May 2021 arXiv preprint arXiv:2106.00746

On-Line Policy Iteration for Infinite Horizon Dynamic Programming

Dimitri Bertsekas†

Abstract

In this paper we propose an on-line policy iteration (PI) algorithm for finite-state infinite horizon discounted dynamic programming, whereby the policy improvement operation is done on-line, only for the states that are encountered during operation of the system. This allows the continuous updating/improvement of the current policy, thus resulting in a form of on-line PI that incorporates the improved controls into the current policy as new states and controls are generated. The algorithm converges in a finite number of stages to a type of locally optimal policy, and suggests the possibility of variants of PI and multiagent PI where the policy improvement is simplified. Moreover, the algorithm can be used with on-line replanning, and is also well-suited for on-line PI algorithms with value and policy approximations.

1. SIMPLIFIED ON-LINE POLICY ITERATION FOR DISCOUNTED PROBLEMS

We introduce new on-line variants of the classical policy iteration (PI) algorithm for finite-state discounted infinite horizon dynamic programming (DP) problems. The common characteristic of these variants is that, in addition to being suitable for on-line implementation, they are simplified in two ways:

(a) They perform policy improvement operations only for the states that are encountered during the on-line operation of the system.

(b) The policy improvement operation is simplified in that it uses approximate minimization over the Q-factors of the current policy at the current state.

Despite these simplifications, we show that our algorithms generate a sequence of improved policies, which converge to a policy with a local optimality property. Moreover, with an enhancement of the policy improvement operation, which involves a form of exploration, they converge to a globally optimal policy.

The motivation for our work comes from the rollout algorithm; see the author's reinforcement learning books [Ber19] and [Ber20], which provide many additional references. This algorithm starts from some available "base policy" and implements on-line an improved "rollout policy," which is the one that would be obtained from the first iteration of the PI algorithm starting from the base policy. In the algorithm of the

† Fulton Professor of Computational Decision Making, ASU, Tempe, AZ, and McAfee Professor of Engineering, MIT, Cambridge, MA.

present paper, the data accumulated from the rollout implementation is used to improve on-line the base policy, and to asymptotically obtain a policy that is either locally or globally optimal.

We assume a discrete-time dynamic system with states $1,\ldots,n$, and we use a transition probability notation. We denote states by the symbol x and successor states by the symbol y. The control/action is denoted by u, and is constrained to take values in a given finite constraint set $U(x)$, which may depend on the current state x. The use of a control u at state x specifies the transition probability $p_{xy}(u)$ to the next state y, at a cost $g(x,u,y)$.

A policy $\pi = \{\mu_0, \mu_1, \ldots\}$ is a sequence of functions from state to control that satisfies the control constraint, i.e., $\mu_k(x) \in U(x)$ for all x and k. Given a policy π and an initial state x_0, the system becomes a Markov chain whose generated trajectory under π, denoted $\{x_0, x_1, \ldots\}$, has a well-defined probability distribution. The corresponding total expected cost is

$$J_\pi(x_0) = \lim_{N\to\infty} E\left\{\sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), x_{k+1}\big) \;\Big|\; x_0,\, \pi\right\}, \qquad x_0 = 1,\ldots,n,$$

where $\alpha < 1$ is the discount factor. The above expected value is taken with respect to the joint distribution of the states $x_1, x_2, \ldots$, conditioned on the initial state being x_0 and the use of π. The optimal cost starting from a state x, i.e., the minimum of $J_\pi(x)$ over all policies π, is denoted by $J^*(x)$. We will view J^* as the vector of the n-dimensional space $\Re^n$ that has components $J^*(1),\ldots,J^*(n)$.†

A stationary policy is a policy of the form $\pi = \{\mu, \mu, \ldots\}$, and for brevity, it is referred to as the "policy μ." The cost of μ starting from state x is denoted by $J_\mu(x)$, and is given by

$$J_\mu(x_0) = \lim_{N\to\infty} E\left\{\sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu(x_k), x_{k+1}\big) \;\Big|\; x_0,\, \mu\right\}, \qquad x_0 = 1,\ldots,n.$$

We can view J_μ as the vector in $\Re^n$ that has components $J_\mu(1),\ldots,J_\mu(n)$. We say that μ is optimal if

$$J_\mu(x) = J^*(x) = \min_\pi J_\pi(x), \qquad x = 1,\ldots,n.$$

It is well known that there exists an optimal stationary policy; see e.g., the books [Ber12], [Put94], which provide an extensive analysis of discounted finite-state infinite horizon DP problems.

