Randomised Procedures for Initialising and Switching Actions in Policy Iteration

Shivaram Kalyanakrishnan
Indian Institute of Technology Bombay
Mumbai 400076 India
shivaram@cse.iitb.ac.in

Neeldhara Misra
Indian Institute of Technology Gandhinagar
Gandhinagar 382355 India
neeldhara.misra@gmail.com

Aditya Gopalan
Indian Institute of Science
Bengaluru 560012 India
aditya@ece.iisc.ernet.in

Abstract

Policy Iteration (PI) (Howard 1960) is a classical method for computing an optimal policy for a finite Markov Decision Problem (MDP). The method is conceptually simple: starting from some initial policy, "policy improvement" is repeatedly performed to obtain progressively dominating policies, until eventually, an optimal policy is reached. Being remarkably efficient in practice, PI is often favoured over alternative approaches such as Value Iteration and Linear Programming. Unfortunately, even after several decades of study, theoretical bounds on the complexity of PI remain unsatisfactory. For an MDP with n states and k actions, Mansour and Singh (1999) bound the number of iterations taken by Howard's PI, the canonical variant of the method, by $O(k^n/n)$. This bound merely improves upon the trivial bound of k^n by a linear factor. However, a randomised variant of PI introduced by Mansour and Singh (1999) does yield an exponential improvement, with its expected number of iterations bounded by $O\left(\left(\left(1 + 2/\log_2(k)\right) k/2\right)^n\right)$.

With the objective of furnishing improved upper bounds for PI, we introduce two randomised procedures in this paper. Our first contribution is a routine to find a good initial policy for PI. After evaluating a number of randomly generated policies, this procedure applies a novel criterion to pick one to initialise PI. When PI is subsequently applied, we show that the expected number of policy evaluations—including both the initialisation and the improvement stages—remains bounded in expectation by $O(k^{n/2})$. The key construction employed in this routine is a *total order* on the set of policies. Our second contribution is a randomised action-switching rule for PI, which admits a bound of $(2 + \ln(k-1))^n$ on the expected number of iterations. To the best of our knowledge, this is the tightest complexity bound known for PI when $k \geqslant 3$.

1 Introduction

The Markov Decision Problem (MDP) (Bellman 1957; Puterman 1994) has been in use for several decades as a formal framework for decision-making under uncertainty. An MDP models an agent whose actions result in stochastic state transitions, while yielding associated rewards. The agent's natural aim is to consistently take actions that lead to high long-term reward. Thus, given an MDP, the central computational question is that of determining an optimal way for the agent to act. This problem, referred to as MDP *planning*, is the focus of this paper.

Formally, an MDP $M = (S, A, R, T, \gamma)$ has a set of states S in which an agent can be, and a set of actions A that the agent can execute. Upon taking action a from state s, the agent receives a reward r, assumed to be a real-valued random variable with mean $R(s, a)$. The action a also transports the agent to a state s', selected from S at random with probability $T(s, a, s')$. In this paper, we assume that S and A are both finite, with $|S| = n \geqslant 2$, and $|A| = k \geqslant 2$.

To fully specify M, we need to define an objective for the agent. A *policy* (assumed stationary, deterministic, and Markovian) is a mapping from S to A: when *following* a policy π, the agent takes action $\pi(s)$ when in state s. A natural objective is to find a policy that maximises the agent's expected long-term reward. Consider an agent that starts in some state s_0 at time 0 and continually follows π. The agent thereby encounters a trajectory over time: $\langle s_0, \pi(s_0), r_0, s_1, \pi(s_1), r_1, s_2, \ldots \rangle$. The *value* of state s under policy π is given by

$$V^\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right], \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor. Setting $\gamma < 1$, as done under the *discounted reward* setting (Sutton and Barto 1998, see Section 3.3), ensures that values are well-defined. In certain MDPs, such as those with a "sink state", values continue to be well-defined for all states and policies under the *total reward* setting (Fearnley 2010), wherein γ is set to 1. In this paper, we make no strict assumption on the reward setting used in defining the MDP—as we will see shortly, we only require that the Policy Iteration algorithm (Howard 1960) be applicable. This algorithm can also be used under the *average reward* setting (Mahadevan 1996), which calls for a more nuanced definition of values. To keep our exposition simple, we stick to (1) throughout this paper, and assume $\gamma < 1$ when we refer to $1/(1-\gamma)$.

Let Π be the set of k^n distinct policies corresponding to M. It is a key property (Bellman 1957) that this set contains a policy π^* such that for $\forall s \in S, \forall \pi \in \Pi$,

$$V^{\pi^\star}(s) \geqslant V^\pi(s). \tag{2}$$

Such a policy $\pi^\star$ is called an *optimal* policy (in general there could be multiple optimal policies for an MDP).

The problem we consider is precisely that of finding an optimal policy for a given MDP $M = (S, A, R, T, \gamma)$.[1] Specifically, we examine the Policy Iteration (PI) (Howard 1960) family of algorithms, which follow the general template of starting with some initial policy, and repeatedly computing *local* improvements until an optimal policy is found. In most MDPs encountered in practice, the number of such improvements (or "iterations") needed to find an optimal policy is relatively small compared to the size of the MDP. However, there do exist MDPs wherein PI may be forced to pass through lengthy improvement sequences (Fearnley 2010). From a theoretical standpoint, it has proven hard to establish tight upper bounds on the number of iterations taken by PI: the best existing upper bounds are of the form $O((k/2)^n)$, when k is large, for a randomised variant of PI proposed by Mansour and Singh (1999). Indeed the aim of this paper is to provide better formal guarantees. To this end, we propose two algorithms.

