01 Jan 2023

Deep Reinforcement Learning for Approximate Policy Iteration: Convergence Analysis and a Post-Earthquake Disaster Response Case Study

Abhijit Gosavi
Missouri University of Science and Technology, gosavia@mst.edu

L. (Lesley) H. Sneed
Missouri University of Science and Technology, sneedlh@mst.edu

L. A. Spearing

Recommended Citation

A. Gosavi et al., "Deep Reinforcement Learning for Approximate Policy Iteration: Convergence Analysis and a Post-Earthquake Disaster Response Case Study," *Optimization Letters*, Springer, Jan 2023.
The definitive version is available at https://doi.org/10.1007/s11590-023-02062-0

Optimization Letters
https://doi.org/10.1007/s11590-023-02062-0

ORIGINAL PAPER

Deep reinforcement learning for approximate policy iteration: convergence analysis and a post-earthquake disaster response case study

A. Gosavi[1] · **L. H. Sneed**[2] · **L. A. Spearing**[3]

Received: 30 May 2022 / Accepted: 21 July 2023

Abstract
Approximate policy iteration (API) is a class of reinforcement learning (RL) algorithms that seek to solve the long-run discounted reward Markov decision process (MDP), via the policy iteration paradigm, without learning the transition model in the underlying Bellman equation. Unfortunately, these algorithms suffer from a defect known as *chattering* in which the solution (policy) delivered in each iteration of the algorithm oscillates between improved and worsened policies, leading to sub-optimal behavior. Two causes for this that have been traced to the crucial policy improvement step are: (i) the inaccuracies in the policy improvement function and (ii) the exploration/exploitation tradeoff integral to this step, which generates variability in performance. Both of these defects are amplified by simulation noise. *Deep* RL belongs to a newer class of algorithms in which the resolution of the learning process is refined via mechanisms such as *experience replay* and/or *deep neural networks* for improved performance. In this paper, a new deep learning approach is developed for API which employs a more accurate policy improvement function, via an enhanced resolution Bellman equation, thereby reducing chattering and eliminating the need for exploration in the policy improvement step. Versions of the new algorithm for both the long-run discounted MDP and semi-MDP are presented. Convergence properties of the new algorithm are studied mathematically, and a post-earthquake disaster response case study is employed to demonstrate numerically the algorithm's efficacy.

Keywords Deep reinforcement learning · Model building · Noise reduction · Approximate policy iteration · Disaster response

Extended author information available on the last page of the article

Published online: 23 September 2023

1 Introduction

Markov decision processes (MDPs) [26] are sequential decision-making problems frequently used to make key decisions within discrete-event systems that *maximize* net *revenues* (usually called *rewards* in operations research) or equivalently *minimize* net *costs*. Semi-MDPs (SMDPs) [22, 26] are a variant of this class of problems in which time is an integral part of the performance metric or objective function by which the system's performance is evaluated. Dynamic programming (DP) methods, which primarily use the so-called *value function* (or cost-to-go function), are used to solve these problems exactly. In recent times, there has been significant interest in reinforcement learning or RL [6, 30], which can solve these problems with less exactitude when the so-called transition model underlying the MDP or SMDP is not available because of complex system dynamics, but a simulator of the system can be constructed with less effort. RL is derived from DP and has two major branches based on the two DP methodologies: value iteration and policy iteration [1]. RL methods derived from value iteration are rooted in the Q-Learning algorithm that employs the so-called Q-function, while those derived from policy iteration belong to a class called *Approximate Policy Iteration* or API (see [2] for a comprehensive survey). RL methods belonging to the value and policy iteration classes are now studied in many textbooks, a subset of which are [3, 8–10, 20].

The underlying mechanism in API is illustrated in Fig. 1. Starting with any arbitrary policy (generally a sub-optimal solution of the problem), the policy's performance is evaluated via the so-called policy evaluation step, an improved policy is generated via the policy improvement step, and the process returns to the policy evaluation step to evaluate the improved policy's performance. This process continues until no further improvement is possible in the policy's performance, when the algorithm is terminated. This mechanism works elegantly when the transition probabilities are known [23, 27, 33], which is referred to as a model-based setting. This is in contrast to the model-free setting in which RL operates. The policy *evaluation* step can be performed in a satisfactory manner using a version of temporal difference learning [30]. However, the policy improvement step is just as crucial for the algorithm's satisfactory behavior, and ideally requires the transition model, which, as stated above, is not available in the model-free setting of RL. Therefore, several schemes have been derived to circumvent this difficulty.

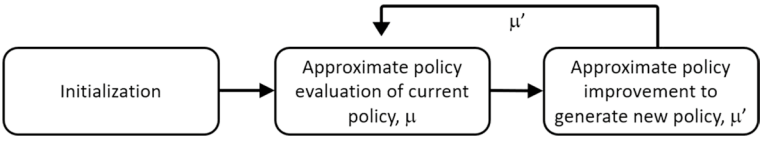

Fig. 1 The scheme underlying approximate policy iteration algorithms: the algorithm's cycle involves one policy evaluation step and one policy improvement step