The theory and algorithms for our problem are conveniently stated with the use of abstract notation, as in the book [Ber18]. In particular, for each policy μ, we introduce the operator $T_\mu : \Re^n \mapsto \Re^n$, which maps a vector $J \in \Re^n$ to the vector $T_\mu J \in \Re^n$ that has components

$$(T_\mu J)(x) = \sum_{y=1}^{n} p_{xy}\big(\mu(x)\big)\big(g(x,\mu(x),y) + \alpha J(y)\big), \qquad x = 1,\ldots,n. \tag{1.1}$$

† In our notation, $\Re^n$ is the n-dimensional Euclidean space and all vectors in $\Re^n$ are viewed as column vectors. Moreover, all vector inequalities $J \le J'$ are meant be componentwise, i.e., $J(x) \le J'(x)$ for all $x = 1,\ldots,n$.

We also introduce the operator $T : \Re^n \mapsto \Re^n$ defined by

$$(TJ)(x) = \min_{u \in U(x)} \sum_{y=1}^{n} p_{xy}(u)\big(g(x,u,y) + \alpha J(y)\big), \qquad x = 1, \ldots, n. \tag{1.2}$$

An important property is that T_μ and T are monotone, i.e., that for all $J, J' \in \Re^n$,

$$T_\mu J \leq T_\mu J', \qquad TJ \leq TJ', \qquad \text{if } J \leq J'.$$

Another important property is that T_μ and T are sup-norm contractions, so that the costs $J_\mu(x)$, $x = 1, \ldots, n$, are the unique solution of Bellman's equation

$$J_\mu(x) = \sum_{y=1}^{n} p_{xy}\big(\mu(x)\big)\big(g(x,\mu(x),y) + \alpha J_\mu(y)\big), \qquad x = 1, \ldots, n, \tag{1.3}$$

which can also be written as the fixed point equation $J_\mu = T_\mu J_\mu$. Similarly, the optimal costs $J^*(x)$, $x = 1, \ldots, n$, are the unique solution of Bellman's equation

$$J^*(x) = \min_{u \in U(x)} \sum_{y=1}^{n} p_{xy}(u)\big(g(x,u,y) + \alpha J^*(y)\big), \qquad x = 1, \ldots, n, \tag{1.4}$$

or $J^* = TJ^*$. A consequence of this is that the following optimality conditions hold

$$T_\mu J^* = TJ^* \qquad \text{if and only if} \qquad \mu \text{ is optimal}, \tag{1.5}$$

$$T_\mu J_\mu = TJ_\mu \qquad \text{if and only if} \qquad \mu \text{ is optimal}. \tag{1.6}$$

The contraction property also implies that the value iteration (VI) algorithms

$$J^{k+1} = T_\mu J^k, \qquad J^{k+1} = TJ^k$$

generate sequences $\{J^k\}$ that converge to J_μ and J^*, respectively, from any starting vector $J^0 \in \Re^n$.

Policy iteration (PI) is a major alternative to VI, and generates a sequence of policies. In the classical form of the algorithm, the current policy μ is improved by finding $\tilde{\mu}$ that satisfies

$$T_{\tilde{\mu}} J_\mu = TJ_\mu$$

[i.e., by minimizing for all x in the right-hand side of Eq. (1.2) with J_μ in place of J]. The improved policy $\tilde{\mu}$ is evaluated by solving the linear system of equations $J_{\tilde{\mu}} = T_{\tilde{\mu}} J_{\tilde{\mu}}$, and then $(J_{\tilde{\mu}}, \tilde{\mu})$ becomes the new cost vector-policy pair, which is used to start a new iteration. Thus the PI algorithm starts with a policy μ^0 and generates a sequence $\{\mu^k\}$ according to

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}, \qquad T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}, \tag{1.7}$$

where on the left we have the policy evaluation equation, and on the right we have the policy improvement equation.