1. Our first contribution is a randomised procedure titled "**Guess-and-Max**", which picks a good initial policy for PI. We show that regardless of the PI algorithm subsequently applied, the total number of policy evaluations remains bounded by $O(k^{n/2})$ in expectation..
2. Our second contribution is a randomised improvement operator for a variant of PI called "Simple" PI (Melekopoglou and Condon 1994). We bound the expected number of iterations in the resulting algorithm, which we call Randomised Simple PI (**RSPI**), by $(2 + \ln(k-1))^n$.

This paper is organised as follows. In Section 2, we give an overview of approaches for MDP planning, and therein highlight the appeal of PI, which we describe in detail. In sections 3 and 4, we present our two algorithms and the corresponding analyses. We summarise our results and conclude the paper in Section 5.

2 Planning Algorithms and their Complexity

In this section, we introduce some basic results pertaining to MDPs before considering algorithms that build upon these results. First, observe from (1) that the *value function* V^π of a policy π, which yields values at each state, can be specified recursively: $\forall s \in S$,

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s').$$

This definition implies that the value function of a policy can be computed efficiently by solving a system of n linear equations, which are called Bellman's Equations. Clearly this operation—called *policy evaluation*—requires no more than $O(n^3)$ arithmetic operations. With an additional $O(n^2 k)$ operations, the *action value function* function Q^π corresponding to policy π can be obtained from V^π: $\forall s \in S, \forall a \in A$,

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s').$$

$Q^\pi(s, a)$ denotes the expected long-term reward obtained when the agent executes action a at state s once, and *thereafter* executes π. The value function corresponding to an optimal policy $\pi^\star$ is termed the *optimal value function*, denoted $V^\star$. Indeed $V^\star$ is the unique solution of another set of equations called Bellman's Optimality Equations: $\forall s \in S$,

$$V^\star(s) = \max_{a \in A} \left(R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^\star(s') \right).$$

$V^\star$, $Q^\star$ and $\pi^\star$ are easy to derive from each other. If $V^\star$ is known, it directly yields $Q^\star = Q^{\pi^\star}$, from which $\pi^\star$ can be readily obtained: $\forall s \in S$, $\pi^\star(s) = \operatorname{argmax}_{a \in A} Q^\star(s, a)$. If we start with $\pi^\star$, then policy evaluation yields $V^\star$, which, in turn, gives $Q^\star$.

Owing to the presence of the $\max$ operator, solving Bellman's Optimality Equations is not as straightforward as policy evaluation. Consequently, numerous planning algorithms have been proposed in the literature. These algorithms can be placed in the following broad categories.

2.1 Linear Programming (LP)

$V^\star$ and $\pi^\star$ can be obtained by solving a linear program based on Bellman's Optimality Equations; typical formulations either involve n variables and nk constraints, or nk variables and n constraints (Littman, Dean, and Kaelbling 1995, see Section 4.1). Assuming that *exact* floating point arithmetic operations can be performed in unit time, algorithms such as the Interior-Point (Karmarkar 1984) and Ellipsoid (Khachiyan 1980) methods for LP yield *weakly* polynomial bounds on the running time. The corresponding bounds are polynomial in n, k, and the number of bits B needed to represent the MDP. The dependence on B arises because these methods seek increasingly better *approximations* of the optimal solution, and this quest naturally gets curtailed by machine precision.

A contrasting approach, which eliminates the dependence on B, involves searching over a discrete set of solutions that is known to contain the optimal one. Thus, by applying the Simplex method (Dantzig 1963) on the linear program resulting from an MDP, bounds can be obtained solely in terms of n and k. While bounds for the canonical Simplex algorithm are in general exponential in n, it has been shown that the method is strongly polynomial for deterministic MDPs (Post and Ye 2013). The best strong bounds for LP are *subexponential* ones arising from randomised vertex-switching methods (Kalai 1992; Matoušek, Sharir, and Welzl 1996): the best that apply to general MDPs are $O(\text{poly}(n, k) \exp(2\sqrt{n}))$ (Gärtner 2002).

2.2 Value Iteration (VI)

Although attractive in theory, LP-based methods seldom perform better in practice when compared to "dynamic programming" techniques that compute $V^\star$ and $\pi^\star$ based on an

[1] Thus, we need to solve a *planning* problem. In the related problem of *learning* (Sutton and Barto 1998), R and T are unknown, but can be queried for samples of rewards and next states.

iterated sequence. Starting with V^0, an arbitrary initial guess of $V^\star$, the Value Iteration (VI) algorithm (Szepesvári 2010, see Section 2.4) applies the following iteration. $\forall s \in S$,

$$V^{t+1}(s) \leftarrow \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^t(s') \right).$$

The "Bellman Optimality Operator" applied above is a contraction in the max-norm, and Banach's Fixed Point Theorem ensures that the sequence $\langle V^0, V^1, V^2, \dots \rangle$ converges to $V^\star$ (Szepesvári 2010, see Appendix A). The number of iterations to convergence is bounded by a polynomial in n, k, $1/(1-\gamma)$, and B (Littman, Dean, and Kaelbling 1995). Some relatively simple changes to the update operator and the sequence of state-updates can speed up VI in practice (Sutton and Barto 1998, see Section 9.4), but the inherent dependence on $1/(1-\gamma)$ implies unbounded growth as $\gamma \to 1$.