An early approach for average reward MDPs and SMDPs employs a Q-function-based technique, bypassing the value function of classical DP, in which a separate function is used to perform policy improvement [17]. Another early approach for discounted MDPs employs the value function in combination with a genetic search for policy improvement [11]. Other works based on approximate policy improvement for discounted reward MDPs include the rollout algorithm [5], where only one step of policy improvement is performed, the so-called relational space technique [14], and the λ-policy iteration algorithm [28]. For discounted reward MDPs, the book [6] presents a class of methods called conservative approximate policy iteration (CAP), along with a phenomenon called *chattering*, associated with the *imperfect* policy improvement step in API. This phenomenon is defined as the oscillation between improved and worsened policies. For instance, during chattering, after an improved policy is generated (see Fig. 1), the policy evaluation and the subsequent policy iteration lead to a worse policy in the next cycle. This phenomenon does not occur when the transition model is available, as the transition probabilities allow for generating a perfect policy improvement function; however, chattering is a characteristic feature of almost all API algorithms, as the policy improvement function is *inaccurate* in current API methods. Research interest in API surged recently after superhuman performance was reported with a computer program playing *AlphaGo* and other computer games at Google via API based on policy gradients and stochastic policies [4, 29]. An additional notable feature of API is that it is faster than the older class of algorithms based on value iteration, e.g., Q-Learning [30], although the latter does *not* chatter.

Another strand of active research in RL is that of deep RL, where the word *deep* is associated with delivering high-resolution features in images and functions (see [25, 32] and references therein). The overarching goal of deep RL is to ameliorate a key drawback of model-free RL by delivering a high-fidelity Q-function or value function that closely approximates the ideal function produced by DP. One version of deep RL uses *experience replays*, which require the simulated transitions to occur repeatedly on a trajectory within the simulator, thereby minimizing the impact of the lack of the transition model on the Q-function. Another version of deep RL employs deep neural networks to generate a more accurate Q-function from the samples from a simulator.

This paper proposes a new algorithm for discounted reward SMDPs, based on a deep learning approach, that *indirectly* constructs the transition model, thereby minimizing the impact of noise and thus the chattering, while sidestepping the issue of heuristic exploration in the policy improvement. The algorithm is henceforth called *M*odel-building *A*pproximate *P*olicy iteration or MAP. Building the transition model, especially the transition probabilities, through counting transitions in a simulator or the real-world system is computationally intensive; yet researchers have

successfully built such models, albeit *without* any deep learning [31, 34, 35]. The novelty of MAP lies in its ability to build via deep learning the transition model required for policy improvement, while bypassing the approach of direct estimation of transition probabilities, and still delivering a high-fidelity version of the policy improvement function. The more accurate policy improvement function produced helps reduce the intensity of the phenomenon of chattering, as MAP makes progress and becomes stable, thereby enhancing the algorithm's efficacy. To the authors' best knowledge, this is the first API algorithm that seeks to reduce chattering and the need for exploration in the policy improvement step. How MAP can be particularly useful in the disaster response context, where the speed and efficiency of such an algorithm is imperative, is demonstrated via a post-earthquake disaster response case study from St. Louis, MO. It is hypothesized that the MAP algorithm will provide decision-support for emergency response quicker than current methods (e.g., CAP and Q-Learning).

2 Background

Notation, definitions, and assumptions needed in this paper are defined below:

- $i, j,$ and l: Index for states
- $\mathcal{S}$: The finite set of decision-making states
- $\mathcal{A}(i)$: The finite set of actions available in state i
- $\mathcal{A}$: The union of sets $\mathcal{A}(i)$ for all $i \in \mathcal{S}$
- μ: A stationary, deterministic policy
- $\mu(i)$: The action chosen in state i under policy μ
- $p(i, a, j)$: The probability of one transition from state i to state j when action $a \in \mathcal{A}(i)$ is chosen in state i
- $r(i, a, j)$: The immediate reward accrued in one transition from state i to state j under action $a \in \mathcal{A}(i)$ in state i
- $t(i, a, j)$: The mean value of the immediate time spent in one transition from state i to state j under action $a \in \mathcal{A}(i)$ in state i
- $\bar{r}(i, a) = \sum_{j \in \mathcal{S}} p(i, a, j) r(i, a, j)$: The mean reward earned in one transition from state i under action $a \in \mathcal{A}(i)$
- $\bar{t}(i, a) = \sum_{j \in \mathcal{S}} p(i, a, j) t(i, a, j)$: The mean time spent in one transition from state i under action $a \in \mathcal{A}(i)$
- $q \equiv (i, a)$: Short for the state-action pair (i, a)
- γ: The time rate of discounting
- $J_{\mu}(i)$: Value function at state i of policy μ
- $J^k(i)$: Estimate of the value function of state i in kth iteration
- α, β, η, and δ: Learning rates in reinforcement learning

- s, k, m, and n: Iteration indices in reinforcement learning
- $R^m(i,a)$ and $T^m(i,a)$: Estimate of $\bar{r}(i,a)$ and $\bar{t}(i,a)$, respectively, in the mth iteration
- $F_\mu^n(i,a)$: Estimate of future state-action value in nth iteration for the (i,a)th state-action pair under policy μ
- $\bar{F}_\mu(i,a) = \sum_{j\in\mathcal{S}} p(i,a,j)J_\mu(j)$: Expected future state-action value for the (i,a)th state-action pair under policy μ
- $w.p.1$: With probability 1

An SMDP is defined as a 5-tuple, $<\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\mathcal{T}>$, where $\mathcal{S}$ and $\mathcal{A}$ are defined above, while $\mathcal{P}$, $\mathcal{R}$, $\mathcal{T}$ denote the set of transition probability, reward, and time matrices, respectively, for every action. The MDP is a special case of the SMDP in which the transition time, $t(.,.,.)$ equals one everywhere. The SMDP will be studied in this paper under the following assumptions [26]:

A1 The Markov chain associated with every stationary, deterministic policy μ is *regular*, i.e., the transition probability matrix associated to the policy converges when raised to a sufficiently high power.

