Proceedings of CONF-MLA 2025 Symposium: Applied Artificial Intelligence Research
DOI: 10.54254/2755-2721/2025.BJ27261

Monte Carlo Algorithm and Its Applications in Reinforcement Learning and Probabilistic Graph Model

Qing Chen

School of Artificial Intelligence, Nanjing University, Nanjing, China
221300074@smail.nju.edu.cn

Abstract. With rapid advances in AI and machine learning, reinforcement learning and probabilistic inference have become increasingly important, with the Monte Carlo algorithm, a numerical method based on random sampling, widely applied in these fields. This paper examines the use of the Monte Carlo algorithm in reinforcement learning and probabilistic inference, assessing its strengths and limitations in addressing complex problems. Through a review of relevant literature and analysis of policy optimization in reinforcement learning and sample generation in probabilistic inference, it discusses the application of Monte Carlo algorithms, and highlights the integration of Monte Carlo and MCMC methods, addressing future challenges and solutions. The results show that Monte Carlo algorithms effectively estimate value functions and optimize policies in reinforcement learning, particularly in unknown environments. In probabilistic inference, MCMC algorithms efficiently generate samples to handle complex probabilistic models, playing a crucial role in applications like Bayesian inference. However, both Monte Carlo and MCMC methods face challenges in computational efficiency and convergence, requiring further optimization and improvement.

Keywords: Monte Carlo algorithm, reinforcement learning, probabilistic graphical model, Markov chain

1. Introduction

The Monte Carlo algorithm has become a vital tool in science and engineering due to its strengths in tackling complex probabilistic problems and processing high-dimensional data. With advances in high-performance computing, it is now widely applied across fields such as computational physics, statistics, and finance. However, with large-scale and multi-scale problems, the algorithm faces high costs, slow convergence, and limited accuracy. In artificial intelligence and statistics, reinforcement learning and probabilistic inference involve extensive sampling and strategy optimization. Monte Carlo methods enhance model performance in these fields, especially when combined with Markov chains for sampling and optimizing complex distributions, though this integration also introduces new challenges. This paper examines Monte Carlo Algorithm (MCA) in reinforcement learning and probabilistic inference, highlighting its strengths and challenges in complex problem solving. By reviewing literature and analyzing policy optimization and sample generation, it discusses MCA applications, focusing on its integration with Markov Chain Monte Carlo and exploring challenges

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.BJ27261

and future directions. This paper presents strategies to improve MCA efficiency and address key challenges. In addition, Monte Carlo methods have potential for further development in artificial intelligence and quantum computing.

2. Overview of monte carlo algorithm

For many complex tasks, deterministic methods may lack practicality or scalability. In contrast, stochastic algorithms provide practical solutions by introducing randomness into the computation, allowing for controlled approximation errors. Based on large-scale random sampling, the MCA provides efficient approximations of expectations or complex solutions. It applies to numerical integration, optimization, physics simulations, reinforcement learning, and probabilistic inference. This effectiveness relies on a structured process of problem modeling, stochastic process design, sample generation, statistical inference, and convergence analysis. For effective approximation, the process begins with formal problem modeling to define the solution space and objectives, then designs a tailored stochastic process to enable representative sampling from the target distribution With sufficient samples from repeated trials, statistical inference is used to estimate target quantities. To ensure accuracy and stability, convergence analysis evaluates these estimates, and if necessary, sampling strategies or sizes are adjusted to enhance computational efficiency and reliability.

Mathematically, the MCA can be formalized as the case where a computable problem $f : \sum* \to \sum*$ exists, the error margin satisfies $0 \leq \varepsilon < 1$, and the computational complexity is defined by a function $T: \mathbb{N} \to \mathbb{N}$. If there exists a randomized algorithm A such that for any input $x \in \sum*$, the following condition holds [2].

$$\Pr(A(x) \neq f(x)) \leq \varepsilon \qquad \text{, Pr(A runs on input x in at most takes T(|x|) steps)=1(1)}$$

If the probability is taken over the random choices made by A within time T(n), then A is called to be considered as MCA for computing f with an error probability of at most ε within time T(n).

Despite its flexibility and broad applicability in solving complex problems, MCA still faces notable limitations in practical applications. To achieve higher accuracy, a large number of samples is often required, hence resulting in slow convergence and rapidly increasing computational costs in high-dimensional problems. In scenarios such as non-episodic tasks for reinforcement learning or high-dimensional Bayesian inference, excessive variance results in instability during convergence. To address these challenges, variance reduction techniques such as importance sampling, as well as hybrid approaches, are commonly employed to improve convergence and computational efficiency across various application scenarios [3].