2.3 Policy Iteration (PI)

We now consider the basis for Policy Iteration (PI), which is the subject of this paper. For a given policy π, let $\mathbf{IS}(\pi)$ be the set of states s on which π is *not* greedy with respect to its own action value function: in other words,

$$\mathbf{IS}(\pi) \stackrel{\text{def}}{=} \left\{ s \in S : Q^\pi(s, \pi(s)) < \max_{a \in A} Q^\pi(s,a) \right\}.$$

PI (Howard 1960) is an elegant approach based on the observation that

1. If $\mathbf{IS}(\pi) \neq \varnothing$, then π can be "improved" by switching to more promising actions on states in $\mathbf{IS}(\pi)$;
2. If $\mathbf{IS}(\pi) = \varnothing$, then π is optimal.

Thus, $\mathbf{IS}(\pi)$ is the set of (locally) "improvable states" in π. Specifically, for each state s in $\mathbf{IS}(\pi)$, let $\mathbf{IA}(\pi, s)$ be the set of actions that improve upon the action taken by π: that is,

$$\mathbf{IA}(\pi, s) \stackrel{\text{def}}{=} \{ a \in A : Q^\pi(s,a) > V^\pi(s) \}.$$

Now, assuming π is not optimal, let π' be a policy that in one or more states s in $\mathbf{IS}(\pi)$, takes an action belonging to $\mathbf{IA}(\pi, s)$, and in the remaining states, takes the same actions as π: that is,

$$\exists s \in S : \pi'(s) \in \mathbf{IA}(\pi, s), \text{ and}$$
$$\forall s \in S : (\pi'(s) = \pi(s)) \vee (\pi'(s) \in \mathbf{IA}(\pi, s)). \quad (3)$$

It can be shown that π' improves upon (or *dominates*) π in the following sense.

Definition 1 ($\succeq, \succ$)**.** *For functions $X : S \to \mathbb{R}$, $Y : S \to \mathbb{R}$, we define $X \succeq Y$ if and only if*

$$\forall s \in S : X(s) \geqslant Y(s),$$

and we define $X \succ Y$ if and only if

$$X \succeq Y \text{ and } \exists s \in S : X(s) > Y(s).$$

For policies $\pi_1, \pi_2 \in \Pi$, we define $\pi_1 \succeq \pi_2$ if and only if

$$V^{\pi_1} \succeq V^{\pi_2},$$

and we define $\pi_1 \succ \pi_2$ if and only if

$$V^{\pi_1} \succ V^{\pi_2}.$$

Theorem 2 (Policy improvement (Szepesvári 2010; Bertsekas 2012))**.** *Let $\pi, \pi' \in \Pi$ be such that $\mathbf{IS}(\pi) \neq \varnothing$, and π' satisfies* (3)*. Then*

$$\pi' \succ \pi.$$

Proof. For convenience, we provide a proof of this "policy improvement theorem", which is standard material in textbooks on MDPs (Szepesvári 2010; Bertsekas 2012). The main tool employed in the proof is the "Bellman Operator" $B^\pi : (S \to \mathbb{R}) \to (S \to \mathbb{R})$, given by: $\forall s \in S$,

$$(B^\pi(X))(s) \stackrel{\text{def}}{=} \sum_{s' \in S} \left(R(s, \pi(s)) + \gamma T(s, \pi(s), s') X(s') \right).$$

A moment's reflection assures us that π' satisfying (3) is equivalent to the following statement.

$$B^{\pi'}(V^\pi) \succ V^\pi. \quad (4)$$

Now, it is easy to verify that for $X : S \to \mathbb{R}$ and $Y : S \to \mathbb{R}$, if $X \succeq Y$, then $B^{\pi'}(X) \succeq B^{\pi'}(Y)$. Thus, applying $B^{\pi'}$ to both sides of (4), we get $(B^{\pi'})^2(V^\pi) \succeq B^{\pi'}(V^\pi)$. Applying $B^{\pi'}$ to both sides again and applying the transitivity of $\succeq$ *ad infinitum*, we obtain:

$$\lim_{l \to \infty} (B^{\pi'})^l (V^\pi) \succeq B^{\pi'}(V^\pi).$$

It is well known that by Banach's Fixed Point theorem , the limit on the left evaluates to $V^{\pi'}$. Combining the above relation with (4) completes the proof. □

Corollary 3 (Policy optimality)**.** *For $\pi \in \Pi$, if $\mathbf{IS}(\pi) = \varnothing$, then π is optimal.*

Proof. Again we argue using the Bellman Operator:

$$\begin{aligned} \mathbf{IS}(\pi) = \varnothing &\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \\ &\implies \forall \pi' \in \Pi : V^\pi \succeq \lim_{l \to \infty} (B^{\pi'})^l (V^\pi) \\ &\iff \forall \pi' \in \Pi : V^\pi \succeq V^{\pi'}. \end{aligned}$$

□

Indeed the PI algorithm essentially puts these results to practice: starting with some initial policy, it repeatedly applies policy improvement until an optimal policy is reached. Note that given a policy π, $\mathbf{IS}(\pi)$ and $\mathbf{IA}(\pi, \cdot)$ can be obtained in $O(n^3 + n^2 k)$ time. Thus, the complexity of PI is essentially determined by the number of iterations needed to reach an optimal policy. Since the algorithm can visit no policy twice (owing to strict improvement), the number of iterations is trivially bounded by k^n.