A2 The functions, $r(.,.,.)$ and $t(.,.,.)$, are both bounded.

A3 The reward is earned at the start of the state transition.

The objective function in this work is the long-run (i.e., infinite-horizon) discounted reward. For each state $i \in \mathcal{S}$, under a given policy μ, it equals:

$$\lim_{C\to\infty} \mathsf{E}\left[\sum_{c=0}^{C} \exp\left(-c\gamma t(x^c,\mu(x^c),x^{c+1})\right) r(x^c,\mu(x^c),x^{c+1}) \middle| x^0 = i\right],$$

where x^c denotes the state in the cth transition of the Markov chain driven by policy μ, and $\mathsf{E}[.]$ denotes the expectation operator over all trajectories under policy μ. The goal of the SMDP is to maximize the function defined above for every $i \in \mathcal{S}$. The Bellman equation for a given policy, which is employed in policy iteration and MAP, is now presented via a fundamental result [1, 26]:

Theorem 1 *Under Assumptions* **A1**: **A3**, *for any stationary deterministic policy*, μ, *there exists a value function* $J_\mu : \mathcal{S} \to \mathfrak{R}$ *satisfying the following system of equations for all* $i \in S$,

$$J_\mu(i) = \bar{r}(i,\mu(i)) + \sum_{j\in S} p(i,\mu(i),j)\exp(-\gamma t(i,\mu(i),j))J_\mu(j). \tag{1}$$

The following two conditions will also be needed for the RL algorithm.

A4 If δ^s denotes the learning rate in the sth iteration, then:

$$\delta^s \in (0,1); \quad \sum_{s=1}^{\infty} \delta^s = \infty; \quad \sum_{s=1}^{\infty} (\delta^s)^2 < \infty.$$

The learning rates, α, η, and β, in the RL algorithms are assumed to follow the above condition, which is standard in stochastic approximation [6].

A5 (i.i.d) Random samples generated in the simulator for any given random variable are mutually independent and belong to an identical distribution.

Next, the root-finding version of the Robbins-Monro theorem is presented (see Prop 4.1, along with Example 4.2, from [6]):

Theorem 2 *Under Assumptions* **A4** *and* **A5**, *consider the stochastic gradient algorithm for minimizing a cost function*, $g : \mathfrak{R}^N \rightarrow \mathfrak{R}$, *which satisfies the conditions*: (*i*) $g(.)$ *is positive everywhere*, (*ii*) $g(.)$ *is continuously differentiable*, *and* (*iii*) $g(.)$*'s derivative is Lipschitz continuous*:

$$y^{s+1}(q) = y^s(q) - \delta^s \left[\nabla g(y^s(q)) + v^s(q)\right] \text{ for } q = 1, 2, \ldots, N$$

in which $v^s(.)$ *denotes the noise in the derivative's estimate in the sth iteration*, *where the noise has a zero mean and its second moment is bounded by a function of the derivative of* $g(.)$. *Then*, *w.p.*1:

$$\lim_{s \rightarrow \infty} \nabla g(y^s(q)) = 0 \text{ for } q = 1, 2, \ldots, N.$$

3 Steps in the MAP algorithm

Steps in the proposed MAP algorithm for a discounted reward SMDP are presented via Algorithm 1. Note that Step A in the algorithm is performed only once, while the algorithm iterates through Step B (policy evaluation, i.e., Eqs. (2) and (3) and Step C (policy improvement, i.e., Eq. (4) for K cycles. Further, note that the final policy delivered by the algorithm after K iterations is μ'. The expression to the right within square brackets in Eq. (4) is the *policy improvement function* discussed in Sect. 1. Finally, note that MAP's MDP version can be obtained by setting $t(.,.,.) = 1$ everywhere in the algorithm.

Algorithm 1 (MAP)

In the steps below: m, k, and n denote the number of iterations in Steps A, B1, and B2, respectively. Initialize $m_{\max}$, $k_{\max}$, and $n_{\max}$ to large positive integers as surrogates for infinity.

Step A: Model-Building:

• Set $m = 0$. For all $l \in \mathcal{S}$ and $u \in \mathcal{A}(l)$, set $R^m(l,u) = 0$ and $T^m(l,u) = 0$.

while $m \leq m_{max}$ **do**

• Let the current state be i. Select action a such that each action is selected with equal probability. Let j be the next state. Update the expected immediate reward and immediate time for (i,a) as follows:

$$R^{m+1}(i,a) \leftarrow (1-\alpha^m)R^m(i,a) + \alpha^m[r(i,a,j)];$$

$$T^{m+1}(i,a) \leftarrow (1-\alpha^m)T^m(i,a) + \alpha^m[t(i,a,j)].$$

• If $m < m_{max}$ increment m by 1 and set $i \leftarrow j$.

end while

• **Repeat** Steps B and C for K iterations, where K is a large integer.

• **Step B: Policy Evaluation** Select any arbitrary policy, μ.

