Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

USING COSIMLA WITHIN POLICY ITERATION FOR MDPS WITH LARGE STATE SPACES

Yifu Tang[1], Peter W. Glynn[2], and Zeyu Zheng[1]

[1]Dept. of Industrial Engineering and Operations Research, University of California, Berkeley, CA, USA
[2]Dept. of Management Science and Engineering, Stanford University, Stanford, CA, USA

ABSTRACT

Classical policy iteration methods to solve Markov Decision Processes (MDPs) incur a computational complexity that critically depends on the size of state space. Such computational complexity can be prohibitive when the size of state space is enormous or even countably infinite. To improve the computational effectiveness, we develop a computational method that strategically integrates policy iteration and a recently developed approach called COSIMLA (Combining Numerical Linear Algebra and Simulation). We provide analysis for the proposed computational method and demonstrate its comparative advantages through numerical experiments.

1 INTRODUCTION

Policy iteration has been a classical and markedly important approach for solving Markov decision process (MDP) when the 4-tuple element (S,A,P,r) of the MDP is given. Usually, S represents the state space, A represents the action space, P represents the transition probability, and r represents the rewards. In the MDP framework, an agent aims to find an action according to every state, denoted as a policy, to maximize the infinite sum of discounted rewards. The policy iteration method, a popular approach to solving MDP, is briefly described as follows. The agent starts with an initial policy. At each step, the agent evaluates the current policy and updates the policy corresponding to each state according to the evaluation. Formulations are provided in Section 2 and we refer to PART I of (Agarwal et al. 2019) for a detailed review of MDP.

At each step of the policy iteration algorithm, a policy needs to be evaluated by computing the value function or the Q-function (defined in Section 2). To evaluate a policy, a corresponding linear system of the Q-function or the value function needs to be solved. Bertsekas (2011) reviews various approaches in policy evaluation, including matrix-inversion-based algorithms (e.g., least-squares temporal difference), and iterative algorithms (e.g., least-squares policy iteration and temporal difference TD(λ)). These simulation-based approaches aim at solving the linear systems of the Q-functions or the value functions in policy iteration processes.

The computational complexity of solving general MDP with infinitely discounted reward via policy iteration has been shown in Corollary 4.1 of Ye (2011) to be a strongly polynomial time algorithm with complexity $O\left(m\cdot\frac{|S|(|S||A|-1)}{1-\gamma}\log\left(\frac{|S|^2}{1-\gamma}\right)\right)$, where $\gamma<1$ is the discount rate, m is the complexity for each iteration step, and $\frac{|S|(|S||A|-1)}{1-\gamma}\log\left(\frac{|S|^2}{1-\gamma}\right)$ is the number of iteration steps. In each iteration step, the agent needs to solve a linear system of order $|S|\cdot|A|$, $m=O(|S|^3)$; see section 1.3 of (Agarwal et al. 2019) for reference. The computational need at each step (solving a linear system) can be prohibitive when the state space $|S|$ is enormously large, let alone for some scenarios where there is technically an infinite number of states (i.e., $|S|=\infty$).

In this work, we provide an approach to strategically combine linear algebra and Monte Carlo simulation that can lead to a significant reduction of computational need. We adopt the recently developed COSIMLA (COmbined SIMulation and numerical Linear Algebra) approach by (Zheng et al. 2022) and (Glynn and Zheng 2023) and integrate that with evaluating the value function or Q-function. We write the Q-function into a regenerative form of cumulative rewards. The idea of COSIMLA is to decompose S into disjoint subsets B and $B^c = S - B$. Then, we truncate the regenerative summation of rewards by a stopping time denoting the first exit of the subset B, denoted by T. The cumulative rewards truncated up to the stopping time T are expressed with a matrix-inversion formula while those outside the subset B can be simulated. Since there is no need to solve linear equations in large state space, the computational complexity is reduced. Besides, when the state space is countably infinite, the classical policy iteration is not fit for use. However, our approach still works to give an evaluation of the Q-function.

In the rest of this paper, we introduce the notations and the formal problem settings in Section 2. In Section 3, we propose the COSIMLA-Assisted policy iteration algorithm for large but finite state spaces and analyze its complexity. In Section 4, we propose the local policy iteration algorithm with theoretical results concerning infinite state spaces. Numerical experiments are conducted in Section 5 and conclusions are presented in Section 6.

2 PROBLEM SETTING FOR MDP AND POLICY ITERATION

Let S be a finite or countably infinite state space, $(X_n : n \geq 0)$ in S be the underlying Markov chain of the Markov decision process, and $A = \{a_1, \cdots, a_K\}$ be a finite action set. For each state $s \in S$, an action $a \in A$ will induce a transition from s to another $s' \in S$ with probability $P(s'|s,a)$, where

$$P(s'|s,a) = \mathbb{P}(X_1 = s'|X_0 = s, A_0 = a).$$

A stationary policy π maps each state $s \in S$ to an action, independent of the history, i.e. $\pi : S \to A$. For each state $s \in S$ and action $a \in A$, the reward function is given by $r(s,a) : S \times A \to [0,1]$, representing the reward given to the agent at state s while taking action a. At each time point $t \in \mathbb{Z}_{\geq 0}$, with a given state $s_t \in S$, a policy takes an action a in A and obtains a reward $r(s,a)$. Then the state jumps to $s' \in S$ with probability $P(s'|s,a)$. For a discount factor $\gamma \in (0,1)$, the performance of a policy given a state $s \in S$ is evaluated by the value function $V^\pi(s)$, defined as

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

The performance of a policy starting at s and a given action a, is evaluated by the Q-function, defined by

