Batch-Switching Policy Iteration

Shivaram Kalyanakrishnan, Utkarsh Mall, Ritish Goyal
Department of Computer Science and Engineering
Indian Institute of Technology Bombay, Mumbai 400076 India
{shivaram, utkarshmall13, ritish}@cse.iitb.ac.in

Abstract

Policy Iteration (PI) is a widely-used family of algorithms for computing an optimal policy for a given Markov Decision Problem (MDP). Starting with an arbitrary initial policy, PI repeatedly updates to a dominating policy until an optimal policy is found. The update step involves switching the actions corresponding to a set of "improvable" states, which are easily identified. Whereas progress is guaranteed even if just one improvable state is switched at every step, the canonical variant of PI, attributed to Howard [1960], switches every improvable state in order to obtain the next iterate. For MDPs with n states and 2 actions per state, the tightest known bound on the complexity of Howard's PI is $O(2^n/n)$ iterations. To date, the tightest bound known across all variants of PI is $O(1.7172^n)$ expected iterations for a randomised variant introduced by Mansour and Singh [1999].

We introduce Batch-Switching Policy Iteration (BSPI), a family of deterministic PI algorithms that switches states in "batches", taking the batch size b as a parameter. By varying b, BSPI interpolates between Howard's PI and another previously-studied variant called Simple PI [Melekopoglou and Condon, 1994]. Our main contribution is a bound of $O(1.6479^n)$ on the number of iterations taken by an instance of BSPI. We believe this is the tightest bound shown yet for any variant of PI. We also present experimental results that suggest Howard's PI might itself enjoy an even tighter bound.

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.[1] An MDP can be used to model an agent whose actions result in stochastic state transitions, while yielding associated rewards. The agent's natural aim is to consistently take actions so as to maximise its *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 is finite, with $|S| = n \geq 2$. We also restrict our attention to 2-action MDPs: that is, we assume $|A| = 2$. As we shall see, this simplified setting itself poses significant challenges for theoretical research to overcome.

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 sum of discounted rewards, for some given discount factor $\gamma \in [0, 1)$. Concretely, consider an agent that starts in some state s_0 at time 0 and continuously 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 defined to be

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

Let Π be the set of distinct policies corresponding to M. It is a key property, as established by Bellman [1957], that this set contains a policy $\pi^\star$ such that for $\forall s \in S, \forall \pi \in \Pi$,

$$V^{\pi^\star}(s) \geq V^\pi(s). \quad (2)$$

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

The problem we consider is precisely that of finding an optimal policy for a given MDP $M = (S, A, R, T, \gamma)$. Specifically, we examine the Policy Iteration (PI) family of algorithms [Howard, 1960], which follow the general template of starting with some initial policy, and repeatedly performing locally improving "switches" until an optimal policy is found. Each non-optimal policy is guaranteed to have a non-empty set of "improvable" states, which are easy to identify.

[1] Sections 1 and 2, which provide background material, reproduce some portions of the text from an earlier publication of the authors [Kalyanakrishnan *et al.*, 2016].

A switch corresponds to changing the actions taken by the current policy on one or more of these improvable states. Algorithms within the PI family are indeed differentiated solely by the rule they apply to pick states for switching. Most popular is Howard's PI [Howard, 1960], which implements a greedy switching rule, wherein every state in the improvable set is switched. Whereas this rule is deterministic, variants of PI with randomised switching rules have also been considered in the literature [Mansour and Singh, 1999].

While PI tends to be extremely efficient on MDPs typically encountered in practice, it has been surprisingly difficult to establish tight theoretical upper bounds on its running time. The tightest known bound on the number of iterations taken by Howard's PI is $O(2^n/n)$ [Mansour and Singh, 1999], which improves upon the trivial bound of 2^n by only a linear factor! A bound of $O(1.7172^n)$ expected iterations applies to a randomised variant of PI that was proposed by Mansour and Singh [1999]: this is the tightest bound known to date for the PI family. Against this backdrop, we propose a new family of deterministic PI algorithms called **Batch-Switching Policy Iteration** (**BSPI**). Algorithms in this family are parameterised by a batch size b that determines the switching rule. Interestingly the family includes both Howard's PI (BSPI with $b = n$) and a previously-studied algorithm called Simple PI [Melekopoglou and Condon, 1994] (BSPI with $b = 1$). By varying b, BSPI effects an interpolation between these seemingly disparate variants of PI.