−−**Step B1: Current State Value:**

• Initialize $k = 0$ and for all $l \in \mathcal{S}$, $J^k(l) = 0$.

while $k \leq k_{max}$ **do**

• The system simulation is started at any arbitrary state. Let the current state be i. The current action is selected to be $\mu(i)$, since the policy being evaluated is μ. Let the next state be j. Update the current state value as follows:

$$J^{k+1}(i) \leftarrow (1-\eta^k)J^k(i) + \eta^k\left[r(i,\mu(i),j) + \exp(-\gamma t(i,\mu(i),j))J^k(j)\right]. \tag{2}$$

• If $k < k_{max}$, increment k by 1 and set $i \leftarrow j$.

end while

−− **Step B2: Future State-Action Value:**

• Initialize: Set $n = 0$. Set future state-action value for policy μ to 0, i.e., for all $l \in \mathcal{S}$ and $u \in \mathcal{A}(l)$, set $F_\mu^n(l,u) = 0$.

while $n \leq n_{max}$ **do**

• Let the current state be i. Select action a so that every action is selected with equal probability in every state. Let the next state be j. Update the future state-action value as follows:

$$F_\mu^{n+1}(i,a) \leftarrow (1-\beta^n)F_\mu^n(i,a) + \beta^n\left[J^\infty(j)\right], \tag{3}$$

where $J^\infty(.)$ is the final value of $J^k(.)$ obtained from Step B1.

• If $n < n_{\max}$, increment n by 1 and set $i \leftarrow j$.

end while

• **Step C: Policy Improvement:**

For each $i \in \mathcal{S}$, select

$$\mu'(i) \in argmax_{a\in A(i)}\left[R^\infty(i,a) + \exp(-\gamma T^\infty(i,a))F_\mu^\infty(i,a)\right], \tag{4}$$

where $R^\infty(.,.)$ and $T^\infty(.,.)$ are the final values of $R^m(.,.)$ and $T^m(.,.)$, respectively, obtained from Step A, while $F_\mu^\infty(.,.)$ denotes the final value of $F_\mu^n(.,.)$ obtained from Step B2. The new policy is μ'. Set $\mu' \leftarrow \mu$.

4 Convergence analysis

Many RL algorithms can be analyzed for convergence by showing that they belong to the following asynchronous update [6]:

$$X^{k+1}(i) \leftarrow (1-\eta^k)X^k(i) + \eta^k\big[G(X^k)(i) + w^k(i)\big], \tag{5}$$

where $G(.)$ denotes a transformation on the vector $\mathbf{X}$, $w^k(i)$ is a noise term in the kth iteration, and $\eta^k = 0$ if $X(i)$ is not updated in the kth iteration of the asynchronous algorithm. The value function $J(.)$ in Eq. (2) (Step B1) is represented in this analysis by a vector, $\mathbf{J}$. The transformation G and the noise vector $\mathbf{w}$ for the update in Eq. (2) are defined as follows:

$$G(J^k)(i) = \sum_{j\in S} p(i,\mu(i),j)\big[r(i,\mu(i),j) + \exp(-\gamma t(i,\mu(i),j))J^k(j)\big] \quad \forall i; \tag{6}$$

$$w^k(i) = r\big(i,\mu(i),\xi^k\big) + \exp\big(-\gamma t\big(i,\mu(i),\xi^k\big)\big)J^k(\xi^k) - \sum_{j\in S} p(i,\mu(i),j)\big[r(i,\mu(i),j) + \exp(-\gamma t(i,\mu(i),j))J^k(j)\big] \quad \forall i, \tag{7}$$

where ξ^k is the random state reached when action $\mu(i)$ is chosen in state i in the kth iteration. The following two lemmas are needed to show convergence.

Lemma 1 *The transformation $G(.)$ is contractive in the infinity norm.*

The proof of the above is closely related to that for $G(.)$'s counterpart used in MDPs and is hence provided in the Appendix.

Lemma 2 *The sequence $\{\mathbf{J}^k\}_{k=1}^{\infty}$ remains bounded.*

The proof of the above has similarities to an analogous result for Q-Learning [18], but since this is a different algorithm, a proof is necessary and presented in the Appendix.

Proposition 1 *The sequence $\{\mathbf{J}^k\}_{k=1}^{\infty}$ in the policy evaluation step (Step B1) converges to a solution of Eq. (1) w.p.1 under Assumptions* **A1**: **A5**.

Proof The noise term defined in Eq. (7) has a conditional mean of zero. Since the sequence $\{\mathbf{J}^k\}_{k=1}^{\infty}$ remains bounded from Lemma 2, the conditional second moment of the noise term is also bounded. The result then follows from Prop 4.4 of [6] after noting that $G(.)$ is contractive (from Lemma 1). □

The following result establishes the convergence of MAP.

Proposition 2 *The MAP algorithm converges to an optimal solution of the SMDP w.p.*1 *under Assumptions* **A1**:**A5**.

Proof Herein the Robbins-Monro theorem (Theorem 2) is used to show convergence of iterates in Steps A and B2. For the iterate, $R^m(.,.)$ in Step A, $g(.)$ in the Robbin-Monro theorem is defined, noting $q \equiv (i,a)$ and $y^s(q) \equiv R^m(i,a)$, as:

$$g(R^m(i,a)) = \frac{1}{2}(R^m(i,a) - \bar{r}(i,a))^2. \tag{8}$$

Then, $\nabla g(R^m(i,a)) = R^m(i,a) - \bar{r}(i,a)$. Also, the noise term here is defined as: $v^m(i,a) = \bar{r}(i,a) - r(i,a,\xi^m)$, where ξ^m denotes the random state reached when action a is chosen in state i. Then, Step A can be written in the form of the Robbins-Monro update as:

$$\begin{aligned} R^{m+1}(i,a) &= R^m(i,a) - \alpha^m[R^m(i,a) - r(i,a,\xi^m)] \\ &= R^m(i,a) - \alpha^m[R^m(i,a) - \bar{r}(i,a) + \bar{r}(i,a) - r(i,a,\xi^m)] \\ &= R^m(i,a) - \alpha^m \nabla g(R^m(i,a)) - \alpha^m v^m(i,a). \end{aligned}$$