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

in which the sequence $(s_t,a_t)_{t\geq 0}$ is taken according to the deterministic policy π. The goal of solving MDP is to find the optimal policy π_* as defined by $\pi_*(s) = \arg\max_\pi V_\pi(s), \forall s \in S$. This corresponds to solving Bellman's Optimality Equation of the Q-function Bellman (1956):

$$Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\max_{a' \in A} Q(s',a')\right], \tag{1}$$

and it is characterized by the deterministic policy $\pi(s) = \arg\max_{a\in A} Q^*(s,a)$ is the global optimal policy, where $Q^*(s,a) := \sup_\pi Q^\pi(s,a)$ and the expectation is taken with the probability distribution $P(\cdot|s,a)$. It is shown in Theorem 1.8 of (Agarwal et al. 2019) that the optimal policy is deterministic, which is a restatement of Bellman (1956). We only consider deterministic stationary policies in this paper. The policy iteration algorithm is a well-adopted approach to obtain the optimal policy. The following are the steps of classical policy iteration:

Algorithm 1: Classical Policy Iteration

```
   Input: Initial policy $\pi_0$, $k = 1$, stopping criterion
1  Each step with policy $\pi_k$;
2  while not stopping criterion do
3      for $x \in S,\ a \in A$ do
4          Compute $Q(s,a)$ via solving the linear system
               $Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} Q(s', \pi(s'))$        (2)
5      end
6      Update $\pi_k(s) = \arg\max_{a \in A} Q^{\pi_{k-1}}(s,a)$ for each $s \in S$;
7      $k \leftarrow k+1$;
8  end
   Output: Final policy $\pi$
```

However, the drawbacks of the classical policy iteration algorithm include: (1) It requires solving large linear systems of dimension $|S| \times |A|$ when computing Q-functions, which suffers a computational complexity of order $O(|S|^3)$. (2) It cannot be adopted when $|S|$ is infinitely large – even computing a Q-function is prohibited since it requires solving a linear system of order ∞.

In this paper, we research the methods of computing Q-functions in large or infinite state spaces, and in the latter case, we provide the local policy iteration algorithm with theoretical analysis and computing processes.

3 COSIMLA-ASSISTED POLICY ITERATION FOR LARGE FINITE STATE SPACE MDP

In this section, we elaborate on the application of COmbining SIMulation and numerical Linear Algebra (COSIMLA) algorithms into the computing process of solving Markov Decision Processes (MDP). We focus on scenarios with finite and large state space, large to the extent that even solving a linear system of equations can be prohibitive. In particular, we assume that $|A| \ll |S| < \infty$ in this section.

3.1 COSIMLA for Q-function Computing

Let a state $x \in S$ and an action $a \in A$, a discount factor $\gamma \in (0,1)$, and a deterministic stationary policy π be fixed. We introduce the process of computing $Q^\pi(x,a)$. We adopt the notations in Section 2. For a reward function $r(s,a)$, denote r^π the vector $(r(s,\pi(s)))_{s\in S}$ of dimension $|S|$. A Markov chain is induced with transition probability matrix $P^\pi : S \times S \to [0,1]$, where

$$P^\pi(x,y) = \mathbb{P}(X_1 = y | X_0 = x, \pi(x)).$$

To compute the Q-function, we introduce a new Markov chain. Let $z \notin S$ be a new state. We define a larger state space $\tilde{S} := S \cup \{z\}$ with transition probabilities on $\tilde{S}$ defined as

$$\begin{cases} \mathbb{P}(X_1 = s | X_0 = z, \pi) = P^\pi(x,s), & \forall s \in S, \\ \mathbb{P}(X_1 = s | X_0 = s', \pi) = \gamma P^\pi(s',s), & \forall s,s' \in S, \\ \mathbb{P}(X_1 = z | X_0 = s, \pi) = 1-\gamma, & \forall s \in S. \end{cases}$$

Then, the probability transition matrix on $\tilde{S}$ is expressed as

$$\tilde{P}^\pi_{x,a} := \begin{pmatrix} \gamma P^\pi & (1-\gamma) e_{N\times 1} \\ p_{a,x} & 0 \end{pmatrix},$$

where $p_{a,x} = (P(y|x,a))_{y\in S} \in \mathbb{R}^{1\times|S|}$ is a row vector, and $e_{|S|\times 1} \in \mathbb{R}^{|S|\times 1}$ is a column vector whose every element is 1. Let $\tau_z = \inf\{n \geq 1 : X_n = z\}$. For some fixed $x \in S$ and a policy π, denote $\mathbb{P}(\cdot), \mathbb{E}(\cdot)$ the probability and expectation under the matrix P^π, while $\tilde{\mathbb{P}}^\pi(\cdot), \tilde{\mathbb{E}}^\pi(\cdot)$ represents those under the matrix $\tilde{P}^\pi_{x,a}$. We denote $\mathbb{P}_s(\cdot) := \mathbb{P}(\cdot|X_0 = s)$, and $\mathbb{E}_s(\cdot) := \mathbb{E}(\cdot|X_0 = s)$, where s represents any state in S. For any function $w : \tilde{S} = S\bigcup\{z\} \to [0,1]$ with $w(z) = 0$, we define

$$\kappa^\pi_{x,a}(w) := \tilde{\mathbb{E}}^\pi_z\left[\sum_{j=0}^{\tau_z-1} w(X_j)\right],$$

representing that, under probability distribution $\tilde{P}^\pi$, the expected cumulated reward of w from the beginning to the first absorbing time τ_z. The following Theorem 1 presents a regenerative formula with respect to $\tilde{S}$ to compute $Q^\pi(x,a)$.

Theorem 1 For any state $x \in S$ and action $a \in A$, we have

$$Q^\pi(x,a) = \gamma\kappa^\pi_{x,a}(r^\pi) + r(x,a). \tag{3}$$

Theorem 1 gives an approach to computing the Q-function through a regenerative formula of a newly constructed Markov Chain $\tilde{S}$. From Theorem 1, we only need to compute or simulate $\kappa^\pi_{x,a}(r)$. To compute $\kappa^\pi_{x,a}(r)$, we apply the COSIMLA procedure, provided in page 2344-2345 of (Zheng et al. 2022). The specific process is elaborated as follows.