Our first contribution is a bound of $O(1.6479^n)$ iterations for an instance of BSPI. This bound therefore becomes the tightest theoretically-proven bound for the PI family. Our analysis makes use of recent results from Gerencsér *et al.* [2015] on the complexity of Howard's PI on small, constant-size MDPs. The bound of $O(1.6479^n)$ iterations, which is achieved when b is set to 7, depends on the analysis of Howard's PI on 7-state MDPs. It appears that our bound will get even tighter if the complexity-analysis of MDPs with 8 or more states becomes available; as yet, the corresponding computation has not been feasible. Nevertheless, we present experimental results to suggest that larger values of b do perform better in practice, and so Howard's PI must remain the method of choice. We consider our experiments themselves a useful contribution, as they reinforces the case for a tighter analysis of Howard's PI. However, our more important contribution is the formulation of Howard's PI as a limiting case of BSPI, which opens up a new approach for analysis.

The remainder of the paper is organised as follows. In Section 2, we describe the PI family of algorithms. Next, in Section 3, we introduce "trajectory-bounding trees", a useful device to track the progress of Howard's PI. In fact the idea of trajectory-bounding trees motivates the BSPI family of algorithms, which we present in Section 4. This section includes theoretical, as well as experimental, analysis related to BSPI. We present concluding remarks in Section 5.

2 Policy Iteration

In this section, we first lay the groundwork for understanding Policy Iteration (PI)[Howard, 1960]. We then present the method and related complexity results.

Policy Evaluation. To begin, we observe from (1) that the *value function* V^π of a policy π, which yields values at each state, can be specified *recursively*: $\forall s \in S$,

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

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

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

$Q^\pi(s, a)$ denotes the expected long-term reward obtained when the agent executes action a at state s *once*, and *thereafter* executes π. We shall shortly see that the computation and comparison of action values lies at the heart of PI.

Policy Improvement. 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: formally,

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

Since we are only dealing with 2-action MDPs, we may equivalently define a state $s \in S$ to belong to $\mathbf{IS}(\pi)$ if and only if $Q^\pi(s, \pi(s)) < Q^\pi(s, \pi(s)^C)$, where for $a \in A$, a^C denotes the element of A other than a (thus, $\{a\} \cup \{a^C\} = A$). $\mathbf{IS}(\pi)$ is the set of (locally) "improvable" states in π, in the sense that taking a different action in any of these states for just one time step yields a strict increase in value. It turns out that persisting with switched actions for ever must continue to yield dominating values. Concretely, let π' be a policy that in one or more states in $\mathbf{IS}(\pi)$, switches the action taken by π, and in the remaining states, takes the same actions as π. In other words, let π' be such that

$$\begin{aligned} &\exists s \in S : (s \in \mathbf{IS}(\pi)) \wedge (\pi'(s) \neq \pi(s)), \text{ and} \\ &\forall s \in S : (s \notin \mathbf{IS}(\pi)) \implies (\pi'(s) = \pi(s)). \end{aligned} \tag{3}$$

Observe that π' is well-defined if only if $\mathbf{IS}(\pi)$ is non-empty. In turn, it can be shown that (1) $\mathbf{IS}(\pi)$ is empty if and only if π is optimal, and (2) if π is non-optimal, then π' *dominates* π in the following sense.

Definition 1 ($\succeq, \succ$)**.** *For functions $X : S \to \mathbb{R}$, $Y : S \to \mathbb{R}$, (1) we define $X \succeq Y$ if and only if $\forall s \in S : X(s) \geq Y(s)$, and (2) we define $X \succ Y$ if and only if $X \succeq Y$ and $\exists s \in S : X(s) > Y(s)$. For policies $\pi_1, \pi_2 \in \Pi$, (1) we define $\pi_1 \succeq \pi_2$ if and only if $V^{\pi_1} \succeq V^{\pi_2}$, and (2) we define $\pi_1 \succ \pi_2$ if and only if $V^{\pi_1} \succ V^{\pi_2}$.*

Theorem 2 (Policy improvement [Bertsekas, 2012])**.** *Let $\pi, \pi' \in \Pi$ be such that $\mathbf{IS}(\pi) \neq \emptyset$, and π' satisfies* (3)*. Then $\pi' \succ \pi$.*

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

The theorem and corollary are standard material in textbooks on MDPs [Bertsekas, 2012], and so we omit proofs. The interested reader might find the proofs provided in our recent paper [Kalyanakrishnan *et al.*, 2016, see Theorem 2 and Corollary 3] especially convenient, as they use the same notation as this paper. The main tool employed in the proofs is the "Bellman Operator" $B^\pi : (S \to \mathbb{R}) \to (S \to \mathbb{R})$. For $X : S \to \mathbb{R}$, $\forall s \in S$,

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

In particular, the proofs use the fact that if $B^\pi(X) \succeq X$, then $\lim_{l \to \infty}(B^\pi)^l(X) = V^\pi \succeq X$.