Since the function $g(.)$ defined in Eq. (8) is always positive and continuously differentiable, and since its derivative, $(R^m(i,a) - \bar{r}(i,a))$, is linear in $R^m(.,.)$, the derivative is Lipschitz continuous; the derivative is also bounded from Assumption **A2**. That the noise term $v^m(i,a) = r(i,a,\xi^m) - \bar{r}(i,a)$ has a zero mean follows from the definition of $\bar{r}(.,.)$ (see Sect. 2) and from the i.i.d nature of the samples (Assumption **A5**). From Assumption **A2**, the variance of the noise is also bounded as a function of $g(.)$'s derivative, which is itself bounded. Then, Theorem 2 applies and the derivative of $g(.)$ converges to zero, i.e.,

$$\lim_{m\to\infty} R^m(i,a) = \bar{r}(i,a) \;\; \forall (i,a) \quad w.p.1. \tag{9}$$

Via very similar arguments and using the definition of $\bar{t}(i,a)$ from Sect. 2, it follows that:

$$\lim_{m\to\infty} T^m(i,a) = \bar{t}(i,a) \;\; \forall (i,a) \quad w.p.1. \tag{10}$$

From Prop. 1, $J^\infty(i) = J_\mu(i) \;\; \forall i$, which is bounded and a constant for any given value of i. Then, using arguments analogous to those above, it follows from the update of Step B2 in Eq. (3) and the definition of $\bar{F}_\mu(i,a)$ from Sect. 2 that

$$\lim_{n\to\infty} F_\mu^n(i,a) = \bar{F}_\mu(i,a) \equiv \sum_{j\in\mathcal{S}} p(i,a,j) J_\mu(j) \;\; \forall (i,a) \quad w.p.1.$$

The above together with Eqs. (9) and (10) implies that the policy iteration step (i.e., Step C) in MAP is equivalent to:

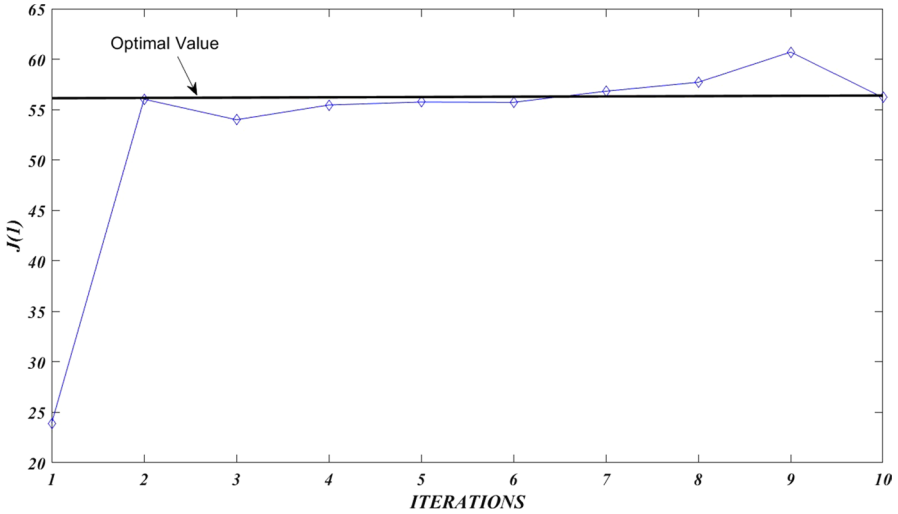

Fig. 2 Plot of MAP's value function for state 1, i.e., $J(1)$, against the number of iterations in the first numerical example: The flat line shows the optimal value, $J^*(1)$, obtained from policy iteration

Table 1 Comparison of value functions from policy iteration, MAP, CAP, and Q-Learning

State	Policy Iteration	MAP	CAP	Q-learning
1	56.10	56.20	56.58	60.54
2	38.76	39.64	38.78	30.22
3	26.07	24.94	25.76	26.54
4	24.97	25.82	26.02	30.36

$$\mu'(i) \in argmax_{a \in A(i)} \left[\bar{r}(i,a) + \exp(-\gamma \bar{t}(i,a)) \sum_{j \in \mathcal{S}} p(i,a,j) J_{\mu}(j) \right],$$

which mimics the policy improvement step in the classical policy iteration algorithm for SMDPs (see Chap 11. of [26]). This ensures the algorithm's convergence to an optimal solution of the SMDP *w.p.*1. □

5 Numerical results and case study

The first example for numerical testing is a discounted SMDP for which the transition model is known, allowing the computation of the optimal policy via DP. The problem has four states and two actions in each state, whose transition model data are provided in the Appendix. The optimal policy derived from the policy

iteration algorithm is: action 2 in states 1 and 3 and action 1 in states 2 and 4. MAP, CAP (see Appendix for details), and Q-Learning (for SMDPs [7]) return the same policy. The log rate, i.e., $log(k)/k$, from [19] was used for all learning rates, as it delivered the best results. Table 1 compares the *optimal* value function obtained from policy iteration of dynamic programming and that obtained from MAP, CAP, and Q-Learning. Figure 2 shows the behavior of MAP in generating the value function for a sample state.

A more complex problem from the domain of post-earthquake disaster response was chosen for testing MAP and benchmarking it against CAP and Q-Learning for SMDPs. The details of this case study are derived from [21].