Assume that B is some subset of $\tilde{S}$ containing z such that inverting a matrix of order $|B|-1$ is computationally applicable. Let $B_z = B - \{z\}$ be a subset of B, $p_c = \left(\tilde{\mathbb{P}}^\pi_y(X_1 \in B^c)\right)_{y\in B_z}$ and $r_0 = (r^\pi(y))_{y\in B_z}$ be column vectors of dimension $|B_z|$, $\varphi_z = (\tilde{P}^\pi_{x,a}(z,x))_{x\in B_z}$ be a row vector of dimension $|B_z|$, $B_0 = \tilde{P}^\pi_{x,a}|_{B_z} \in \mathbb{R}^{|B_z|\times|B_z|}$, and $T = \inf\{n \geq 0 : X_n \in B^c\}$. Then

$$\kappa^\pi_{x,a}(r) = \varphi_z(I-B_0)^{-1}r_0 + \tilde{\mathbb{P}}^\pi_z(T < \tau_z)\tilde{\mathbb{E}}^\pi_z\left[\sum_{j=T}^{\tau_z-1} r(X_j)\middle| T < \tau_z\right],$$

$$\tilde{\mathbb{P}}^\pi_z(T < \tau_z) = \tilde{\mathbb{P}}^\pi_z(X_1 \in B^c) + \varphi_z(I-B_0)^{-1}p_c,$$

$$\tilde{\mathbb{P}}^\pi_z(X_T = y|T < \tau_z) = \frac{\tilde{P}^\pi_{x,a}(z,y) + \varphi_z(I-B_0)^{-1}p_y}{\tilde{\mathbb{P}}^\pi_z(X_1 \in A^c) + \varphi_z(I-B_0)^{-1}p_c}.$$

We refer to the proof and the asymptotic convergence in Section 2 of (Zheng et al. 2022). COSIMLA combines numerical linear algebra and simulation to compute $\kappa^\pi_{x,a}(r)$ using the above formulas. The computation processes are elaborated as follows:

1. Compute $\kappa_1 := \varphi_z(I-B_0)^{-1}r_0$, using numerical linear algebra to solve the linear system of order $|B_0|$;
2. Compute $\kappa_2 := \tilde{\mathbb{P}}^\pi_z(T < \tau_z)$. The value of $\tilde{\mathbb{P}}^\pi_z(X_1 \in B^c)$ is directly computed and $\varphi_z(I-B_0)^{-1}p_c$ is computed via applying numerical linear algebra to solve the linear system of order $|B_0|$;
3. Simulate $\tilde{\mathbb{E}}^\pi_z\left[\sum_{j=T}^{\tau_z-1} r(X_j)\middle| T < \tau_z\right]$, using Monte Carlo Simulation.

In the simulation process, we simulate n paths. The starting point y of each path follows the distribution $\tilde{\mathbb{P}}(X_T = y|T < \tau_z)$. We stop simulating each path when it achieves the endpoint z. Denote the cumulative rewards by $\bar{R}_1, \cdots, \bar{R}_n$. Since

$$\tilde{\mathbb{E}}^\pi_z\left[\sum_{j=T}^{\tau_z-1} r(X_j)\middle| T < \tau_z\right] = \sum_{y\in S}\tilde{\mathbb{P}}^\pi_z(X_T = y|T < \tau_z)\tilde{\mathbb{E}}^\pi_z\left[\sum_{j=T}^{\tau_z-1} r(X_j)\middle| T < \tau_z, X_T = y\right],$$

$\frac{1}{n}\sum_{i=1}^{n}\bar{R}_i$ is an estimation of $\tilde{\mathbb{E}}_z^{\pi}\left[\sum_{j=T}^{\tau_z-1} r(X_j)\middle| T<\tau_z\right]$. In conclusion,

$$\hat{Q}(x,a) := \gamma\cdot\left(\kappa_1+\kappa_2\cdot\frac{1}{n}\sum_{i=1}^{n}\bar{R}_i\right)+r(x,a)$$

is our estimation of $Q(x,a)$.

Remark 1 The above processes can be done in parallel to save computation time. The expected computational complexity of computing a single $Q(s,a)$'s value is of order $O(|B_z|^3)+O(n\mathbb{E}_{X_T}\tau_z)$. Considering the $O(|S|)$ complexity of setting z in $\tilde{S}$, the expected computational complexity is at most $O(|B_z|^3+\frac{n}{(1-\gamma)^2}+|S|)$.

If we assume further that any state $s\in S$ can only transfer to no more than M states in one step, where M is a positive integer, then the computational complexity of computing the value $Q(s,a)$ can potentially reach $O(|B_z|^3+\frac{n}{(1-\gamma)^2}+M)$, a constant regarding to $|S|$. The complexity of each step can potentially reach $O(|S|)$ dependency on S in this case.

Remark 2 To increase the computational accuracy, it is suggested to choose B such that (i) $\tilde{\mathbb{P}}_z(X_1\in B)$ is near to 1; (ii) X_T is easy to simulate; (iii) $|B|$ is as large as possible.

3.2 COSIMLA-Assisted Policy Iteration

In the policy iteration process, at each step, we use our approach proposed above to solve the Q-function. We conclude the COSIMLA-assisted policy iteration as follows.