The idea behind PI is to put Theorem 2 to practice. Starting with some initial policy, PI repeatedly applies policy improvement until an optimal policy is reached. For a given policy π, it takes at most $O(n^3)$ time to compute $\mathbf{IS}(\pi)$. Thus, the complexity of PI is dictated 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 upper-bounded by $|\Pi| = 2^n$.

Complexity of Policy Iteration. The key design choice within PI is in the implementation of the switching rule. Theorem 2 holds for *every* choice of π' where some non-empty subset of $\mathbf{IS}(\pi)$ is "switched". In the canonical variant of PI (called *Howard's PI* [Howard, 1960] or *Greedy PI*), *every* improvable state is switched in every iteration; that is, the subset chosen for switching is $\mathbf{IS}(\pi)$ itself. Mansour and Singh [1999] show that Howard's PI requires at most $6(2^n/n)$ iterations (for *every* MDP and *every* initial policy). This bound was subsequently tightened by Hollanders *et al.* [2014] to $(2+o(1))(2^n/n)$. 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 $\mathbf{IS}(\pi)$. They show that the expected number of iterations of this algorithm (for *every* MDP and *every* initial policy) is $O(1.7172^n)$. The analysis of both algorithms is based on the argument that (1) there exist only a "small" number of policies with "small" improvement sets; and (2) improving any policy with a "large" improvement set must exclude several other policies from ever being visited.

Hollanders *et al.* [2012], adapting a construction by Fearnley [2010], furnish an MDP and an initial policy that take Howard's PI through $\Omega(2^{n/7})$ iterations. However, it must be noted that this MDP has as many as $\Omega(n)$ actions in some states. The tightest lower bound for Howard's PI on 2-action MDPs is only $\Omega(n)$ [Hansen and Zwick, 2010]; it remains an open question whether the algorithm can indeed take an exponential number of iterations on such MDPs. Note that a lower bound of $\Omega(2^{n/2})$ iterations has been shown on 2-action MDPs for *Simple PI*, a variant of PI in which the improvable state with the highest index (assuming a fixed indexing over S) is switched [Melekopoglou and Condon, 1994].

PI is especially appealing from a theoretical standpoint because it facilitates "strong" running-time bounds for MDP planning: that is, bounds solely in terms of n. Value Iteration, another popular approach, enjoys a polynomial dependence on n, but at the cost of a dependence on the discount factor γ and the quality of the desired approximation [Littman *et al.*, 1995].[2] Linear Programming offers yet another alternative for deriving strong bounds. While bounds for the canonical Simplex algorithm are exponential in n, the method is strongly polynomial for deterministic MDPs [Post and Ye, 2013]. The best strong bounds for MDP planning are from a randomised vertex-switching method [Gärtner, 2002], and of the form $O(\text{poly}(n)\exp(2\sqrt{n}))$ (expected). The tightest worst-case bound is obtained by formulating a "unique sink orientation of a cube" based on the set of policies, and applying a deterministic algorithm that runs in $O(\text{poly}(n)1.61^n)$ time [Szabó and Welzl, 2001].

3 Trajectory-Bounding Trees

In this section and the next, we present the core theoretical contributions of the paper. We shall assume that the states are named such that $S = \{s_1, s_2, \ldots, s_n\}$, and the two actions are the bits **0** and **1** (thus, $A = \{\mathbf{0}, \mathbf{1}\}$). We also find it convenient to denote policies by n-length bit-strings, each bit representing the action taken at the state with the bit's position as its index. For example, **011** will be a policy for a 3-state MDP that takes action **0** at state s_1, action **1** at state s_2, and action **1** at state s_3.

We begin by considering a run of Howard's PI on any given 2-state MDP. Since there are only 4 possible policies, any run can take at most 4 iterations (one iteration is one policy evaluation). However, a careful look informs us that in fact no more than 3 iterations will ever be performed.

Proposition 4 (Motivating example)**.** *On every 2-state MDP, Howard's PI terminates in at most 3 iterations.*

Proof. Without loss of generality, assume that the initial policy is $\pi_1 = \mathbf{00}$. If π_1 is already optimal, we are done. If it is not optimal, then by Corollary 3, $\mathbf{IS}(\pi_1)$ must be non-empty.

If $\mathbf{IS}(\pi_1) = \{s_1\}$, then $V^{\pi_1}(s_2) \geq Q^{\pi_1}(s_2, \mathbf{1})$, or equivalently, $V^{\pi_1} \succeq B^{\mathbf{01}}(V^{\pi_1})$. By repeatedly applying the Bellman operator (as done in the proof of Theorem 2), we conclude that $\pi_1 \succeq \mathbf{01}$. Thus, PI cannot encounter policy **01** during this run, and must therefore terminate after at most 3 iterations. A similar argument applies if $\mathbf{IS}(\pi_1) = \{s_2\}$, which would imply that $\pi_1 \succeq \mathbf{10}$.