A specific area prone to a major earthquake with a large number of seismically deficient buildings is modeled. An example of this is the downtown area of St. Louis, MO, which is about 240 km from the New Madrid Seismic Zone [15]. Three major incidents that could spur from an earthquake, Gas Leakage (G), Fire (F), and Building Collapse (BC), are used in the model. The state of the system is defined by the various incidents present. There are 8 states in the specific system studied here: $\{Stable\}$ (pre-earthquake), $\{G\}$, $\{F\}$,$\{BC\}$, $\{G, F\}$, $\{G, BC\}$, $\{F, BC\}$, and $\{G, F, BC\}$, numbered respectively 0 through 7. When an earthquake occurs, the system transitions to any of the so-called *primary* states, numbered 1 through 4, with probability PP (provided in the Appendix); these are the decision-making states. While only 4 decision-making states are included in the model, deriving the transition model is tedious, as there are numerous non-decision-making states driven by complex transition mechanisms. On the other hand, the system can be simulated easily using RL algorithms. When one of these 4 states is reached, a decision is made in regards to which of two agencies is asked to respond: A local agency or a federal agency. This corresponds to the two actions allowed in each decision-making state. After the agency is called, it takes time, called travel time, for the respondents to arrive on the scene. During this time, the system can transition into one of the significantly worse, so-called *secondary* states, numbered 5 through 7, or it can remain in the same state. The probabilities of these transitions, SP, are modeled from [21], which are also provided in the Appendix. After the respondents arrive on the scene, the time spent to neutralize the hazards and shift any injured persons to hospitals, eventually bringing the situation under control, is called the response time.

A local agency typically arrives sooner on the scene, i.e., has a shorter travel time, but is likely to have a lower volume of resources available and limited capability, i.e., has a longer response time. On the other hand, a federal agency typically takes longer to arrive, i.e., has a longer travel time, but is likely to have additional resources, i.e., has a shorter response time. There is thus a tradeoff in choosing the agency. A suitable decision needs to be made by the disaster managers depending on the level of hazard indicated in the primary state. A cost is assumed in the model for being in each of the 7 primary and secondary states. The cost incurred in a state, s, equals $HS(s)\tau(s)$, where $HS(.)$ is a hazard score associated to state s, and $\tau(s)$ equals the time for which the incident poses hazardous conditions for the persons affected [16]. Cost data used in the experiments are provided in the Appendix.

The solution delivered by MAP, as well as by CAP and Q-Learning, is to call the local agency for states 1, 2, and 4 and the federal agency for state 3. This is a

Table 2 Comparison of MAP, CAP, and Q-Learning in which K denotes the number of iterations needed in MAP and CAP

Metric	MAP	CAP	Q-Learning
K	10	81	–
Runtime (s)	3.8398	5.9403	4.2132

reasonable solution, as 1, 2, and 4 are lower hazard states (fire, gas leakage, or a combination of the two), while 3 (building collapse) is a higher hazard state. The algorithm thus demonstrates satisfactory behavior without requiring any explicit knowledge of the underlying system's dynamics, but numerical tests on instances of larger dimensions are beyond the scope of this paper. The computer programs were written in MATLAB and executed on a personal computer in a university setting using an Intel processor Intel(R)-Core(TM) equipped with 3.30 GHz within a 64-bit operating system. The log-rule learning rates were used again for all algorithms. Table 2 shows that MAP compares favorably against CAP and Q-Learning in runtime and against CAP in number of iterations needed for convergence; as Q-Learning is based on value iteration, which has a different nature, it cannot be compared on the basis of number of iterations. Further, to demonstrate the reduced chattering in MAP, Fig. 3 contrasts the runtime behavior of MAP and CAP.

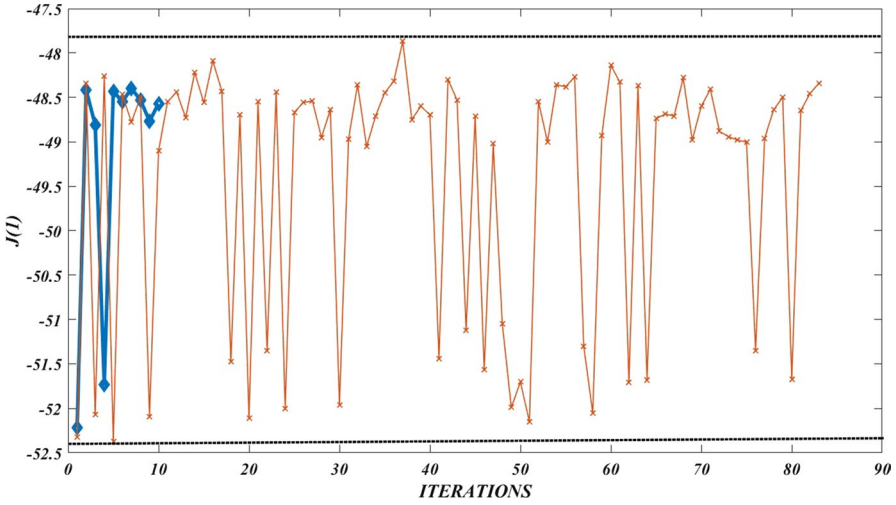

Fig. 3 Plot of $J(1)$, the value function of state 1, against the number of iterations for MAP and CAP: The diamonds represent MAP's values, and the stars represent CAP's values. The plot shows more significant chatter in CAP than in MAP in the zone of oscillation (shown within dotted lines) and a larger number of iterations needed by CAP for convergence

6 Conclusions and future research

This paper presented a new RL algorithm called Model-building Approximate Policy iteration, or MAP, based on a deep learning approach for approximate policy iteration (API). The novelty of MAP lies in its ability to build elements of the transition model required for policy improvement, while bypassing the approach of direct estimation of the transition probabilities, but delivering a high fidelity version of the policy improvement function. To the authors' best knowledge, this is the first API algorithm that reduces chattering associated with API and the need to explore via the policy improvement step. The algorithm was studied with a discounted SMDP, which applies more frequently in real-world problems than the more commonly studied MDP. Numerical testing showed MAP was able to return the optimal policy in the case of a problem with a known transition model. For a more complex problem *without* a known transition model, MAP was shown to return a satisfactory solution, delivering the solution faster than CAP and Q-Learning and with less chattering than CAP.