3. Monte carlo-based policy evaluation and improvement in reinforcement learning

3.1. Reinforcement learning and Markov Decision Process

In reinforcement learning, agent-environment interaction is modeled via a Markov Decision Process (MDP), with the objective of maximizing long-term cumulative rewards in uncertain or partially known settings [4]. Specifically, the MDP is described in terms of quintuples (X, A, R, P, γ). The state transitions satisfy the Markov property, meaning that the next state depends only on the current state and action, without dependence on the past history. Here, X denotes the state space, A the action space, $R: X \times A \to \mathbb{R}$ the reward function, $P: X \times A \times X \to [0,1]$ the state transfer probability, and $\gamma \in [0,1]$ the discount factor, which balances immediate rewards and long-term returns.

DOI: 10.54254/2755-2721/2025.BJ27261

Therefore, the core problem to be solved by reinforcement learning is to solve the optimal strategy $\pi*$ that maximizes the expected cumulative reward, which can be formally expressed as

$$\pi^* = \arg\max_{\pi} E_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{2}$$

In contrast to dynamic programming methods that rely on known models, reinforcement learning often operates under the constraint of unknown or imprecise state transition probabilities. As such, Monte Carlo method has become an important tool for policy evaluation in reinforcement learning, which does not rely on explicit models, but rather approximate the estimation of the value function $V^\pi(s)$ or state-action value function $Q^\pi(s,a)$ by sampling trajectories through interaction with the environment. However, such sampling-based policy estimation faces challenges like high variance and slow convergence, prompting the development of on-policy and off-policy MCA to improve sample efficiency and optimize learning.

3.2. On-policy Monte Carlo reinforcement learning

On-policy Monte Carlo methods iteratively improve the policy to maximize cumulative rewards by repeatedly sampling complete episodes under the current policy and making unbiased estimation of the value function based on actual observed returns. The core idea is that, given a fixed behavior policy π , multiple full episodes are generated from the start state to a terminal state. Whenever a state-action pair (s, a) is accessed, its value function is updated using the actual return G_t observed in that episode. The value function of a pair (s,a) can be incrementally updated as follows:

$$Q(s,a) \leftarrow Q(s,a) + \frac{1}{N(s,a)} \sum_{t \in T(s,a)} (G_t - Q(s,a)) \tag{3}$$

where $N(s,a)$ is the number of times of the selected action (a) in state (s), $T(s,a)$ is the set of all time steps of the selected action (a) of in state (s), and G_t denotes the cumulative reward of the discounts from the beginning of the current time step t to the end of the episode:

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} R\left(s_k, a_k\right) \tag{4}$$

Each time an episode is completed, all visited state-action pairs are updated once according to the formula above, and the policy π is adjusted based on these updated values $Q(s,a)$. To balance exploiting the best-known actions and exploring potentially better ones, the ϵ-greedy method is commonly employed. This method selects the action with the highest value most of the time, but with a small probability ϵ, it chooses a random action to encourage exploration, thereby obtaining more diverse sample trajectories and avoiding local optima [5]. Through repeated episode sampling, the value function averages are computed continuously to iteratively improve the current policy. By relying solely on sampled trajectories, on-policy Monte Carlo methods avoid the need for a known environment model when estimating value functions. They work well in discrete tasks like board games and turn-based path planning, especially when state transitions are unknown or difficult to model. However, they require complete episodes and suffer from high variance, slow convergence, and high computational cost. As such, they underperform in non-episodic tasks and combine with temporal-difference methods for faster, more stable learning.

DOI: 10.54254/2755-2721/2025.BJ27261

3.3. Off-policy Monte Carlo reinforcement learning

Through importance sampling, off-policy Monte Carlo reinforcement learning can estimate value functions and update policies using data collected under a different behavior policy. This allows the agent to use data from a behavior policy to estimate and update a different target policy [6]. In offline and limited scenarios, where direct interaction with the environment under the target policy is impossible, learning depends on pre-collected data. Thus, the agent collects state-action-reward sequences by executing the behavior policy. Due to the potential mismatch between the behavior and target policy distributions, importance weights are assigned to trajectory returns. These weights are computed as follows.

$$w_t = \prod_{k=t}^{T-1} \frac{\pi'(a_k|s_k)}{\pi(a_k|s_k)} \tag{5}$$

where w_t represents the trajectory weight starting from time step t , π is the target strategy, and π' is the behavioral strategy. These weighted samples are used to obtain an unbiased estimate of the value function under the target policy, enabling policy improvement that brings the learning process closer to the optimal solution expected under π.