If $\mathbf{IS}(\pi_1) = \{s_1, s_2\}$, then Howard's PI will update to a new policy $\pi_2 = \mathbf{11}$. Again, if π_2 is optimal, our run has completed in 2 iterations; if not, then either $\mathbf{IS}(\pi_2) = \{s_1\}$ or $\mathbf{IS}(\pi_2) = \{s_2\}$ (it is not possible that $\mathbf{IS}(\pi_2) = \{s_1, s_2\}$, as Theorem 2 would then imply $\pi_1 \succ \pi_2$). If $\mathbf{IS}(\pi_2) = \{s_1\}$, then Howard's PI will update to $\pi_3 = \mathbf{10}$. s_1 cannot be in the improvement set of π_3, as that would imply $\mathbf{00} \succ \pi_3$; s_2 cannot be in the improvement set of π_3, as that would imply $\mathbf{11} \succ \pi_3$. Thus π_3 must be optimal. A similar argument applies if $\mathbf{IS}(\pi_2) = \{s_2\}$. □

The argument we have illustrated with $n = 2$ can be generalised as follows. Given a policy $\pi \in \Pi$ and a subset of states $IS \subseteq S$, we consider the event that IS is indeed the

[2] Polynomial bounds involving $1/(1-\gamma)$ have also been shown for Howard's PI [Ye, 2011; Hansen *et al.*, 2013; Scherrer, 2013].

improvement set of π. If so, Theorem 2 enables us to infer a set of policies, $L^+_{\pi,IS}$, that must provably dominate π:

$$L^+_{\pi,IS} \stackrel{\text{def}}{=} \{\pi' \in \Pi : \exists s \in IS(\pi'(s) \neq \pi(s)) \wedge \forall s \in S \setminus IS(\pi'(s) = \pi(s))\}.$$

A symmetric argument (as illustrated in Proposition 4) identifies another set of policies, $L^-_{\pi,IS}$, that π must provably dominate or equal in value (note that $L^-_{\pi,IS}$ contains π, too):

$$L^-_{\pi,IS} \stackrel{\text{def}}{=} \{\pi' \in \Pi : \forall s \in IS(\pi'(s) = \pi(s))\}.$$

The pair of π and IS can possibly be encountered by PI only if $L^+_{\pi,IS}$ does not contain any policies that can already be guaranteed to be dominated or equaled in value by π. Since PI makes strict progress with each iteration, we can infer that trajectories taken by the method are constrained as follows.

Proposition 5 (Constraint on trajectories taken by PI)**.** *If $(\pi_1, IS_1), (\pi_2, IS_2), \ldots, (\pi_t, IS_t)$ is a trajectory encountered by PI, it must satisfy, for $1 \leq i < j \leq t$:*

$$L^-_{\pi_i,IS_i} \cap L^+_{\pi_j,IS_j} = \emptyset. \quad (4)$$

The effect of (4) in constraining the path taken by policy iteration can be visualised in the form of a "trajectory-bounding tree" (TBT), shown in Figure 1 for $n = 2$ (top row) and $n = 3$ (middle and bottom rows).

- Each node in a TBT corresponds to a policy along with an improvement set; one tree is grown with each possible improvement set occurring at the root.
- Nodes with empty improvement sets are leaves.
- Every child contains the policy obtained by switching the parent's policy appropriately—in this paper, we only consider greedy switching (as in Howard's PI).
- Together the children of a node contain every improvement set that does not violate (4).

Clearly, every trajectory taken by Howard's PI must be a path from the root to some leaf in the corresponding TBT. However, we cannot be certain that there exist MDPs and starting policies to instantiate every possible path in the TBTs. In this sense, the trees only "upper-bound" the trajectories taken by Howard's PI. We conclude from Figure 1 that on 3-state MDPs, Howard's PI can take at most 5 iterations. We obtain bounds for higher values of n by implementing a procedure based on the logic described above. The corresponding bounds for $n = 4, 5, 6$ are $8, 13, 21$, respectively. While the complexity of TBTs for $n = 7$ makes it infeasible for the authors to compute the tree depths, we can infer a bound of 33 iterations from results published by Gerencsér *et al.* [2015].