The elegant structure of PI facilitates complexity bounds that are independent of B and γ, thereby contributing to the method's theoretical appeal.[2] The key design choice within PI is in the implementation of the improvement operator. Theorem 2 holds for every choice of π' where some non-empty subset of $\mathbf{IS}(\pi)$ is "switched", and for each state s that is thereby chosen for switching, π' can adopt any from

[2] Polynomial bounds involving $1/(1-\gamma)$ have also been shown for PI (Ye 2011; Scherrer 2013).

action $\mathbf{IA}(\pi, s)$. *Which non-empty subset of* $\mathbf{IS}(\pi)$ *must we choose, and to which actions must we switch? Does our choice facilitate a tighter bound on the number of iterations?*

In the canonical variant of PI (called *Howard's PI*) (Howard 1960), *all* switchable actions are switched in every iteration: that is, the subset chosen for switching is $\mathbf{IS}(\pi)$ itself. Thus, Howard's PI implements a greedy variant of policy improvement, and is sometimes referred to as *Greedy PI*. Mansour and Singh (1999) show that regardless of the strategy used for picking improving actions, Howard's PI requires at most $13k^n/n$ iterations. Hollanders *et al.* (2014) tighten this bound to $(k/(k-1)+o(1))(k^n/n)$.

Mansour and Singh (1999) also propose a randomised version of PI in which the subset of states to be improved is chosen uniformly at random from among the non-empty subsets of $\mathbf{IS}(\pi)$. For this algorithm, they show that the expected number of iterations is bounded by $O\left(\left(\left(1+2/\log_2(k)\right)k/2\right)^n\right)$; again, the bound holds for every choice of action-switching rule. A tighter bound of $2^{0.78n}$ holds for $k = 2$. In this case, there is at most one improving action per state, and so no choice in switching.

Fearnley (2010) provides an MDP construction and an initial policy that takes Howard's PI through $\Omega(2^{n/7})$ iterations. While Fearnley's construction is for the total and average reward models, it has subsequently also been extended to the discounted reward model (Hollanders, Gerencsér, and Delvenne 2012). These recent lower bounds for Howard's PI follow several years after Melekopoglou and Condon (1994) showed a lower bound of $\Omega(2^{n/2})$ for *Simple PI*, a variant in which the improvable state with the highest index (assuming a fixed indexing over S) is chosen for improvement. This lower bound is shown on a two-action MDP.

Interestingly, we show a *positive* result based on Simple PI: faced with choice in action-switching (that is, when $k \geqslant 3$), we argue that it is efficient to choose an action uniformly at random from the improving set. To the best of our knowledge, the corresponding bound of $(2+\ln(k-1))^n$ expected iterations is the tightest complexity bound known for the PI family of algorithms when $k \geqslant 3$. Before describing this result in Section 4, we present an approach that can be coupled with *any* variant of PI to obtain an expected complexity bound of $O(\text{poly}(n,k)k^{n/2})$ for MDP planning.

3 Guess-and-Max

In this section, we describe Guess-and-Max, a simple guessing strategy that can be used to "quickly" find a policy that is already "close" to optimal. The idea underlying Guess-and-Max is that if PI is to get "stuck" in a long chain of improvement, then random guessing can help "jump" to policies higher up the chain. Unfortunately, the $\succ$ relation is a *partial order*, and does not offer a ready basis for comparing an arbitrary pair of policies. The central idea in Guess-and-Max is to construct a *total order* (denoted $\gg$) on the set of policies, and measure progress along the chain it induces.

Definition 4 (Total order $\gg$)**.** *For* $\pi \in \Pi$*, we define*

$$V(\pi) \stackrel{def}{=} \sum_{s \in S} V^{\pi}(s).$$

Let L *be an arbitrary total order on* Π*, which we shall use for tie-breaking. (For example,* L *could be taken as the lexicographic ordering of the set of all words of length* n *over the alphabet* A*, each word representing a policy.) For* $\pi_1, \pi_2 \in \Pi$*, we define* $\pi_1 \gg \pi_2$ *if and only if*

$$V(\pi_1) > V(\pi_2), \text{ or}$$
$$V(\pi_1) = V(\pi_2) \text{ and } \pi_1 L \pi_2.$$

Observe that since L is a total order, so is $\gg$: that is,

1. $\forall \pi_1, \pi_2 \in \Pi : \pi_1 \gg \pi_2$ and $\pi_2 \gg \pi_1 \implies \pi_1 = \pi_2$;
2. $\forall \pi_1, \pi_2, \pi_3 \in \Pi : \pi_1 \gg \pi_2$ and $\pi_2 \gg \pi_3 \implies \pi_1 \gg \pi_3$;
3. $\forall \pi_1, \pi_2 \in \Pi : \pi_1 \gg \pi_2$ or $\pi_2 \gg \pi_1$.