The preceding PI algorithm has several weaknesses, which make it unsuitable for problems with a large number of states n. The principal difficulty is that at each iteration k, the current policy μ^k must be evaluated at all states, i.e., $J_{\mu^k}(x)$ must be computed for all x. A second difficulty is that the policy improvement operation

$$\mu^{k+1}(x) \in \arg\min_{u\in U(x)} \sum_{y=1}^{n} p_{xy}(u)\big(g(x,u,y) + \alpha J_{\mu^k}(y)\big), \qquad x = 1,\ldots,n, \tag{1.8}$$

must be performed at each state x, which makes it unsuitable for problems where the number of state-control pairs (x,u) is large. The purpose of this paper is to propose variants of the PI algorithm that alleviate both of these difficulties.

2. ON-LINE VARIANTS OF POLICY ITERATION

We introduce a PI algorithm, which starts at time 0 with a state-policy pair (x_0,μ^0) and generates on-line a sequence of state-policy pairs (x_k,μ^k). We view x_k as the current state of a system operating on line under the influence of the policies $\mu^1,\mu^2,\ldots$. In our algorithm, μ^{k+1} *may differ from* μ^k *only at state* x_k. The control $\mu^{k+1}(x_k)$ and the state x_{k+1} are generated as follows:

We consider the Q-factors

$$Q_{\mu^k}(x_k,u) = \sum_{y=1}^{n} p_{x_k y}(u)\big(g(x_k,u,y) + \alpha J_{\mu^k}(y)\big), \tag{2.1}$$

and we select the control $\mu^{k+1}(x_k)$ from within the constraint set $U(x_k)$ with sufficient accuracy to satisfy the sequential improvement condition

$$Q_{\mu^k}\big(x_k,\mu^{k+1}(x_k)\big) \le J_{\mu^k}(x_k), \tag{2.2}$$

with strict inequality whenever this is possible.† For $x \ne x_k$ the policy controls are not changed:

$$\mu^{k+1}(x) = \mu^k(x) \quad \text{for all } x \ne x_k.$$

† By this we mean that if $\min_{u\in U(x_k)} Q_{\mu^k}(x_k,u) < J_{\mu^k}(x_k)$ we select a control u_k that satisfies

$$Q_{\mu^k}(x_k,u_k) < J_{\mu^k}(x_k), \tag{2.3}$$

and set $\mu^{k+1}(x_k) = u_k$, and otherwise we set $\mu^{k+1}(x_k) = \mu^k(x_k)$ [so Eq. (2.2) is satisfied]. Such a control selection may be obtained by a number of schemes, including brute force calculation and random search based on Bayesian optimization. The needed values of the Q-factor Q_{μ^k} and cost J_{μ^k} may be obtained in several ways, depending on the problem at hand, including by on-line simulation.

The next state x_{k+1} is generated randomly according to the transition probabilities $p_{x_k x_{k+1}}\big(\mu^{k+1}(x_k)\big)$.

We first show that the current policy is monotonically improved.

Proposition 2.1: We have

$$J_{\mu^{k+1}}(x) \leq J_{\mu^k}(x), \qquad \text{for all } x \text{ and } k,$$

with strict inequality for $x = x_k$ (and possibly other values of x) if $\min_{u \in U(x_k)} Q_{\mu^k}(x_k, u) < J_{\mu^k}(x_k)$.

Proof: The policy update is done under the condition (2.2). By using the monotonicity of $T_{\mu^{k+1}}$, we have for all $\ell \geq 1$,

$$T^{\ell+1}_{\mu^{k+1}} J_{\mu^k} \leq T^{\ell}_{\mu^{k+1}} J_{\mu^k} \leq J_{\mu^k}, \tag{2.4}$$

so by taking the limit as $\ell \to \infty$ and by using the convergence property of VI ($T^{\ell}_{\mu^{k+1}} J \to J_{\mu^{k+1}}$ for any J), we obtain $J_{\mu^{k+1}} \leq J_{\mu^k}$. Moreover, the algorithm selects $\mu^{k+1}(x_k)$ so that

$$(T_{\mu^{k+1}} J_{\mu^k})(x_k) = Q_{\mu^k}(x_k, u_k) < J_{\mu^k}(x_k) \qquad \text{if} \quad \min_{u \in U(x_k)} Q_{\mu^k}(x_k, u) < J_{\mu^k}(x_k),$$