Gerencsér *et al.* [2015] undertake the computation of the size of "order-regular" (OR) matrices, whose rows satisfy constraints imposed by acyclic unique sink orientations of cubes [Szabó and Welzl, 2001]. Although the authors of the present paper conceived the idea of TBTs independently, they have subsequently realised that TBTs implementing greedy switching bear a one-to-one correspondence with OR matrices. Essentially, (4), if coupled with greedy policy improvement, can be shown to be equivalent to the constraint

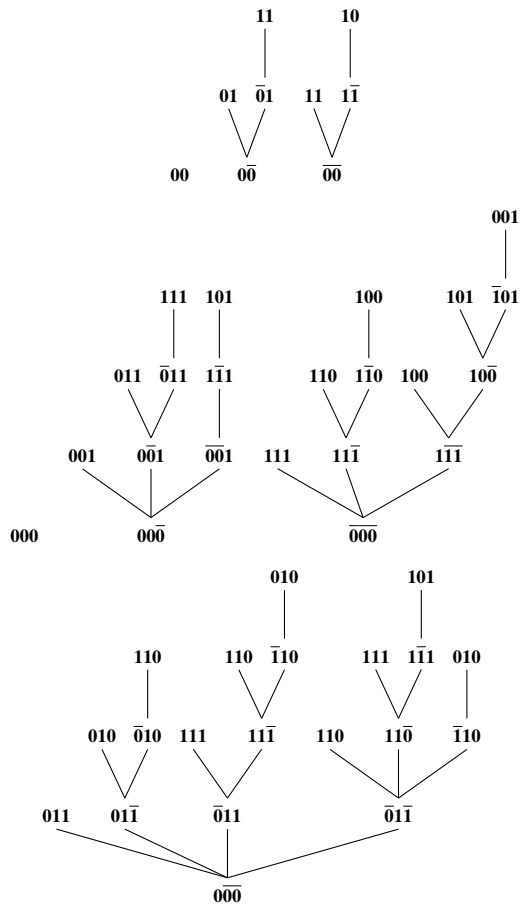


Figure 1: The set of trajectory-bounding trees (TBTs) for 2-state MDPs (top row) and 3-state MDPs (middle and bottom rows). Each node corresponds to a policy and an improvement set. States in the improvement set have a line drawn above the corresponding action. For every set of paths that differ solely in the naming of states and actions, only one is shown (for example: **01**, **10**, and **11** are not shown because **00** is; $\overline{\mathbf{00}}$—$\overline{\mathbf{1}}\mathbf{1}$—**01** is not shown because $\overline{\mathbf{00}}$—$\mathbf{1}\overline{\mathbf{1}}$—**10** is).

on OR matrices [Hansen, 2012, see Lemma 3.3.1]. While it exceeds the scope of this paper to demonstrate the equivalence, the authors willingly attribute the ideas underlying TBTs to work that predates this paper [Hansen, 2012; Gerencsér *et al.*, 2015]. Interestingly, previous work, too, has not been able to furnish tight bounds for $n \geq 8$.

4 Batch-Switching Policy Iteration

Although TBTs help bound the complexity of Howard's PI on small, constant-size MDPs, they do not automatically yield improved bounds for general (n-state) MDPs. The key technical idea of this paper is to fashion a variant of PI that exploits available bounds for TBTs to derive tighter bounds for general MDPs. The first step in our approach is to partition the set of states into *batches*, and to implement the logic of TBTs "within each batch". Second, assuming a fixed indexing of batches, we restrict switching for policy improvement to the "active" batch with the *highest* index (much as in Sim-

ple PI [Melekopoglou and Condon, 1994]). This strategy induces a recursive structure for our analysis to exploit. Below we formalise these ideas and present the Batch-Switching Policy Iteration (BSPI) algorithm.

We assume that a batch size $b \in \{1, 2, \ldots, n\}$ is given as an input to BSPI, which partitions the set of states $S = \{s_1, s_2, \ldots, s_n\}$ into $m = \lceil n/b \rceil$ sets. Each set B_j in the partition, $j \in \{1, 2, \ldots m\}$, is of size b, possibly except B_m, which could be smaller:

$$B_j = \{s_{b(j-1)+1}, s_{b(j-1)+2}, \ldots, s_{\min\{bj,n\}}\}.$$

If $\pi \in \Pi$ is the policy currently being evaluated, and it is not optimal, BSPI considers the *highest*-indexed set in the partition that intersects $\mathbf{IS}(\pi)$, and switches all those states in the intersection. In other words, the iterate π' takes the same actions as π on all states except those in $B_j \cap \mathbf{IS}(\pi)$, where j is the largest element in $\{1, 2, \ldots m\}$ such that a non-empty intersection results. Below is a full description of BSPI. Observe that if b is set to 1, the algorithm is the same as Simple PI [Melekopoglou and Condon, 1994]. If run with $b = n$, BSPI is equivalent to Howard's PI.