Since $\gg$ is a total order, we find it convenient to associate a unique index with each policy π to denote the number of policies it dominates under $\gg$:

$$I^{\pi}_{\gg} \stackrel{\text{def}}{=} \left|\{\pi' \in \Pi : \pi \gg \pi'\}\right|.$$

Clearly this index ranges from 1 to $|\Pi|$, with the highest index belonging to an optimal policy. Note that $\forall \pi, \pi' \in \Pi : I^{\pi}_{\gg} \geqslant I^{\pi'}_{\gg} \iff \pi \gg \pi'$. Hereafter, we refer to $I^{\pi}_{\gg}$ as the "$\gg$-index" of π.

Given any two policies, observe that they can be compared under $\gg$ with just two policy evaluations (in $O(n^3)$ time). More crucially, observe that $\gg$ is designed to be "respected" by $\succ$, and consequently by policy improvement.

Lemma 5 (Policy improvement respects $\gg$)**.** *For* $\pi, \pi' \in \Pi$*, if policy improvement to* π *yields* π'*, then* $\pi' \gg \pi$.

Proof. Since policy improvement to π yields π', it follows from Theorem 2 that $\pi' \succ \pi$. In turn, this implies that $V(\pi') > V(\pi)$, and hence $\pi' \gg \pi$. □

Having defined the total order $\gg$, we are now ready to apply it in Guess-and-Max, shown below. The procedure selects t policies uniformly at random from Π, and returns the one that dominates all $t-1$ others under $\gg$.[3] Running Guess-and-Max with $t = \lceil k^{n/2} \rceil$, we show that with high probability, the policy returned has a high $\gg$-index.

Procedure **Guess-and-Max**(t)
Select policy π uniformly at random from Π.
Repeat $t-1$ times:
  Select policy π' uniformly at random from Π.
  If $\pi' \gg \pi$ **then**
    $\pi \leftarrow \pi'$.
Return π.

Lemma 6. *Let* π_g *be the policy returned by Guess-and-Max*$\left(\lceil k^{n/2} \rceil\right)$*. For* $c \geqslant 1$*, we have:*

$$\mathbb{P}\left\{ I^{\gg}_{\pi_g} < k^n - ck^{n/2} \right\} < \frac{1}{e^c}.$$

[3] To select a policy uniformly at random from Π, one may simply visit each state and pick an action uniformly at random from A for that state. The complexity of this operation is clearly $O(nk)$.

Proof. Let $\Pi_{Top} \subseteq \Pi$ be the set of policies with indices at least $k^n - ck^{n/2}$: that is,

$$\Pi_{Top} \stackrel{\text{def}}{=} \left\{\pi \in \Pi : I_\pi^{\gg} \geqslant k^n - ck^{n/2}\right\}.$$

By construction, the policy π_g returned by Guess-and-Max must be in Π_{Top} if any one of the $\lceil k^{n/2} \rceil$ policies it has generated at random is in Π_{Top}. Therefore,

$$\mathbb{P}\left\{I_{\pi_g}^{\gg} < k^n - ck^{n/2}\right\} \leqslant \left(1 - \frac{|\Pi_{Top}|}{|\Pi|}\right)^{\lceil k^{n/2} \rceil}$$
$$\leqslant \left(1 - \frac{c}{k^{n/2}}\right)^{k^{n/2}} < \frac{1}{e^c}. \quad \square$$

From Lemma 5, we see that if policy improvement to π yields π', then $I_{\pi'}^{\gg} > I_\pi^{\gg}$. Thus, if Guess-and-Max returns a policy with $\gg$-index at least $k^n - i$, then any PI algorithm that is subsequently applied will *at most* make i policy improvements ($i + 1$ policy evaluations) to reach an optimal policy. We use this observation to bound the total number of policy evaluations performed by sequencing Guess-and-Max($\lceil k^{n/2} \rceil$) and PI. Here is our first main result.

Theorem 7 (Guess-and-Max bound)**.** *Consider an algorithm that runs Guess-and-Max*$\left(\lceil k^{n/2} \rceil\right)$*, takes the returned policy* π^g *as an initial policy for PI, and then runs PI until an optimal policy is found. The expected number of policy evaluations performed by the algorithm is* $O\left(k^{n/2}\right)$.