Compared to on-policy methods, off-policy Monte Carlo reinforcement learning makes full use of data that do not originate from the target policy, improving sample efficiency and increasing data diversity. This makes it particularly suitable for scenarios where real-time interaction is infeasible, such as medical records, user behavior logs, or offline recommendation systems. However, the use of importance sampling can introduce high variance, especially when the behavior policy diverges significantly from the target policy. To mitigate this, practical implementations often incorporate techniques like weight truncation, variance regularization, or adaptive resampling to balance bias and variance, thus improving stability and computational tractability. In real-world settings, offline reinforcement learning frameworks may further enhance data utility via mixed behavior policies, multi-stage replay, or model-based auxiliary estimates. These mechanisms help off-policy methods learn effectively under sampling constraints and enable its integration with temporal-difference learning and deep networks for better generalization.

4. Monte carlo approaches in probabilistic graphical inference

4.1. Probabilistic graphical models and Markov chains

In probabilistic inference, the main challenge is to estimate hidden variables or unknown parameters with partial observations. This becomes especially difficult when dealing with high-dimensional or complex distributions, where direct computation is often infeasible. Probabilistic graphical models combine probability theory with graph theory to efficiently represent dependencies among random variables using directed or undirected graph structures. And they are key tools for modeling joint distributions [7]. In a probabilistic graphical model, nodes represent one or more random variables, and edges encode conditional independence relationships among them. This structure allows the joint probability distribution of a complex, high-dimensional system to be factorized into a product of local conditional distributions, significantly simplifying inference.

Once the graph is constructed, inference targets marginal and conditional distributions. However, exact inference in large-scale graphical models is often computationally intractable, as it requires summing or integrating over the entire state space, with complexity growing exponentially with the number of variables. As such, approximate inference is commonly used in practice, except for a few

DOI: 10.54254/2755-2721/2025.BJ27261

special structures like tree-structured graphs. In the context of probabilistic inference, approximate methods are generally categorized into deterministic techniques such as variational inference and sampling-based methods, with Markov Chain Monte Carlo (MCMC) as a key example of the latter. In a Markov chain, the next state depends only on the current state, not on past states.

$$P(X_{t+1} \mid X_t) = P(X_{t+1} \mid X_1, X_2, \ldots, X_t) \tag{6}$$

To enable efficient sampling from complex joint distributions through MCMC, a Markov chain defines a stochastic process through its state space, transition matrix, and transition probabilities [8].

4.2. Markov chain monte carlo methods and their applications

In high-dimensional probabilistic inference where direct sampling is infeasible, MCMC offers an effective alternative by generating samples through a constructed Markov chain. Upon convergence to the target stationary distribution, its state sequence enables approximation of posterior statistics including means, variances, and marginals [9]. In this method, candidate states are proposed based on the current state, allowing the Markov chain to evolve toward the target distribution. Whether a candidate is accepted depends on the Metropolis-Hastings acceptance probability, enabling the chain to approximate the target distribution over time.

$$A\left(x^{*} \;\; x^{t-1}\right) = \min \;\; \left(1, \; \frac{p(x^{*})\, Q\left(x^{t-1} \;\; x^{*}\right)}{p(x^{t-1})\, Q\left(x^{*} \;\; x^{t-1}\right)}\right) \tag{7}$$

where $\mathrm{p}(\bullet)$ is the target distribution, and $\mathrm{q}(\bullet|\bullet)$ is the proposal distribution. If accepted, the chain transitions to that state. Otherwise, it remains in the current state. After sufficient iterations, once the Markov chain converges to its stationary distribution, samples drawn from the chain can be used to approximate expectations, marginal probabilities, and other statistics under the target distribution.

Different transition mechanisms and acceptance strategies give rise to various MCMC variants. For example, the Metropolis-Hastings algorithm is a general method using flexible proposals and acceptance probabilities, suitable for complex distributions. When conditional distributions can be directly sampled, Gibbs Sampling, as a special case of Metropolis-Hastings, efficiently samples by alternately updating variables. And it is especially suited for high-dimensional models with clear conditional independence structures, like latent Dirichlet allocation for latent variable analysis. In Bayesian inference, MCMC generates samples from the posterior distribution to estimate parameter means, variances, and credible intervals. It can also be combined with methods like importance sampling and bridge sampling to approximate model evidence, supporting model comparison and selection. Additionally, the method is used to infer latent structures. For example, Gibbs Sampling effectively captures topic distributions in documents within latent Dirichlet allocation.

5. Future development and challenges

5.1. Mathematical basis and present limitations

Both Monte Carlo reinforcement learning and probabilistic inference adopt Markov processes and stochastic sampling to estimate parameters or future states. In reinforcement learning, Markov decision processes guide policy improvement via state transitions to maximize long-term returns. In probabilistic inference, Markov chains sample to approximate posterior distributions. Both methods

DOI: 10.54254/2755-2721/2025.BJ27261

rely on continuous sampling and updating through interaction with the environment or data. For example, Q-learning iteratively updates value functions within the state-action space, similar to how Bayesian inference continually updates posterior distributions with new data, reflecting their commonality in handling uncertainty and information updates [10]. However, these approaches face multiple practical limitations. Monte Carlo reinforcement learning supports incremental updates and does not require an environment model, making it applicable to discrete or continuous state spaces, but its computational complexity grows linearly with sample size. In high-dimensional settings, it faces the curse of dimensionality, which significantly increases the computational burden. The high variance inherent in stochastic sampling may lead to unstable estimates and slow convergence. For non-episodic tasks such as continuous or infinite-horizon tasks, Monte Carlo methods are generally less efficient than stepwise update strategies such as temporal difference learning. While MCMC efficiently samples complex, high-dimensional distributions and is highly adaptable, it struggles with convergence assessment, diagnostic costs, and computational demands on large datasets [11].