Algorithm 2: COSIMLA-Assisted Policy Iteration for Large Finite State Space

```
  Input: Initial policy $\pi_0$, $k=1$, stopping criterion
1 Each step with policy $\pi_k$;
2 while not stopping criterion do
3     while $x\in S$ do
4         Construct $\tilde{P}_x^{k-1}$;
5         Compute $Q^{\pi_{k-1}}(s,a)$ for every pair $(s,a)$ via COSIMLA in Section 3.1;
6         Update $\pi_k(s)=\arg\max_{a\in A} Q^{\pi_{k-1}}(s,a)$, $s\in S$;
7     end
8     $k\leftarrow k+1$
9 end
  Output: Final policy $\pi$
```

When $|B_z| << |S|$, then the COSIMLA-Assisted Policy Iteration Algorithm in computing the Q-function has a computational complexity of order $O(|S|^2)$ dependency on $|S|$, which reduces the computational complexity compared to $O(|S|^3)$ dependency on $|S|$ of classical policy iteration in one single step.

4 INFINITE STATE SPACE WITH LOCAL POLICY ITERATION

We research policy improvement in the case $|S|=\infty$ in this section. Let $S_0\subset S$ be a finite subset. Assume that every policy induces a positive recurrent Markov chain and that any state can only transfer to a finite number of other states in one jump (finite transfer). Our approach for $|S|=\infty$ is to only update policies locally on S_0 and keep the actions on the remaining states unchanged. The local policy iteration is defined as

$$\pi_{k+1}(s)=\begin{cases}\arg\max_{a\in A} Q^{\pi_k}(s,a), s\in S_0,\\ \pi_k(s), s\in S_0^c.\end{cases} \tag{4}$$

Definition 1 A policy $\tilde{\pi}_*$ is called locally optimal if it satisfies the following equation:

$$Q^{\tilde{\pi}_*}(s,a) = r(s,a) + \gamma \sum_{s' \in S_0} P(s'|s,a) \max_{a' \in A} Q^{\tilde{\pi}_*}(s',a') + \gamma \sum_{s' \in S_0^c} P(s'|s,a) Q^{\tilde{\pi}_*}(s', \pi_0(s')). \tag{5}$$

(5) is called the local-Bellman Optimality Equation. Furthermore, for a policy π and discount factor $\gamma \in (0,1)$, the local Bellman optimality operator $\tilde{T} : \Pi \times \mathbb{R}^{|S| \times |A|} \to \mathbb{R}^{|S| \times |A|}$ is defined to be

$$(\tilde{T}(\pi, Q))(s,a) := r(s,a) + \gamma \sum_{s' \in S_0} P(s'|s,a) \max_{a' \in A} Q(s',a') + \gamma \sum_{s' \in S_0^c} P(s'|s,a) Q(s', \pi(s')).$$

For simplicity, $\tilde{T}(\pi, Q^\pi)$ is denoted by $\tilde{T}Q^\pi$.

We next show that the locally optimal policy $\tilde{\pi}_*$ exists and the local policy iteration process converges to some limit $\tilde{\pi}_\infty$, i.e., $\pi_k \to \pi_\infty$. Then, we show $\tilde{\pi}_* = \pi_\infty$. Furthermore, the globally optimal policy π^* is defined to be satisfying (1). We give bounds for the error gap between the Q-functions of π_∞ and π^*, i.e., $\|Q^{\pi_\infty} - Q^{\pi^*}\|_\infty$.

Theorem 2 Given any deterministic policy π_0, there exists $\tilde{\pi}_*$, satisfying that

$$Q^{\tilde{\pi}_*}(s,a) = r(s,a) + \gamma \sum_{s' \in S_0} P(s'|s,a) \max_{a' \in A} Q^{\tilde{\pi}_*}(s',a') + \gamma \sum_{s' \in S_0^c} P(s'|s,a) Q^{\tilde{\pi}_*}(s', \pi_0(s')).$$

Furthermore, $\tilde{T}(\tilde{\pi}_*, Q^{\tilde{\pi}_*}) = Q^{\tilde{\pi}_*}$, $\tilde{\pi}_*|_{S_0} = \pi_0|_{S_0}$, and local policy iteration converges to $\tilde{\pi}_*$.

Corollary 3 (Convergence rate) For any $k \geq 1$,

$$\|Q^* - Q^{\pi_{k+1}}\|_\infty \leq \gamma \|Q^* - Q^{\pi_k}\|_\infty.$$

Intuitively, given the deterministic policy π_0, updating the policy in a finite subset S_0 will locally increase the reward of the policy. Since the policy outside S_0 remains unchanged, this process will not make the policy "worse".

We next bound the difference between the locally optimal policy $\pi_\infty = \tilde{\pi}_*$ and the globally optimal policy π_*. The following definition is to give measurements to the optimality of the initial policy π_0.

A policy π is called 1-step locally optimal in S_0^c if $\pi(s) = \arg\max_{a \in A} r(s,a)$. The total variance difference of two probability distributions p_1 and p_2 on S is defined as $\|p_1 - p_2\|_{TV} := 2\sup_A (p_1(A) - p_2(A))$, in which the sup is taken through all the subsets of S. We then have the following theorem.

Theorem 4 Assume that π_0 is 1-step locally optimal in S_0^c. Denote the locally optimal policy by $\tilde{\pi}_*$, the globally optimal policy by π_*, and $\tilde{Q} = Q^{\tilde{\pi}_*}, Q^* = Q^{\pi_*}$. Then we have

$$\begin{aligned} \|Q^* - \tilde{Q}\|_\infty &\leq \frac{\gamma^2}{2(1-\gamma)^2} \sup_{s' \in S_0^c} \max_{a' \in A} \|P(\cdot|s',a') - P(\cdot|s', \pi_0(s'))\|_{TV} \\ &\leq \frac{2\gamma^2}{(1-\gamma)^2}. \end{aligned}$$

Remark 3 Theorem 4 provides both an instance-dependent error bound and a worst-case bound of the error between the locally optimal policy and the globally optimal policy.

To evaluate a policy in S_0, Theorem 1 and the computing procedure of $\kappa^\pi_{x,a}$ still works when $|S| = \infty$. We conclude the local policy iteration as follows.