[cf. Eq. (2.3)], so that by using Eq. (2.4), we have $J_{\mu^{k+1}}(x_k) < J_{\mu^k}(x_k)$. **Q.E.D.**

Convergence to a Locally Optimal Policy

We next discuss the convergence and optimality properties of the algorithm. We introduce a definition of local optimality of a policy, whereby the policy selects controls optimally only within a subset of states.

Definition 2.1: Given a subset of states X and a policy μ, we say that μ *is locally optimal over* X if μ is optimal for the problem where the control is restricted to take the value $\mu(x)$ at the states $x \notin X$, and is allowed to take any value $u \in U(x)$ at the states $x \in X$.

Roughly speaking, μ is locally optimal over X, if μ is acting optimally within X, but under the (incorrect) assumption that once the state of the system gets to a state x outside X, there will be no option to select control other than $\mu(x)$. Thus if the choices of μ outside X are poor, its choices within X may also be poor.

Mathematically, μ is locally optimal over X if

$$J_\mu(x) = \min_{u \in U(x)} \sum_{y=1}^{n} p_{xy}(u)\big(g(x,u,y) + \alpha J_\mu(y)\big), \qquad \text{for all } x \in X,$$

$$J_\mu(x) = \sum_{y=1}^{n} p_{xy}\big(\mu(x)\big)\Big(g\big(x,\mu(x),y\big) + \alpha J_\mu(y)\Big), \qquad \text{for all } x \notin X,$$

which can be written compactly as

$$(T_\mu J_\mu)(x) = (TJ_\mu)(x), \qquad \text{for all } x \in X. \tag{2.5}$$

Note that this is different than (global) optimality of μ, which holds if and only if the above condition holds for all $x = 1, \ldots, n$, rather than just for $x \in X$ [cf. Eq. (1.6)]. However, it can be seen from Definition 2.1 that a (globally) optimal policy is also locally optimal within any subset of states.

Our main convergence result is the following.

Proposition 2.2: Let $\overline{X}$ be the subset of states that are repeated infinitely often within the sequence $\{x_k\}$. Then the corresponding sequence $\{\mu^k\}$ converges finitely to some policy $\overline{\mu}$ in the sense that $\mu^k = \overline{\mu}$ for all k after some index $\overline{k}$. Moreover $\overline{\mu}$ is locally optimal within $\overline{X}$, while $\overline{X}$ is invariant under $\overline{\mu}$, in the sense that

$$p_{xy}\big(\overline{\mu}(x)\big) = 0 \qquad \text{for all } x \in \overline{X} \text{ and } y \notin \overline{X}.$$

Proof: The cost function sequence $\{J_{\mu^k}\}$ is monotonically nondecreasing (cf. Prop. 2.1). The number of policies μ is finite in view of the finiteness of the state and control spaces. Therefore, the number of corresponding functions J_μ is also finite, so J_{μ^k} converges in a finite number of steps to some $\bar{J}$, which in view of the algorithm's construction [selecting $u_k = \mu^k(x_k)$ if $\min_{u \in U(x_k)} Q_{\mu^k}(x_k, u) = J_{\mu^k}(x_k)$; cf. Eq. (2.3)], implies that μ^k will remain unchanged at some $\overline{\mu}$ with $J_{\overline{\mu}} = \bar{J}$ after some sufficiently large k.