Proof. Since the number of policy evaluations under Guess-and-Max is $O\left(k^{n/2}\right)$, it suffices to show that the expected number of policy improvements in the PI phase is also $O\left(k^{n/2}\right)$. We bound this quantity below, after (1) observing that for $c \geqslant 0$, if $I_{\pi^g}^{\gg} \geqslant k^n - ck^{n/2}$, then the number of policy improvements is at most $ck^{n/2}$; and (2) by applying Lemma 6,

$$\sum_{c=1}^{\lceil k^{n/2} \rceil} \mathbb{P}\{k^n - (c-1)k^{n/2} > I_{\pi_g}^{\gg} \geqslant k^n - ck^{n/2}\} \cdot ck^{n/2}$$
$$\leqslant k^{n/2} + \sum_{c=2}^{\lceil k^{n/2} \rceil} \mathbb{P}\{I_{\pi_g}^{\gg} < k^n - (c-1)k^{n/2}\} \cdot ck^{n/2}$$
$$\leqslant k^{n/2} + \sum_{c=2}^{\infty} \frac{ck^{n/2}}{e^c} < 2k^{n/2}. \quad \square$$

In deriving the $O(k^{n/2})$ bound, we have accounted for the worst case of policy improvement—that PI will only improve the $\gg$-index by one every iteration. On the contrary, PI algorithms invariably take much larger leaps up the total order, and so can be initialised with fewer than $\lceil k^{n/2} \rceil$ guesses. In fact the randomised variant of Mansour and Singh (1999) *provably* takes a large number of large leaps, and so can possibly be combined with Guess-and-Max to get a tighter bound than $O(k^{n/2})$.

A second remark pertains to our choice of $V(\cdot)$ in the definition of $\gg$. As the reader may observe, any convex combination of the individual state values (rather equal weighting) will preserve the properties of $V(\cdot)$ we require here. At the moment, we do not know if the total order is strictly necessary for constructing a routine akin to Guess-and-Max. Yet, it does give useful intuition and it simplifies our analysis.

4 Randomised Simple PI

As briefly described in Section 2, Simple PI (Melekopoglou and Condon 1994) is a variant of PI in which exactly one state s^+ is improved in each iteration: assuming that the states are uniquely indexed, s^+ is the improvable state with the highest index. Let us assume that $S = \{s_1, s_2, \ldots, s_n\}$. If $\mathbf{IS}(\pi) \neq \varnothing$, then $s^+ = s_{i^+}$, where

$$i^+ = \max_{i \in \{1,2,\ldots,n\}, s_i \in \mathbf{IS}(\pi)} i.$$

Melekopoglou and Condon (1994) show a two-action MDP and a starting policy from which Simple PI must take at least $\Omega(2^{n/2})$ iterations. We show that the structure of Simple PI can, in fact, be used to an *advantage* when there are more than two actions: indeed switching to an action a^+ that is picked uniformly at random from $\mathbf{IA}(\pi, s^+)$ results in a favourable bound. This choice contrasts with the common practice of switching to actions that maximise $Q^\pi(s^+, \cdot)$. We are unaware of any theoretical analysis of this greedy improvement strategy. Below is a full description of Randomised Simple PI (RSPI), which is our randomised variant of Simple PI.

Algorithm **RSPI**
$\pi \leftarrow$ Arbitrary policy in Π.
Repeat
  Evaluate π; derive $\mathbf{IS}(\pi)$, $\mathbf{IA}(\pi, \cdot)$.
  If $\mathbf{IS}(\pi) \neq \varnothing$
    $i^+ \leftarrow \max_{i \in \{1,2,\ldots,n\}, s_i \in \mathbf{IS}(\pi)} i$.
    $a^+ \leftarrow$ Element of $\mathbf{IA}(\pi, s_{i^+})$ chosen uniformly at random.
    For $i \in \{1, 2, \ldots, n\}$:
      If $i = i^+$ **then**
        $\pi'(s_i) \leftarrow a^+$
      Else
        $\pi'(s_i) \leftarrow \pi(s_i)$.
    $\pi \leftarrow \pi'$.
Until $\mathbf{IS}(\pi) = \varnothing$.
Return π.

The structure of Simple PI facilitates an interpretation wherein an n-state MDP is solved by *recursively* solving several $(n-1)$-state MDPs. Imagine that we start with a policy π, in which action a is taken from state s_1. By construction, the algorithm ensures that a will get switched (if at all) only when a policy π' is reached such that $\mathbf{IS}(\pi') = \{s_1\}$. At this point, the "remaining MDP" (over states $s_2, s_3, \ldots, s_n$) has effectively been "solved" such that π' dominates (in terms of $\succeq$) every other policy π'' for which $\pi''(s_1) = a$.

Having thereby solved for the setting "a at s_1", the algorithm picks a provably-improving action b at s_1, solves for this new setting, and proceeds in this manner until an optimal action is picked at s_1. By picking uniformly at random among the improving actions at s_1, RSPI ensures that the expected number of switches to reach an optimal action at s_1 is $O(\log(k))$. Recursion leads to a bound of $O((\log(k))^n)$ on the expected number of policy evaluations performed by RSPI. Below we formalise this argument.

Definition 8 (a-optimal policy)**.** *For $a \in A$, we define a policy $\pi \in \Pi$ to be a-optimal if and only if*

$$\pi(s_1) = a \text{ and } \forall \pi \in \Pi : \pi'(s_1) = a \implies \pi \succeq \pi'.$$

Must there be an a-optimal policy for every action $a \in A$? Clearly, if an optimal policy $\pi^\star$ is such that $\pi^\star(s_1) = a$, then $\pi^\star$ must be an a-optimal policy. On the other hand, consider non-optimal policies that pick action a at s_1. The following lemma provides a necessary and sufficient condition for such policies to be a-optimal. Indeed it is evident from the lemma that for every $a \in A$, there must exist an a-optimal policy.