Algorithm 3: COSIMLA-Assisted Local Policy Iteration for Infinite State Space

```
    Input: Initial policy $\pi_0$, $k=1$, stopping criterion, A finite subset $S_0$
1   Each step with policy $\pi_k$;
2   while not stopping criterion do
3       for $s \in S_0$ do
4           Compute $Q^{\pi_{k-1}}(s,a)$ for every pair $(s,a)$ via COSIMLA in Section 3.1;
5           Update $\pi_k(s) = \arg\max_{a\in A} Q^{\pi_{k-1}}(s,a)$ for $s \in S_0$;
6       end
7       $k \leftarrow k+1$
8   end
    Output: Final policy $\pi$
```

Remark 4 Taking the notations defined in Section 3, the computational complexity of a Q-function is $O(|B_z|^3 + \frac{n}{(1-\gamma)^2})$, which is finite. The positive recurrent assumption guarantees that the simulation process can stop within a finite time.

5 NUMERICAL EXPERIMENT

In this section, we design experiments in finite S settings to research the accuracy and efficiency of COSIMLA-assisted policy iteration. Notations in Section 2 are adopted. We design an MDP on a birth-death chain on a subset of non-negative integers $S = \{0,1,\cdots,T\}$. Let the action space be $A = \{a_1,\cdots,a_K\}$. For each $s \in S$ and $i \in [K]$, let $p_s^{(i)}$ and $q_s^{(i)}$ be randomly generated positive real numbers satisfying that $p_s^{(i)} + q_s^{(i)} \le 1$, and $r(s,a_i) \in [0,1]$ is also chosen randomly. The states $(X_n : n \ge 0)$ are defined as

$$X_{n+1} = \max(\min(X_n + Z_n - 1, T), 0),$$

where $Z_n's$ are independent random variables taking values in $\{0,1,2\}$. For any state s and action a_i, Z_n is defined to follow the probability distribution according to action a_i and state $X_n = s$,

$$\mathbb{P}(Z_n = 0|a_i, X_n = s) = q_s^{(i)}, \mathbb{P}(Z_n = 2|a_i, X_n = s) = p_s^{(i)}, \mathbb{P}(Z_n = 0|a_i, X_n = s) = 1 - p_s^{(i)} - q_s^{(i)},$$

and on the boundary,

$$\mathbb{P}(Z_n \in \{0,1\}|a_i, X_n = 0) = p_0^{(i)}, \mathbb{P}(Z_n = 2|a_i, X_n = 0) = 1 - p_0^{(i)},$$

$$\mathbb{P}(Z_n = 0|a_i, X_n = T) = q_T^{(i)}, \mathbb{P}(Z_n \in \{1,2\}|a_i, X_n = T) = 1 - q_T^{(i)}.$$

Then, with a given deterministic stationary policy π, the transition matrix is

$$P^\pi = \begin{bmatrix} 1-p_0^{\pi(0)} & p_0^{\pi(0)} & 0 & & \\ q_1^{\pi(1)} & 1-q_1^{\pi(1)}-p_1^{\pi(1)} & p_1^{\pi(1)} & \cdots & \\ 0 & q_2^{\pi(2)} & 1-q_2^{\pi(2)}-p_2^{\pi(2)} & p_2^{\pi(2)} & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ & & & q_T^{\pi(T)} & 1-q_T^{\pi(T)} \end{bmatrix},$$

which is of dimension $|S|$. In the COSIMLA-assisted algorithm, for each state $s \in S$ and positive integers b and n, we set $B_0 = \{(s-b)\vee 0, (s-b)\vee 0+1, \cdots, (s+b)\wedge |S|\}$ and simulate n paths of cumulative rewards. We design the following two experiments. In every experiment setting, the classical algorithm and the COSIMLA-assisted algorithm work on a task to compute the Q-function with a given policy π. We denote the Q-function computed via classical algorithm by $\hat{Q}_1$, and that via COSIMLA-assisted algorithm by $\hat{Q}_2$.

Experiment 1. Let K, γ, and the number of simulation paths n be fixed. We increase the size of state space, $|S|$, to compare the implementation CPU time between the classical algorithm and the COSIMLA-assisted algorithm. The truncation set size b is fixed throughout this experiment. Results are presented in Table 1.

Experiment 2. We fix $|S|$ and compute the Q-function via the COSIMLA-assisted algorithm on different sizes b and different numbers of simulation paths n. We run our approach to compute the Q-function for $N = 20$ times, denoted as $\hat{Q}_2^{(1)}, \cdots, \hat{Q}_2^{(N)}$. We regard $\hat{Q}_1$ as the true value of the Q-function in our setting and report the mean l^∞ error $\tilde{E}$, defined as $\tilde{E} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{Q}_1 - \hat{Q}_2^{(i)}\|_\infty$. Results are presented in Table 2.

All experiments are conducted on a Mac Book Pro laptop with an 8-core CPU and 16GB memory. The implementation CPU time and the result of the estimations of Q^π are recorded.

Table 1: The CPU time comparison of Classical Algorithm and COSIMLA-assisted Algorithm.