We will show that the local optimality condition (2.5) holds for $X = \overline{X}$ and $\mu = \overline{\mu}$. In particular, we have $x_k \in \overline{X}$ and $\mu^k = \overline{\mu}$ for all k greater than some index, while for every $x \in \overline{X}$, we have $x_k = x$ for infinitely many k. It follows that for all $x \in \overline{X}$,

$$Q_{\overline{\mu}}\big(x, \overline{\mu}(x)\big) = J_{\overline{\mu}}(x), \tag{2.6}$$

while by the construction of the algorithm,

$$Q_{\overline{\mu}}\big(x,u\big) \geq J_{\overline{\mu}}(x), \qquad \text{for all } u \in U(x), \tag{2.7}$$

since the reverse would imply that $\mu^{k+1}(x) \neq \mu^k(x)$ for infinitely many k [cf. Eq. (2.3)]. Condition (2.6) can be written as $J_{\overline{\mu}}(x) = (T_{\overline{\mu}}J_{\overline{\mu}})(x)$ for all $x \in \overline{X}$, and combined with Eq. (2.7), implies that $(T_{\overline{\mu}}J_{\overline{\mu}})(x) = (TJ_{\overline{\mu}})(x)$ for all $x \in \overline{X}$. This is the local optimality condition (2.5) with $X = \overline{X}$ and $\mu = \overline{\mu}$.

To show that $\overline{X}$ is invariant under $\overline{\mu}$, we argue by contradiction: if this were not so, there would exist a state $x \in \overline{X}$ and a state $y \notin \overline{X}$ such that

$$p_{xy}\big(\overline{\mu}(x)\big) > 0,$$

implying that y would be generated following the occurrence of x infinitely often within the sequence $\{x_k\}$, and hence would have to belong to $\overline{X}$ (by the definition of $\overline{X}$). **Q.E.D.**

Note an implication of the invariance property of the set $\overline{X}$ shown in the preceding proposition. We have that $\overline{\mu}$ is (globally) optimal under the assumption that for every policy there is no strict subset of states that is invariant.

A Counterexample to Global Optimality

The following deterministic example † shows that the policy $\overline{\mu}$ obtained by the algorithm need not be (globally) optimal. Here there are three states 1, 2, and 3. From state 1 we can go to state 2 at cost 1, and to state 3 at cost 0, from state 2 we can go to states 1 and 3 at cost 0, and from state 3 we can go to state 2 at cost 0 or stay in 3 at a high cost (say 10). The discount factor is $\alpha = 0.9$. Then it can be verified that the optimal policy is

$$\mu^*(1) : \text{Go to 3}, \qquad \mu^*(2) : \text{Go to 3}, \qquad \mu^*(3) : \text{Go to 2},$$

with optimal costs

$$J^*(1) = J^*(2) = J^*(3) = 0,$$

while the policy

$$\overline{\mu}(1) : \text{Go to 2}, \qquad \overline{\mu}(2) : \text{Go to 1}, \qquad \overline{\mu}(3) : \text{Stay at 3},$$

is strictly suboptimal, but is locally optimal over the set of states $\overline{X} = \{1, 2\}$. Moreover our on-line PI algorithm, starting from state 1 and the policy $\mu^0 = \overline{\mu}$, oscillates between the states 1 and 2, and leaves the policy μ^0 unchanged. Note also that $\overline{X}$ is invariant under $\overline{\mu}$, consistently with Prop. 2.2.

On-Line Variants of Policy Iteration with Global Optimality Properties

To address the local versus global convergence issue illustrated by the preceding example, we consider an alternative scheme, whereby in addition to u_k, we generate an additional control at a randomly chosen

† Thanks are due to Yuchao Li for constructing this example and for commenting on other aspects of the paper.

state $\overline{x}_k \neq x_k$.† In particular, assume that at each time k, in addition to u_k and x_{k+1} that are generated according to Eq. (2.3), the algorithm generates randomly another state $\overline{x}_k$ (all states are selected with positive probability), performs a policy improvement operation at that state as well, and modifies accordingly $\mu^{k+1}(\overline{x}_k)$. Thus, in addition to a policy improvement operation at each state within the generated sequence $\{x_k\}$, there is an additional policy improvement operation at each state within the randomly generated sequence $\{\overline{x}_k\}$.

Because of the random mechanism of selecting $\overline{x}_k$, it follows that at every state there will be a policy improvement operation infinitely often, which implies that the policy $\overline{\mu}$ ultimately obtained is (globally) optimal. Note also that *we may view the random generation of the sequence* $\{\overline{x}_k\}$ *as a form of exploration*. The probabilistic mechanism for generating the random sequence $\{\overline{x}_k\}$ may be guided by some heuristic reasoning, which aims to explore states with high cost improvement potential.