Lemma 9. *$\forall a \in A$, let $\pi \in \Pi$ be a policy such that $\pi(s_1) = a$ and π is not optimal. Then π is an a-optimal policy if and only if $\mathbf{IS}(\pi) = \{s_1\}$.*

Proof. Assume that π is an a-optimal policy, and yet there is some state $s_i \in \mathbf{IS}(\pi)$ for $2 \leqslant i \leqslant n$. Clearly, switching to an improving action in s_i must result in a policy that still takes action a in s_1, and by Theorem 2, improves upon π in terms of $\succ$. Thus, by Definition 8, π cannot be an a-optimal policy, and so our premise is contradicted.

Next we show that if $\mathbf{IS}(\pi) = \{s_1\}$, then π is a-optimal. Note that if $\mathbf{IS}(\pi) = \{s_1\}$, then for every policy $\pi' \in \Pi$ such that $\pi'(s_1) = a$, we have that for $2 \leqslant i \leqslant n$, $Q^\pi(s_i, \pi'(s_i)) \leqslant V^\pi(s_i)$. Therefore, $V^\pi \succeq B^{\pi'}(V^\pi)$. As in the proof of Theorem 2, we now apply $B^{\pi'}$ repeatedly to both sides, to get that $V^\pi \succeq V^{\pi'}$. Thus, π is a-optimal. □

From Lemma 9 and the construction of Simple PI, it is immediate that the trajectory taken by the algorithm has the following structure.

Lemma 10. *Let the sequence of policies visited by Simple PI on a run be $\pi_1, \pi_2, \ldots, \pi_T$, where π_T is an optimal policy. For every $a \in A$, if there is a policy π_i in this sequence such that $\pi_i(s_1) = a$, then there exists $j \in \{i, i+1, \ldots, T\}$ such that π_j is an a-optimal policy, and for all $i \leqslant t \leqslant j$, $\pi_t(s_1) = a$.*

The crucial result underpinning our analysis is that policies that are optimal with respect to different actions at s_1 are themselves comparable (under $\succeq$), and further, their relative order is encoded in their sets of improving actions. As a consequence, by merely evaluating an a-optimal policy $\pi^\star_a$, $a \in A$, Simple PI "knows" the *complete* set of actions $b \in A$ such that a b-optimal policy will strictly dominate $\pi^\star_a$.

Lemma 11. *For $a, b \in A$, let $\pi^\star_a$ be an a-optimal policy, and $\pi^\star_b$ be a b-optimal policy. (1) If $b \in \mathbf{IA}(\pi^\star_a, s_1)$, then $\pi^\star_b \succ \pi^\star_a$. (2) If $b \notin \mathbf{IA}(\pi^\star_a, s_1)$, then $\pi^\star_a \succeq \pi^\star_b$.*

Proof. (1) If $b \in \mathbf{IA}(\pi^\star_a, s_1)$, let policy $\pi_b \in \Pi$ take the same actions as $\pi^\star_a$ in states $s_2, s_3, \ldots, s_n$, but $\pi_b(s_1) = b$. We see that $B^{\pi_b}(V^{\pi^\star_a}) \succ V^{\pi^\star_a}$; the repeated application of B^{π^b} to both sides yields $V^{\pi_b} \succ V^{\pi^\star_a}$. Since $V^{\pi^\star_b} \succeq V^{\pi_b}$, $\pi^\star_b \succ \pi^\star_a$. (2) If $b \notin \mathbf{IA}(\pi^\star_a, s_1)$, then $Q^{\pi^\star_a}(s_1, b) \leqslant V^{\pi^\star_a}(s_1)$. Since from Lemma 9, we have that $\mathbf{IS}(\pi^\star_a) = \{s_1\}$, we also have that for $i \in \{2, 3, \ldots, n\}$, $Q^{\pi^\star_a}(s_i, \pi^\star_b(s_i)) \leqslant V^{\pi^\star_a}(s_i)$. Consequently, $V^{\pi^\star_a} \succeq B^{\pi^\star_b}(V^{\pi^\star_a})$. Through the repeated application of $B^{\pi^\star_b}$ to both sides, we get $V^{\pi^\star_a} \succeq V^{\pi^\star_b}$. □

We are now ready to prove our final result.

Theorem 12 (RSPI bound)**.** *The expected number of policy evaluations performed by RSPI is at most $(2 + \ln(k-1))^n$.*

Proof. If we track the action taken at s_1 along the trajectory of policies visited by Simple PI, lemmas 10 and 11 imply that the sequence must be of the form $a_1^{t_1} a_2^{t_2} \ldots a_k^{t_k}$, where for $i \in \{1, 2, \ldots, k\}$, a_i is some action in A and $t_i \geqslant 0$. Moreover, for $i, j \in \{1, 2, \ldots, k\}, i < j$, if π is an a_i-optimal policy and π' is an a_j-optimal policy, then $\pi' \succeq \pi$. Lemma 9 also informs us that the last occurrence of action a in this sequence must belong to an a-optimal policy.