Settings			Classical	COSIMLA-assisted
$\lvert S\rvert = 1\times 10^3$	$K = 3$	$\gamma = 0.85$, $n = 50$	1.9 s	21.8 s
$\lvert S\rvert = 5\times 10^3$	$K = 3$	$\gamma = 0.85$, $n = 50$	1 min 15 s	2 min 10 s
$\lvert S\rvert = 1\times 10^4$	$K = 3$	$\gamma = 0.85$, $n = 50$	6 min 47 s	3 min 50 s
$\lvert S\rvert = 2\times 10^4$	$K = 3$	$\gamma = 0.85$, $n = 50$	Memory Crashed	9 min 3 s
$\lvert S\rvert = 10^3$	$K = 3$	$\gamma = 0.7$, $n = 50$	1.9 s	12.8 s
$\lvert S\rvert = 5\times 10^3$	$K = 3$	$\gamma = 0.7$, $n = 50$	1 min 11 s	1 min 15 s
$\lvert S\rvert = 10^4$	$K = 3$	$\gamma = 0.7$, $n = 50$	8 min 40 s	3 min 1s
$\lvert S\rvert = 1\times 10^3$	$K = 2$	$\gamma = 0.8$, $n = 10$	1.2 s	3.3 s
$\lvert S\rvert = 5\times 10^3$	$K = 2$	$\gamma = 0.8$, $n = 10$	36.4 s	24.4s
$\lvert S\rvert = 1\times 10^4$	$K = 2$	$\gamma = 0.8$, $= 10$	2 min 54 s	1 min 6 s
$\lvert S\rvert = 2\times 10^4$	$K = 2$	$\gamma = 0.8$, $n = 10$	>20 min	3 min 34s

In Experiment 1, whenever the classical algorithm provides $\hat{Q}_1$, the l^∞ differences between $\hat{Q}_1$ and $\hat{Q}_2$ are less than 10^{-6}. In Experiment 2, the mean of the $\|Q\|_{l^\infty}$ in each setting is greater than 1, thus we report the absolute mean error. Furthermore, we have the following observations.

- From Experiment 1, we observe that, starting from approximately $|S| = 10^4$, the COSIMLA algorithm is less time-consuming and more memory-preserving, compared to the classical algorithm.
- From Experiment 2, we observe that, for a fixed number of simulation times n, when the size of truncation set b increases, the error between the $\hat{Q}_2$ and the true Q-function decreases beyond the order of magnitude.
- From Experiment 2, we observe that, for a fixed size of truncation set b, when the number of simulation times n increases, the error between $\hat{Q}_2$ and the true Q-function decreases, yet remains in the same order of magnitude.

Table 2: The CPU time and the mean l^∞ error of COSIMLA-assisted Algorithm for different numbers of simulation paths n and truncation set size b.

Settings	Mean Error $\tilde{E}$	Mean CPU Time
$\lvert S\rvert = 1\times 10^3$ $K=2$, $\gamma=0.8$		
$n=10$, $b=10$	3.9×10^{-3}	2.4 s
$n=25$, $b=10$	2.6×10^{-3}	5.4 s
$n=50$, $b=10$	1.8×10^{-3}	9.9 s
$n=10$, $b=20$	1.4×10^{-6}	2.7 s
$n=25$, $b=20$	8.5×10^{-7}	5.5 s
$n=50$, $b=20$	5.8×10^{-7}	10.3 s
$n=10$, $b=30$	2.16×10^{-10}	3.4 s
$n=25$, $b=30$	1.29×10^{-10}	6.4 s
$n=50$, $b=30$	9.8×10^{-11}	11.4 s
$\lvert S\rvert = 5\times 10^3$ $K=3$, $\gamma=0.77$		
$n=10$, $b=10$	2.5×10^{-3}	23.9 s
$n=25$, $b=10$	1.5×10^{-3}	42.6 s
$n=50$, $b=10$	1.0×10^{-3}	1 min 14 s
$n=10$, $b=20$	1.6×10^{-6}	27.7 s
$n=25$, $b=20$	8.6×10^{-7}	45.0 s
$n=50$, $b=20$	6.6×10^{-7}	1 min 15 s
$n=10$, $b=30$	1.14×10^{-10}	29.6 s
$n=25$, $b=30$	6.5×10^{-11}	49.7 s
$n=50$, $b=30$	4.67×10^{-11}	1 min 21 s

6 CONCLUSION

In this paper, we propose an approach to compute Q-functions (Theorem 1) in Markov Decision Processes with the assistance of combining numerical linear algebra and simulation (COSIMLA) (Zheng et al. 2022) to evaluate $\kappa^{\pi}_{x,a}$ in Theorem 1. We also introduce a local policy iteration approach in infinite state space Markov Decision Processes with COSIMLA's assistance and give a convergence rate of the local policy iteration algorithm and a bound for the error between the local optimal policy and the global optimal policy. The COSIMLA-Assisted Policy iteration shows high accuracy, low computational complexity, and low memory consumption in large state spaces.

ACKNOWLEDGMENTS

We would love to express our appreciation and thanks to the anonymous reviewers, for their time, comments and suggestions. They are very helpful to the improvement of our manuscript.

A PROOFS OF THEOREMS

A.1 Proof of Theorem 1

For $s \in S$, we have

$$\begin{aligned}
V^{\pi}(s) &= \mathbb{E}_s\left[\sum_{j=0}^{\infty}\gamma^j r^{\pi}(X_j)\right] = \sum_{j=0}^{\infty}\sum_{t\in S}\mathbb{P}_s(X_j=t)\gamma^j r(X_j) \\
&= \sum_{j=0}^{\infty}\sum_{t\in S}\gamma^j P^j(s,t)r(X_j) = \sum_{j=0}^{\infty}\sum_{t\in S}(\gamma P)^j(s,t)r(X_j) \\
&= \sum_{j=0}^{\infty}\sum_{t\in S}\tilde{\mathbb{P}}_s^{\pi}(X_j=t,\tau_z>t)r(X_j) = \tilde{\mathbb{E}}_z^{\pi}\left[\sum_{j=0}^{\tau_z-1}r(X_j)\middle|X_0=s\right].
\end{aligned}$$

Note that

$$\kappa_{x,a}^{\pi}(r) = \tilde{\mathbb{E}}_z^{\pi}\left[\sum_{j=0}^{\tau_z-1}r(X_j)\right] = \sum_{s\in S}\tilde{P}_{x,a}^{\pi}(z,s)\tilde{\mathbb{E}}_x^{\pi}\left[\sum_{j=0}^{\tau_z-1}r(X_j)\middle|X_0=s\right] = \sum_{s\in S}P(s|x,a)V^{\pi}(s),$$

thus, $Q^{\pi}(x,a)=\gamma\kappa_{x,a}^{\pi}(r)+r(x,a)$. Hence, we complete the proof. □