3. CONCLUDING REMARKS

We have developed a new on-line PI algorithm, which has a cost improvement property at each iteration, and offers some optimality guarantees. The algorithm is motivated and inspired by the rollout algorithm (a single iteration of PI), which is well-suited for on-line implementation; see the books [BeT96], [Ber17], [Ber19], [Ber20], and the references given there. Note that if there were no policy updates at all, our algorithm would become equivalent to the rollout algorithm with base policy μ^0. Thus our algorithm may be viewed as an enhanced version of a simplified rollout algorithm, which uses information collected on-line to improve the current base policy (at essentially no additional computational cost). A logical conclusion is that our algorithm should perform no worse than the rollout algorithm, which is known to perform well in practice, as it can be viewed as a single step of Newton's method, which underlies the PI algorithm [Kle68], [PoA69], [Hew71], [PuB78], [PuB79], [Ber20].

The structure of our algorithm suggests interesting possibilities for exploration, as well as combinations with approximation in policy and value space schemes, whereby policy and cost function approximations are constructed using the data that is collected on-line. Value and policy space approximation could also be applied to problems with infinite state and control spaces. There are many possibilities for the use of parallel computation within these contexts. Moreover, the on-line structure of our algorithm makes it suitable for adaptive control contexts, where some of the system and cost parameters may be changing over time.

Other possible variations of our algorithm include PI schemes with multistep lookahead, as well as optimistic variants, whereby the Q-factors (2.1) are evaluated approximately by using truncated simulation.

† It is also possible to choose multiple additional states at time k for a policy improvement operation, and this is well-suited for the use of parallel computation.

Extensions to other infinite horizon DP models, such as semi-Markov and average cost problems, are also possible. A particularly interesting class of models is deterministic and stochastic shortest path problems with an additional termination state t. For such problems, each generated system trajectory consists of a finite number of states and terminates at t, so the variant given at the end of the preceding section, which guarantees global optimality in the limit, does not apply. However, it is possible to construct a globally optimal policy by restarting the system from a randomly chosen initial state each time it reaches t. These possibilities are subjects for further research and computational experimentation.

4. REFERENCES

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[Ber12] Bertsekas, D. P., 2012. Dynamic Programming and Optimal Control, Vol. II, 4th edition, Athena Scientific, Belmont, MA.

[Ber17] Bertsekas, D. P., 2017. Dynamic Programming and Optimal Control, Vol. I, 4th Edition, Athena Scientific, Belmont, MA.

[Ber18] Bertsekas, D. P., 2018. Abstract Dynamic Programming, 2nd Edition, Athena Scientific, Belmont, MA (can be downloaded from the author's website).

[Ber19] Bertsekas, D. P., 2019. Reinforcement Learning and Optimal Control, Athena Scientific, Belmont, MA.

[Ber20] Bertsekas, D. P., 2020. Rollout, Policy Iteration, and Distributed Reinforcement Learning, Athena Scientific, Belmont, MA.

[Kle68] Kleinman, D. L., 1968. "On an Iterative Technique for Riccati Equation Computations," IEEE Trans. Aut. Control, Vol. AC-13, pp. 114-115.

[Hew71] Hewer, G., 1971. "An Iterative Technique for the Computation of the Steady State Gains for the Discrete Optimal Regulator," IEEE Trans. on Automatic Control, Vol. 16, pp. 382-384.

[PoA69] Pollatschek, M. A., and Avi-Itzhak, B., 1969. "Algorithms for Stochastic Games with Geometrical Interpretation," Management Science, Vol. 15, pp. 399-415.

[PuB78] Puterman, M. L., and Brumelle, S. L., 1978. "The Analytic Theory of Policy Iteration," in Dynamic Programming and Its Applications, M. L. Puterman (ed.), Academic Press, N. Y.

[PuB79] Puterman, M. L., and Brumelle, S. L., 1979. "On the Convergence of Policy Iteration in Stationary Dynamic Programming," Mathematics of Operations Research, Vol. 4, pp. 60-69.

[Put94] Puterman, M. L., 1994. Markovian Decision Problems, J. Wiley, N. Y.