Algorithm **BSPI**
Input parameter $b \in \{1, 2, \ldots, n\}$.
$\pi \leftarrow$ Arbitrary policy in Π.
Evaluate π; derive $\mathbf{IS}(\pi)$.
While $\mathbf{IS}(\pi) \neq \emptyset$
  $j \leftarrow \lceil n/b \rceil$.
  While $B_j \cap \mathbf{IS}(\pi) = \emptyset$
    $j \leftarrow j - 1$.
  For $i \in \{1, 2, \ldots, n\}$:
    If $s_i \in (B_j \cap \mathbf{IS}(\pi))$ **Then**
      $\pi'(s_i) \leftarrow \pi(s_i)^C$
    Else
      $\pi'(s_i) \leftarrow \pi(s_i)$.
  $\pi \leftarrow \pi'$.
  Evaluate π; derive $\mathbf{IS}(\pi)$.
Return π.

We proceed to provide a bound on the number of iterations taken by BSPI. For simplicity, our analysis assumes that n is a multiple of b (thus $m = n/b$). We shall denote by $\tau(b)$ the maximal number of nodes that can be encountered along any path from root to leaf (both inclusive) in a TBT corresponding to a b-state MDP. The concatenation of bit-strings w_1 and w_2 shall be denoted $w_1 w_2$.

In our view of policies as strings, a *prefix* of length l fixes the actions taken at $s_1, s_2, \ldots, s_l$. The following lemmas establish the centrality of prefixes in the analysis of BSPI. Concretely, fix $r \in \{1, 2, \ldots, m\}$, and fix x, an arbitrary bit-string of length $(r-1)b$, which the lemmas consider as a prefix.

Lemma 6 (Prefix contiguity)**.** *Let $\pi = xy$ and $\pi' = x'y'$ (with $|x'| = (r-1)b$) be policies visited consecutively by BSPI, such that $x' \neq x$. Then, for every policy $\pi'' = x''y''$ (with $|x''| = (r-1)b$) visited subsequently, $x'' \neq x$.*

Proof. Since π and π' have different $(r-1)b$-length prefixes, we infer from the construction of BSPI that there are no improvable states in the "y" portion of π. It follows that π dominates or equals in value every policy of the form $x\bar{y}$, and so, by Theorem 2, no such policy can be visited after π. □

In other words, the occurrences of a prefix along a trajectory taken by BSPI must be contiguous. As the key step in our analysis, we consider policies that have x as a prefix, followed by a middle b-length segment y, and a suffix z with *no* improvable states. We show that BSPI can visit no more than $\tau(b)$ such policies.

Lemma 7 (Sandwiched segments)**.** *BSPI can encounter at most $\tau(b)$ policies of the form $\pi = xyz$, where (1) y is a bit-string of length b, and (2) for all $j \in \{r+1, r+2, \ldots, m\}$: $\mathbf{IS}(\pi) \cap B_j = \emptyset$ (the "z" part of π has no improvable states).*

Proof. We prove the lemma by contradiction. Assume that BSPI encounters at least $\tau(b) + 1$ distinct policies of the form specified in the lemma. Let the first $\tau(b) + 1$ policies of such form visited, in sequence, be $\pi_1 = xy_1z_1, \pi_2 = xy_2z_2, \ldots, \pi_{\tau(b)+1} = xy_{\tau(b)+1}z_{\tau(b)+1}$, with $|y_1| = |y_2| = \cdots = |y_{\tau(b)+1}| = b$. By Theorem 2, we have

$$\pi_{\tau(b)+1} \succ \pi_{\tau(b)} \succ \cdots \succ \pi_1. \tag{5}$$

We focus on the "y" part of the policies, which corresponds to states in B_r. Specifically we consider the improvable states therein. For $t \in \{1, 2, \ldots, \tau(b) + 1\}$, let $IS_t = \mathbf{IS}(\pi_t) \cap B_r$. Now consider the sequence $(y_1, IS_1), (y_2, IS_2), \ldots, (y_{\tau(b)+1}, IS_{\tau(b)+1})$. By construction, BSPI ensures that for $t \in \{1, 2, \ldots, \tau(b)\}$, y_{t+1} is the string obtained by taking y_t and switching all the states in IS_t. Thus, the sequence must follow a path in the TBT for b-state MDPs rooted at (y_1, IS_1) . However, since the sequence exceeds $\tau(b)$ in length, it must violate (4): that is, there must exist i and j, $1 \leq i < j \leq \tau(b) + 1$, such that $L^-_{y_i, IS_i} \cap L^+_{y_j, IS_j} \neq \emptyset$.

Indeed let $y^\star$ be a b-length bit-string that belongs both to $L^-_{y_i, IS_i}$ and to $L^+_{y_j, IS_j}$. Consider the policy $xy^\star z_j$, and also consider condition (2) of the lemma. We see that $xy^\star z_j$ differs from π_j only on states in $\mathbf{IS}(\pi_j)$, and it differs from π_i (if at all) only on states in $S \setminus \mathbf{IS}(\pi_i)$. Therefore, we must have

$$xy^\star z_j \succ \pi_j \text{ and } \pi_i \succeq xy^\star z_j. \tag{6}$$

From (6), we get $\pi_i \succ \pi_j$, which contradicts (5). □

We are now ready to bound the number of occurrences of the prefix x in any trajectory taken by BSPI.