The reduced chattering, and ultimately the reduced runtime offered by MAP, as compared to CAP, paves the way for many different applications including future extensions to large scale-disaster response problems. Examples of such problems include equitable disaster response, rescue, and sheltering in the field [12], where disaster managers must balance multiple competing pressures and make decisions quickly. Future research can use MAP to help mitigate the disproportionate disaster impacts experienced by vulnerable populations [13].

Appendix

Proof (Lemma 1) Note that since $\gamma > 0$ and $t(.,.,.) > 0$, there exists a $\lambda \in (0,1)$ such that:

$$\max_{i,j\in\mathcal{S};a\in\mathcal{A}(i)} \exp(-\gamma t(i,a,j)) \leq \lambda. \tag{11}$$

From $G(.)$'s definition in Eq. (6), for any two vectors, $\mathbf{J}^k$ and $\overline{\mathbf{J}}^k$:

$$G(J^k)(i) - G(\overline{J}^k)(i) = \sum_{j\in S} p(i,\mu(i),j)\exp(-\gamma t(i,\bar{\mu}(i),j)\left[J^k(j) - \overline{J}^k(j)\right] \quad \forall i.$$

$$\text{Then, } \forall i: \left|G(J^k)(i) - G(\overline{J}^k)(i)\right| \leq \sum_{j\in\mathcal{S}} p(i,\mu(i),j)\exp(-\gamma t(i,\bar{\mu}(i),j)\max_{j\in\mathcal{S}}\left|J^k(j) - \overline{J}^k(j)\right|$$

$$\leq \sum_{j\in\mathcal{S}} p(i,\mu(i),j)\lambda \max_{j\in\mathcal{S}}\left|J^k(j) - \overline{J}^k(j)\right| \text{ from } Eq.(11)$$

$$= \sum_{j\in\mathcal{S}} p(i,\mu(i),j)\lambda||\mathbf{J}^k - \overline{\mathbf{J}}^k||_\infty$$

$$= \lambda||\mathbf{J}^k - \overline{\mathbf{J}}^k||_\infty \sum_{j\in\mathcal{S}} p(i,\mu(i),j) = \lambda||\mathbf{J}^k - \overline{\mathbf{J}}^k||_\infty.$$

Then, $\max_{i\in\mathcal{S}}\left|G(J^k)(i) - G(\overline{J}^k)(i)\right| = ||G(\mathbf{J}^k) - G(\overline{\mathbf{J}}^k)||_\infty \leq \lambda||\mathbf{J}^k - \overline{\mathbf{J}}^k||_\infty$. □

Proof (Lemma 2) It is claimed that for every $i \in \mathcal{S}$:

$$|J^k(i)| \leq M(1 + \lambda + \lambda^2 + \cdots + \lambda^k), \tag{12}$$

$$\text{where } M = \max\left\{\max_{i,j\in\mathcal{S},a\in\mathcal{A}(i)} |r(i,a,j)|, \max_{i\in\mathcal{S}} |J^1(i)|\right\}$$

and λ is defined via Eq. (11). Then, the result follows from the fact that:

$$\limsup_{k\to\infty} |J^k(i)| \leq M\frac{1}{1-\lambda} \text{ for all } i \in \mathcal{S}.$$

The claim in (12) is proved via induction. In asynchronous updating, two cases can occur:

Case 1 The value for a state visited in the kth iteration is updated: $J^{k+1}(i) = (1-\eta^k)J^k(i) + \eta^k\left[r(i,a,j) + \exp(-\gamma t(i,a,j))J^k(j)\right]$.

Case 2 The value for a state not visited in the kth iteration is not updated: $J^{k+1}(i) = J^k(i)$.

When the update is carried out as in Case 1:

$$|J^2(i)| \leq (1-\eta^1)|J^1(i)| + \eta^1|r(i,a,j) + \exp(-\gamma t(i,a,j)J^1(j)|$$

$$\leq (1-\eta^1)|J^1(i)| + \eta^1|r(i,a,j) + \lambda J^1(j)| \text{ from (11)}$$

$$\leq (1-\eta^1)M + \eta^1 M + \eta^1\lambda M \text{ from } M's \text{ definition}$$

$$< (1-\eta^1)M + \eta^1 M + \lambda M = M(1+\lambda) \text{ since } \eta^1 < 1$$

When the update is carried out as in Case 2: $|J^2(i)| = |J^1(i)| \leq M \leq M(1+\lambda)$. The claim thus holds for $k = 1$. When the claim holds for $k = m$, one has that:

$$|J^m(i)| \leq M(1 + \lambda + \lambda^2 + \cdots + \lambda^m) \quad \forall i.$$

