. New York: IEEE.

35. Song F, Zhou YT, Wang Y, et al. Smart collaborative distribution for privacy enhancement in moving target defense. Inform Sciences 2019; 479: 593–606.

36. Zhu J, Xu C, Guan J, et al. Differentially private distributed online algorithms over time-varying directed networks. IEEE T Signal Inf Process Over Netw 2018; 4(1): 4–17.

37. Gao R and Ma X. Dynamic data publishing with differential privacy via reinforcement learning. In: Proceedings of the 2019 IEEE 43rd annual computer software and applications conference, Milwaukee, WI, 15–19 July 2019, vol. 1, pp.746–752. New York: IEEE.

38. Ding J, Errapotu SM, Zhang H, et al. Stochastic ADMM based distributed machine learning with differential privacy. In: Proceedings of the security and privacy in communication networks—15th EAI international conference (SecureComm), Orlando, FL, 23–25 October 2019, vol. 304, pp.257–277. Cham: Springer.

39. Cheng H, Yu P, Hu H, et al. Towards decentralized deep learning with differential privacy. In: Proceedings of the 12th international conference on cloud computing (CLOUD 2019; held as part of the Services Conference Federation), San Diego, CA, 25–30 June 2019, vol. 11513, pp.130–145. Cham: Springer.

40. Dai C, Xiao Z, Wan X, et al. Reinforcement learning with safe exploration for network security. In: Proceedings of the 2019 IEEE international conference on acoustics, speech and signal processing (ICASSP), Brighton, 12–17 May 2019, pp.3057–3061. New York: IEEE.

41. Paszke A, Gross S, Massa F, et al. PyTorch: an imperative style, high-performance deep learning library. In: Proceedings of the 33rd conference on neural information processing systems, Vancouver, BC, Canada, 8–14 December 2019, pp.8026–8037. Cambridge: The MIT Press.

42. Todorov E, Erez T and Tassa Y. MuJoCo: a physics engine for model-based control. In: Proceedings of the 2012 IEEE/RSJ international conference on intelligent robots and systems, Vilamoura-Algarve, 7–12 October 2012, pp.5026–5033. New York: IEEE.

43. Duan Y, Chen X, Houthooft R, et al. Benchmarking deep reinforcement learning for continuous control. In: Proceedings of the 33rd international conference on machine learning, New York, 19–24 June 2016, pp.1329–1338. New York: ACM.

44. Abadi M, Agarwal A, Barham P, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems, https://arxiv.org/abs/1603.04467 (2015, accessed 5 February 2020).

45. OpenAI Gym, https://gym.openai.com/ (2018, accessed 10 February 2020).