Lemma 8 (Prefix occurrences)**.** *BSPI can encounter at most $\tau(b)^{m-r+1}$ policies of the form $\pi = xy$.*

Proof. We prove this lemma by induction on r. As the base case, consider $r = m$, for which the claim follows directly from Lemma 7. Our induction hypothesis is that the claim holds for $r = q + 1$, where $q \in \{1, 2, \ldots, m-1\}$. Now consider $r = q$.

If BSPI never visits a policy prefixed by x, there is nothing left to prove. Otherwise, let the first such policy visited be xy_1z_1, where y_1 is a b-length bit-string. By the induction hypothesis, BSPI can visit policies of the form xy_1z at most $(\tau_b)^{m-q}$ times. Therefore, within $(\tau_b)^{m-q}$ iterations, the algorithm must visit a policy of the form $xy_1z_1^\star$, in which the $z_1^\star$ portion has no improvable states. If y_1, too, has no improvable states in $xy_1z_1^\star$, it follows from Lemma 6 that no more

x-prefixed policies will be visited, and so we are done. If y_1 does contain improvable states in $xy_1z_1^\star$, BSPI will update to a new policy $xy_2z_1^\star$. We can now repeat the argument in this paragraph by taking $xy_2z_1^\star = xy_2z_2$.

If the argument above is indeed made τ_b times, it implies that BSPI has visited policies $xy_1z_1^\star, xy_2z_2^\star, \ldots, xy_{\tau(b)}z_{\tau(b)}^\star$, where the "$z^\star$" portion of each policy has no improvable states. It follows from Lemma 7 that the "y" portion of $xy_{\tau(b)}z_{\tau(b)}^\star$ can have no improvable states, and so x can be a prefix in no policy that is subsequently visited. The total number of x-prefixed policies visited by BSPI is therefore at most $\tau(b) \cdot \tau(b)^{m-q} = \tau(b)^{m-q+1}$. □

Invoking Lemma 8 with an empty prefix x (that is, $r = 1$) gives us our final complexity bound for BSPI.

Theorem 9 (BSPI bound)**.** *BSPI, when run with parameter b, terminates after at most $\tau(b)^{n/b}$ iterations.*

Our assumption that n is a multiple of b is easily implemented by adding a few dummy states to the MDP; for general n, the bound is $\tau(b)^{\lceil n/b \rceil} \leq \tau(b) \cdot \tau(b)^{n/b}$, which is $O(\tau(b)^{n/b})$ if b is a constant.

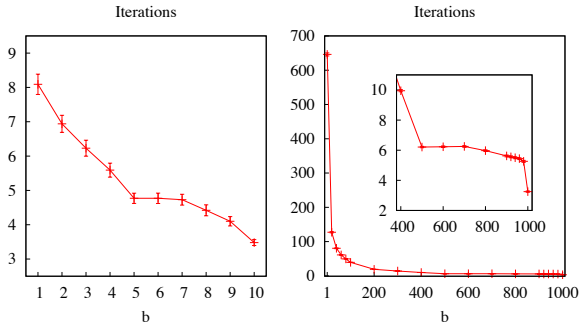
Effect of Batch Size. We observe a remarkable pattern when we invoke Theorem 9 with values of b for which $\tau(b)$ is known. As shown in Table 1, the bound gets progressively tighter as b is increased from 1 to 7. If it can be formally shown that $\tau(b)^{1/b}$ is a non-increasing function of b, it would follow that Howard's PI itself enjoys a bound of 1.6479^n.

In the absence of any such theoretical guarantee for $n \geq 8$, we undertake some preliminary experiments. We generate n-state MDPs through a procedure that picks $n/5$ states, with equal probability, as "target" states for each state-action pair: transition probabilities are picked uniformly at random from $[0, 1]$ and subsequently scaled to sum to 1. Each reward is a sample drawn from a standard normal distribution. The discount factor γ is set to 0.99.

As seen in Figure 2(a), the running time of BSPI (aggregated over MDPs and initialisations) decreases as b is increased. It should also be noted that the actual number of iterations in practice (≈ 5 for $b = 5$) is much smaller than the bounds given by Theorem 9 ($13^{10/5} = 169$ for $b = 7$). The decreasing trend in Table 1 motivates the hypothesis that BSPI will progressively run faster as b is increased beyond 7; the empirical results strongly support this hypothesis. For 1000-state MDPs (Figure 2(b)), Howard's PI is two orders

Table 1: Dependence of bound in Theorem 9 on b.

b	$\tau(b)$	Base of exponent in bound ($\tau(b)^{1/b}$)
1	2	2
2	3	1.7321
3	5	1.7100
4	8	1.6818
5	13	1.6703
6	21	1.6611
7	33	1.6479



(a) $n = 10$ (b) $n = 1000$

Figure 2: Performance of BSPI. Each point is an average over 100 randomly-generated MDP instances and initial policies.

of magnitude more efficient than BSPI with $b = 7$. While these aggregate experimental results from a specific family on MDPs are by no means definitive, they certainly motivate a deeper investigation into the structure of BSPI, which would also open up a new approach to the analysis of Howard's PI.