5.2. Prospects and advanced optimization strategies

Improvements in computation and inference are needed to overcome the limitations of Monte Carlo methods in sampling efficiency and scalability. At the sampling stage, importance sampling and stratified sampling refine sampling distributions and divide the sample space to increase sample effectiveness and minimize variance, reducing the required sample size. In high-dimensional spaces, variational inference improves convergence speed by approaching sampling as an optimization task.

By integrating temporal difference learning with sampling, dynamic programming can be used to more efficiently approximate optimal policies [12]. At the computational level, parallel computing and GPU acceleration technologies enhance the execution efficiency of large-scale sampling. To enhance data utilization, techniques such as sparse representations, low-rank approximations, and deep learning can be applied to enable more efficient policy learning and function estimation with fewer samples. Meanwhile, online learning and dynamic adjustment of sampling strategies enable real-time optimization of sampling distributions according to current estimates, thereby accelerating convergence. Despite ongoing challenges such as computational complexity and convergence speed, Monte Carlo methods still hold significant potential in areas like model interpretability, big data analysis, and quantum computing. Simulating complex distributions helps deepen understanding of model outputs and supports latent feature analysis in machine learning applications like generative adversarial networks. By leveraging the parallelism of quantum computing, Quantum Monte Carlo expands problem-solving capabilities in quantum physics and chemistry [13]. With continuous advancements in algorithms and hardware, Monte Carlo methods will continue to play a greater role in high-dimensional parameter estimation and large-scale data processing.

6. Conclusion

This paper investigates the central role and applicability of Monte Carlo methods in reinforcement learning and probabilistic inference, focusing on their practical value in unknown environments and high-dimensional complex distributions. In reinforcement learning, Monte Carlo methods estimate value functions by sampling trajectories, avoiding reliance on explicit state transition models, while also balancing sampling bias and sample efficiency in both on-policy and off-policy designs. In probabilistic inference, MCMC builds Markov chains that converge to target distributions, enabling accurate approximation of posteriors and supporting high-dimensional tasks like Bayesian analysis and latent variable modeling. Due to slow convergence and high variance, Monte Carlo methods in

DOI: 10.54254/2755-2721/2025.BJ27261

high dimensions often rely on adaptive sampling or integration with TD learning and deep networks to improve efficiency. Future research should combine Monte Carlo with deep models to enhance inference under uncertainty.

References

[1] Xiang, J.Y., Wang, Z.H. and Deng, Y.Y. (2024). A Review of Machine Learning Classification Based on Random Forest Algorithm. Artificial Intelligence and Robotics Research, 13(1), 143-152.
[2] Guidotti, N.L., Acebrón, J.A. and Monteiro, J.A (2024). Fast Monte Carlo Algorithm for Evaluating Matrix Functions with Application in Complex Networks. J Sci Comput 99, 41.
[3] Kuznetsov, I. (2024). Guided Exploration in Reinforcement Learning via Monte Carlo Critic Optimization. Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, 2348-2350.
[4] Lin, C. (2022). Theoretical and applied research on reinforcement learning methods. Computer Science and Applications, 12(3), 554-564.
[5] Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.
[6] Geist, M., & Scherrer, B. (2014). Off-policy learning with eligibility traces: A survey. Journal of Machine Learning Research, 15, 289-333.
[7] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of state calculations by fast computing machines. Journal of Chemical Physics, 21(6), 1087-1092.
[8] Mausam, & Kolobov, A. (2012). Planning with Markov Decision Processes: An AI Perspective. Morgan & Claypool.
[9] Hastings, W.K. (1970). Monte Carlo sampling methods using Markov chains and their applications. Biometrika, 57(1), 97-109.
[10] Watkins, C.J.C.H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3), 279-292.
[11] Gelman, A., Carlin, J.B., Stern, H.S., et al. (2013). Bayesian Data Analysis (3rd ed.). CRC Press.
[12] Tsitsiklis, J.N., & Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control, 42(5).
[13] Chiribella, G., D'Ariano, G.M., & Perinotti, P. (2009). Quantum Monte Carlo methods: Algorithms and applications. Reviews in Mathematical Physics, 21(8), 1105-1155.