A.2 Proof of Theorem 2

First, we show $Q^{\pi_{k+1}} \geq Q^{\pi_k}$. For any action a, any state s and any policy π, denote

$$\pi(a|s) = \begin{cases} 1, & \textit{if } \pi(s)=a, \\ 0, & \textit{otherwise}. \end{cases}$$

For any two states $s,s' \in S$ and any two actions $a,a' \in A$, define $\bar{P}^{\pi} \in \mathbb{R}^{|S||A|\times|S||A|}$ by $\bar{P}^{\pi}(s',a'|s,a) := \mathbb{P}(X_1=s',A_1=s'|X_0=s,A_0=a)$. Then, we have

$$\begin{aligned}
Q^{\pi_k}(s,a) &= r(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)\sum_{a'\in A}\pi_k(a'|s')Q^{\pi_k}(s',a') \\
&\leq r(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)\sum_{a'\in A}\pi_{k+1}(a'|s')Q^{\pi_k}(s',a') \\
&= r(s,a)+\gamma\sum_{s'\in S}\sum_{a'\in A}\bar{P}^{\pi_{k+1}}(s',a'|s,a)Q^{\pi_k}(s',a') \\
&\leq \sum_{j=0}^{\infty}\gamma^j(\bar{P}^{\pi_{k+1}})^j(s',a'|s,a)r(s,a) = Q^{\pi_{k+1}}(s,a),
\end{aligned}$$

in which we used the definition of local policy iteration (4). Since any Q-function is bounded by $\frac{1}{1-\gamma}$, the sequence $\{Q^{\pi_k}\}_{k=1}^{\infty}$ has a limit. We next show that $Q^{\pi_{k+1}} \geq \tilde{T}Q^{\pi_k}$.

$$\begin{aligned}
Q^{\pi_{k+1}}(s,a) &= r(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)\sum_{a'\in A}\pi_{k+1}(a'|s')Q^{\pi_k}(s',a') \\
&= r(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)Q^{\pi_{k+1}}(s',\pi_{k+1}(s')) \\
&\geq r(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)Q^{\pi_k}(s',\pi_{k+1}(s')) \\
&= r(s,a)+\gamma\sum_{s'\in S_0}P(s'|s,a)\max_{a'\in A}Q^{\pi_k}(s',a')+\gamma\sum_{s'\in T_0}P(s'|s,a)Q^{\pi_k}(s',\pi_0(s')) \\
&= (\tilde{T}Q^{\pi_k})(s,a).
\end{aligned}$$

Observing that $\tilde{T}Q^{\pi_k} \geq Q^{\pi_k}$, we then obtain

$$\cdots \leq Q^{\pi_k}(s,a) \leq \tilde{T}Q^{\pi_k}(s,a) \leq Q^{\pi_{k+1}}(s,a) \leq \cdots$$

For any two Q-functions with the same policy π_0 in S_0^c,

$$\begin{aligned}
\sup_{s\in S, a\in A} |\tilde{T}Q_1(s,a) - \tilde{T}Q_1(s,a)| &= \sup_{s\in S, a\in A} \gamma \bigg| \sum_{s'\in S_0} P(s'|s,a)\left(\max_{a'\in A} Q_1(s',a') - \max_{a''\in A} Q_2(s',a'')\right) \\
&\quad + \sum_{s'\in S_0^c} P(s'|s,a) \sum_{a'\in A} \pi_0(a'|s')(Q_1(s',a') - Q_2(s',a'))\bigg| \\
&\leq \sup_{s\in S, a\in A} \gamma \sum_{s'\in S} P(s'|s,a) \max_{a'\in A} |Q_1(s',a') - Q_2(s',a')| \\
&\leq \sup_{s\in S, a\in A} \gamma |Q_1(s,a) - Q_2(s,a)|.
\end{aligned}$$

In other words,

$$\left\|\tilde{T}Q_1 - \tilde{T}Q_2\right\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty. \tag{6}$$

Let $\tilde{Q}^*(s,a) = \lim_{k\to\infty} Q^{\pi_k}(s,a)$ for every $(s,a) \in S \times A$, then we note that

$$\begin{aligned}
\left\|\tilde{T}\tilde{Q}^* - \tilde{Q}^*\right\|_\infty &\leq \left\|\tilde{T}\tilde{Q}^* - \tilde{T}Q^{\pi_k}\right\|_\infty + \left\|\tilde{T}Q^{\pi_k} - Q^{\pi_k}\right\|_\infty + \left\|Q^{\pi_k} - \tilde{Q}^*\right\|_\infty \\
&\leq (\gamma+1)\left\|\tilde{Q}^* - Q^{\pi_k}\right\|_\infty + \|Q^{\pi_{k+1}} - Q^{\pi_k}\|_\infty \\
&\to 0,
\end{aligned}$$

as $k \to \infty$. Thus, $\tilde{T}\tilde{Q}^* = \tilde{Q}^*$, and $\tilde{\pi}_*(s) := \arg\max_a \tilde{Q}^*(s,a), s \in S_0$ satisfies (5). Hence we complete the proof. □

A.3 Proof of Corollary 3

Since (6), we have $\|Q^* - Q^{\pi_{k+1}}\|_\infty \leq \left\|Q^* - \tilde{T}Q^{\pi_k}\right\|_\infty = \left\|\tilde{T}Q^* - \tilde{T}Q^{\pi_k}\right\|_\infty \leq \gamma\|Q^* - Q^{\pi_k}\|_\infty$. Then we complete the proof. □