Up to now, we have not made use of the fact that RSPI uses randomised action selection. Observe that the consequence of so doing is that from a_i, the trajectory can "jump ahead" to a_j, skipping all intermediate actions: that is, $t_{i+1} = t_{i+2} = \cdots = t_{j-1} = 0$. In fact, observe that under RSPI, a_j is picked uniformly at random from the set $\{a_l, a_{l+1}, \ldots, a_k\}$ where l is the least number greater than j such that if $\pi^\star_{a_l}$ is an a_l-optimal policy and $\pi^\star_{a_j}$ is an a_j-optimal policy, then $\pi^\star_{a_l} \succ \pi^\star_{a_j}$. For $i \in \{1, 2, \ldots, k\}$, let τ_i be the expected number of actions visited at s_1 *subsequent* to visiting action a_i. We get $\tau_k = 0$, and for $i \in \{1, 2, \ldots, k-1\}$:

$$\tau_i \leqslant 1 + \frac{1}{k-i} \sum_{i'=i+1}^{k} \tau_{i'}.$$

From this relation, it can be verified for $1 \leqslant i \leqslant (k-1)$ that τ_i is upper-bounded by the $(k-i)^{th}$ harmonic number $H_{k-i} = \sum_{j=1}^{k-i}(1/j)$. In the worst case, we start at a_1, and take at most H_{k-1} steps in expectation to reach a_k. Thus the expected number of actions visited at s_1 by RSPI is bounded by $1 + H_{k-1} \leqslant 2 + \ln(k-1)$.

The bound in the claim is trivial for every 1-state MDP. If for every $(n-1)$-state MDP, $n \geqslant 2$, the expected number of policy evaluations is upper-bounded by B, then the argument above implies an upper bound of $(2 + \ln(k-1))B$ for every n-state MDP. Our proof is done. □

5 Conclusion

In this paper, we consider PI, an elegant algorithm for MDP planning. For an n-state, k-action MDP, we propose a novel scheme for initialising PI, which leads to an overall bound of $O(k^{n/2})$ policy evaluations in expectation. We also analyse a randomised version of Simple PI (RSPI), which yields

a bound of $(2 + \ln(k-1))^n$ policy evaluations in expectation. After decades of research that have improved bounds for the PI family, often by polynomial factors and constants, the gains made by RSPI for $k \geqslant 3$ mark a key theoretical milestone.

Acknowledgements

SK, NM, and AG were partially supported by DST INSPIRE Faculty grants IFA13-ENG-65, IFA12-ENG-31, and IFA13-ENG-69, respectively. A substantial portion of this work was carried out when SK and NM were visiting the Indian Institute of Science as DST INSPIRE Faculty Fellows The authors are grateful to Supratik Chakraborty, Romain Hollanders, and anonymous reviewers for providing useful comments.

References

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press, 1^{st} edition.

Bertsekas, D. P. 2012. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 4^{th} edition.

Dantzig, G. B. 1963. *Linear Programming and Extensions*. Princeton University Press.

Fearnley, J. 2010. Exponential lower bounds for policy iteration. In *Proceedings of the Thirty-seventh International Colloquium on Automata, Languages and Programming (ICALP 2010)*, 551–562. Springer.

Gärtner, B. 2002. The random-facet simplex algorithm on combinatorial cubes. *Random Structures and Algorithms* 20(3):353–381.

Hollanders, R.; Gerencsér, B.; Delvenne, J.; and Jungers, R. M. 2014. Improved bound on the worst case complexity of policy iteration. URL: http://arxiv.org/pdf/1410.7583v1.pdf.

Hollanders, R.; Gerencsér, B.; and Delvenne, J. 2012. The complexity of policy iteration is exponential for discounted Markov decision processes. In *Proceedings of the Fifty-first IEEE Conference on Decision and Control (CDC 2012)*, 5997–6002. IEEE.

Howard, R. A. 1960. *Dynamic programming and Markov processes*. MIT Press.

Kalai, G. 1992. A subexponential randomized simplex algorithm. In *Proceedings of the twenty-fourth annual ACM Symposium on Theory of Computing (STOC 1992)*, 475–482. ACM.

Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica* 4(4):373–396.

Khachiyan, L. G. 1980. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics* 20(1):53–72.

Littman, M. L.; Dean, T. L.; and Kaelbling, L. P. 1995. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, 394–402. Morgan Kauffman.

Mahadevan, S. 1996. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning* 22(1–3):159–195.

Mansour, Y., and Singh, S. 1999. On the complexity of policy iteration. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 1999)*, 401–408. Morgan Kaufmann.

Matoušek, J.; Sharir, M.; and Welzl, E. 1996. A subexponential bound for linear programming. *Algorithmica* 16(4/5):498–516.

Melekopoglou, M., and Condon, A. 1994. On the complexity of the policy improvement algorithm for Markov decision processes. *INFORMS Journal on Computing* 6(2):188–192.

Post, I., and Ye, Y. 2013. The simplex method is strongly polynomial for deterministic Markov decision processes. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*, 1465–1473. Morgan Kauffman.

Puterman, M. L. 1994. *Markov Decision Processes*. Wiley.

Scherrer, B. 2013. Improved and generalized upper bounds on the complexity of policy iteration. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 386–394. Curran Associates.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Szepesvári, C. 2010. *Algorithms for Reinforcement Learning*. Morgan & Claypool.

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.