5 Conclusion

In this paper, we consider Policy Iteration (PI) [Howard, 1960], an elegant and widely-used family of algorithms for MDP planning. Although PI is efficient in practice, its theoretical analysis has been a hard nut to crack even for 2-action MDPs. It took nearly four decades to show the first non-trivial upper bound of $O(2^n/n)$ for Howard's PI [Mansour and Singh, 1999], a bound that has thereafter been optimised even for constant factors [Hollanders *et al.*, 2014]. We present a worst-case bound of $O(1.6479^n)$ iterations for PI, which is an *exponential* improvement both over the tightest existing worst case bound of $O(2^n/n)$, and the tightest existing expectation bound of $O(1.7172^n)$. Our result therefore takes a position of prominence in the body of literature on PI.

Our bound of $O(1.6479^n)$ iterations is shown for an instance of Batch-Switching Policy Iteration (BSPI), a family of PI algorithms that we introduce in this paper. By varying a batch size b, BSPI interpolates between Howard's PI and another previously-studied variant called Simple PI [Melekopoglou and Condon, 1994]. For analysing BSPI, we devise a construct called trajectory-bounding trees, which have also been considered previously in the form of "order-regular" matrices [Hansen, 2012; Gerencsér *et al.*, 2015]. Both our theoretical analysis and supplementary experiments suggest that Howard's PI might indeed be more efficient than the BSPI variant we formally bound. We believe the ideas introduced in this paper can benefit the study of Howard's PI.

Recently, the authors have proposed improved randomised algorithms for MDPs with more than two actions per state [Kalyanakrishnan *et al.*, 2016]. In future work, we aim to generalise BSPI and its analysis to the same setting.

Acknowledgments

Shivaram Kalyanakrishnan was partially supported by DST INSPIRE Faculty grant DST/INSPIRE/04/2013/001156.

References

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

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

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

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

[Gerencsér *et al.*, 2015] Balázs Gerencsér, Romain Hollanders, Jean-Charles Delvenne, and Raphaël M. Jungers. A complexity analysis of policy iteration through combinatorial matrices arising from unique sink orientations, 2015. URL: http://arxiv.org/pdf/1407.4293v2.pdf.

[Hansen and Zwick, 2010] Thomas Dueholm Hansen and Uri Zwick. Lower bounds for Howard's algorithm for finding minimum mean-cost cycles. In *Proceedings of the Twenty-second International Symposium on Algorithms and Computation (ISAAC 2011)*, pages 425–426. Springer, 2010.

[Hansen *et al.*, 2013] Thomas Dueholm Hansen, Peter Bro Miltersen, and Uri Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM*, 60(1):1–16, 2013.

[Hansen, 2012] Thomas Dueholm Hansen. *Worst-case Analysis of Strategy Iteration and the Simplex Method*. PhD thesis, Department of Computer Science, Aarhus University, July 2012.

[Hollanders *et al.*, 2012] Romain Hollanders, Balázs Gerencsér, and Jean-Charles Delvenne. 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)*, pages 5997–6002. IEEE, 2012.

[Hollanders *et al.*, 2014] Romain Hollanders, Balázs Gerencsér, Jean-Charles Delvenne, and Raphaël M. Jungers. Improved bound on the worst case complexity of policy iteration, 2014. URL: http://arxiv.org/pdf/1410.7583v1.pdf.

[Howard, 1960] Ronald A. Howard. *Dynamic programming and Markov processes*. MIT Press, 1960.

[Kalyanakrishnan *et al.*, 2016] Shivaram Kalyanakrishnan, Neeldhara Misra, and Aditya Gopalan. Randomised procedures for initialising and switching actions in policy iteration. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, 2016. To appear.

[Littman *et al.*, 1995] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, pages 394–402. Morgan Kaufmann, 1995.

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

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

[Post and Ye, 2013] Ian Post and Yinyu Ye. 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)*, pages 1465–1473. Morgan Kaufmann, 2013.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes*. Wiley, 1994.

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

[Szabó and Welzl, 2001] Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *Proceedings of the Forty-second Annual Symposium on Foundations of Computer Science (FOCS 2001)*, pages 547–555. IEEE, 2001.

[Ye, 2011] Yinyu Ye. 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, 2011.