A.4 Proof of Theorem 4

Note that,

$$\tilde{Q}(s,a) = r(s,a) + \gamma \sum_{s'\in S_0} P(s'|s,a) \max_{a'\in A} \tilde{Q}(s',a') + \gamma \sum_{s'\in S_0^c} P(s'|s,a)\tilde{Q}(s',\pi_0(s')),$$

and

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'\in S} P(s'|s,a) \max_{a'\in A} Q^*(s',a').$$

Taking the difference between them and noting that $\tilde{\pi}_*|_{S_0} = \pi_0|_{S_0}$, we have,

$$\begin{aligned}
&Q^*(s,a) - \tilde{Q}(s,a) \\
=&\gamma \sum_{s'\in S} P(s'|s,a)(\max_{a'\in A} Q^*(s',a') - \max_{a'\in A} \tilde{Q}(s',a')) + \gamma \sum_{s'\in S_0^c} P(s'|s,a)(\max_{a'\in A} \tilde{Q}(s',a') - \tilde{Q}(s',\pi_0(s'))) \\
\leq&\gamma \sum_{s'\in S} P(s'|s,a) \max_{a'\in A}(Q^*(s',a') - \tilde{Q}(s',a')) + \gamma \sum_{s'\in S_0^c} P(s'|s,a)(\max_{a'\in A} \tilde{Q}(s',a') - \tilde{Q}(s',\pi_0(s'))) \\
\leq&\gamma\|\tilde{Q} - Q^*\|_\infty + \gamma \sum_{s'\in S_0^c} P(s'|s,a)(\max_{a'\in A} \tilde{Q}(s',a') - \tilde{Q}(s',\pi_0(s'))).
\end{aligned}$$

This implies that

$$
\begin{aligned}
&(1-\gamma)\|\tilde{Q}-Q^*\|_\infty \\
&\leq \sup_{s\in S} \gamma \sum_{s'\in S_0^c} P(s'|s,a)(\max_{a'\in A} \tilde{Q}(s',a') - \tilde{Q}(s',\pi_0(s'))) \\
&\leq \gamma \sup_{s'\in S_0^c} \left(\max_{a'\in A} \tilde{Q}(s',a') - \tilde{Q}(s',\pi_0(s')) \right) \\
&= \gamma \sup_{s'\in S_0^c} \max_{a'\in A} \left(r(s',a') + \gamma \sum_{s''\in S} P(s''|s',a')\tilde{Q}(s'',\tilde{\pi}_*(s'')) - r(s',\tilde{\pi}_*(s')) - \gamma \sum_{s''\in S} P(s''|s',\tilde{\pi}_*(s'))\tilde{Q}(s'',\tilde{\pi}_*(s'')) \right) \\
&\leq \gamma^2 \sup_{s'\in S_0^c} \max_{a'\in A} \sum_{s''\in S} (P(s''|s',a') - P(s''|s',\tilde{\pi}_*(s'')))\tilde{Q}(s'',\tilde{\pi}_*(s')) \\
&\leq \frac{\gamma^2}{1-\gamma} \sup_{s'\in S_0^c} \max_{a'\in A} \sum_{s''\in S} \left(P(s''|s',a') - P(s''|s',\tilde{\pi}_*(s')) \right)_+ \\
&\leq \frac{\gamma^2}{2(1-\gamma)} \sup_{s'\in S_0^c} \max_{a'\in A} \|P(\cdot|s',a') - P(\cdot|s',\pi_0(s'))\|_{TV},
\end{aligned}
$$

where $(x)_+$ denotes $\max(x,0)$ for any real number x. Finally, noting that $\|P(\cdot|s',a') - P(\cdot|s',\pi_0(s'))\|_{TV} \leq 2$ always holds, we complete the proof. $\square$

REFERENCES

Agarwal, A., N. Jiang, S. M. Kakade, and W. Sun. 2019. "Reinforcement Learning: Theory and Algorithms". *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep* 32:96.

Bellman, R. 1956. "Dynamic programming and Lagrange Multipliers". *Proceedings of the National Academy of Sciences* 42(10):767–769.

Bertsekas, D. P. 2011. "Approximate Policy Iteration: A Survey and Some New Methods". *Journal of Control Theory and Applications* 9(3):310–335.

Glynn, P. W. and Z. Zheng. 2023. "Cosimla with General Regeneration Set to Compute Markov Chain Stationary Expectations". In *2023 Winter Simulation Conference (WSC)*, 469–479. IEEE.

Ye, Y. 2011. "The Simplex and Policy-iteration Methods are Strongly Polynomial for the Markov Decision Problem with a Fixed Discount Rate". *Mathematics of Operations Research* 36(4):593–603.

Zheng, Z., A. Infanger, and P. W. Glynn. 2022. "Combining Numerical Linear Algebra with Simulation to Compute Stationary Distributions". In *2022 Winter Simulation Conference (WSC)*, 2342–2353. IEEE.

AUTHOR BIOGRAPHIES

YIFU TANG is currently a PhD Student at University of California, Berkeley. His email address is yifutang@berkeley.edu.

PETER W. GLYNN is the Thomas Ford Professor in the Department of Management Science and Engineering (MS&E) at Stanford University. He is a Fellow of INFORMS and of the Institute of Mathematical Statistics, has been co-winner of Best Publication Awards from the INFORMS Simulation Society in 1993, 2008, and 2016, and was the co-winner of the John von Neumann Theory Prize from INFORMS in 2010. In 2012, he was elected to the National Academy of Engineering. His research interests lie in stochastic simulation, queueing theory, and statistical inference for stochastic processes. His email address is glynn@stanford.edu and his homepage is https://web.stanford.edu/~glynn/index.html.

ZEYU ZHENG is an associate professor in the Department of Industrial Engineering & Operations Research at University of California Berkeley. His email address is zyzheng@berkeley.edu and his homepage is http://zheng.ieor.berkeley.edu.