Under Case 1: $\forall i$:

$$
\begin{aligned}
|J^{m+1}(i)| &\leq (1-\eta^m)|J^m(i)| + \eta^m |r(i,a,j) + \exp(-\gamma t(i,a,j))J^m(j)| \\
&\leq (1-\eta^m)|J^m(i)| + \eta^m |r(i,a,j) + \lambda J^m(j)| \\
&\leq (1-\eta^m)M(1+\lambda+\lambda^2+\cdots+\lambda^m) + \eta^m M + \eta^m \lambda M(1+\lambda+\lambda^2+\cdots+\lambda^m) \\
&= M(1+\lambda+\lambda^2+\cdots+\lambda^m) + \eta^m M\lambda^{m+1} \\
&\leq M(1+\lambda+\lambda^2+\cdots+\lambda^m) + M\lambda^{m+1} = M(1+\lambda+\lambda^2+\cdots+\lambda^m+\lambda^{m+1}).
\end{aligned}
$$

Under Case 2: $\forall i$:

$$
|J^{m+1}(i)| = |J^m(i)| \leq M(1+\lambda+\lambda^2+\cdots+\lambda^m) \leq M(1+\lambda+\lambda^2+\cdots+\lambda^m+\lambda^{m+1}).
$$

□

Data for discounted SMDP with known transition model $\mathbf{P}_a$, $\mathbf{R}_a$, and $\mathbf{T}_a$, the transition probability, reward, and time matrices for action a, respectively, used are:

$$
\mathbf{P}_1 = \begin{bmatrix} 0.6, 0.2, 0.1, 0.1 \\ 0.2, 0.3, 0.1, 0.4 \\ 0.1, 0.6, 0.2, 0.1 \\ 0.2, 0.4, 0.2, 0.2 \end{bmatrix}; \mathbf{P}_2 = \begin{bmatrix} 0.5, 0.1, 0.2, 0.2 \\ 0.2, 0.4, 0, 0.4 \\ 0.2, 0.5, 0.1, 0.2 \\ 0.6, 0.2, 0.1, 0.1 \end{bmatrix}; \mathbf{R}_1 = \begin{bmatrix} 6, -5, 0, 12 \\ 7, 120, 3, 1 \\ 5, 4, 12, 3 \\ 7, 48, 10, 10 \end{bmatrix};
$$

$$
\mathbf{R}_2 = \begin{bmatrix} 100, 17, 0, 10 \\ -14, 13, 0, 1 \\ 9, 40, 7, 12 \\ 12, 12, 10, 14 \end{bmatrix}; \mathbf{T}_1 = \begin{bmatrix} 1, 5, 2, 5 \\ 120, 60, 30, 30 \\ 2, 7, 12, 14 \\ 135, 55, 45, 30 \end{bmatrix}; \mathbf{T}_2 = \begin{bmatrix} 50, 75, 20, 12 \\ 7, 2, 7, 2 \\ 35, 65, 30, 20 \\ 50, 30, 50, 70 \end{bmatrix}
$$

Other constants in MAP were set to the following values: $\gamma = 0.1$, $m_{\max} = k_{\max} = n_{\max} = 100$, and $K = 5$.

Data for post-earthquake disaster response SMDP $\mathbf{PP} = [0.375, 0.2, 0.35, 0.075]$, where $PP(s)$ denotes the probability of going from the stable state to a primary state s. The following values were used for SP matrix:

$$
\mathbf{SP} = \begin{bmatrix} 0.1, 0.3, 0, 0.5, 0, 0, 0.1 \\ 0, 0.1, 0, 0.35, 0, 0.35, 0.2 \\ 0, 0, 0.1, 0, 0.5, 0.2, 0.2 \\ 0, 0, 0, 0.1, 0, 0, 0.9 \end{bmatrix},
$$

where $SP(s_1, s_2)$ denotes the probability of transition from a state, s_1, in the set of primary states to a state,s_2, in the union of the sets of the primary and secondary states. The following values were used for the hazard scores: $HS(1) = log(2)$, $HS(2) = log(4)$, $HS(3) = log(8)$, $HS(4) = log(6)$, $HS(5) = log(10)$, $HS(6) = log(12)$, and $HS(7) = log(14)$. Natural logarithms were used here for costs, as RL algorithms can suffer from computer overflow with large absolute values for costs, especially in discounted problems [24]. The response time for a state s is defined following [21]:

$RT_c(s) = \sum_{d \in \mathcal{I}(s)} RT(d)\phi$, where $\mathcal{I}(s)$ denotes the incidents present in state s, $RT(d)$ denotes the response time for an incident, d, and ϕ is a correction factor that equals 1 for a state that contains one incident, 1.2 for a state that contains two incidents, and 1.3 for a state that contains three incidents. The following values were used (all in hours): $RT(G) = 7 + \frac{5}{X}$, $RT(F) = 21 + \frac{15}{X}$, and $RT(BC) = 35 + \frac{25}{X}$, where $X = 1$ for the local agency and $X = 2$ for the federal agency. Finally, for the local and federal agencies, the travel time were $TRIA(0.5, 1, 1.5)$ and $TRIA(4, 5, 6)$, respectively, in *hrs*, where *TRIA* denotes the triangular distribution. Other constants in MAP were set as follows: $\gamma = 0.1$, $m_{\max} = k_{\max} = n_{\max} = 10,000$, and $K = 10$.

Steps in conservative approximate policy iteration (CAP) CAP uses Q-values instead of the future state-action values and bypasses model building. It has the same structure as MAP (see Algorithm 1 in main text) with the following exceptions: Step A (model-building) is skipped; Step B2 is replaced by evaluation of the Q-values, i.e., Eqn. (3) is replaced by: Update the Q-value for (i, a) as follows:

$$Q^{n+1}(i,a) \leftarrow (1-\beta^n)Q^n(i,a) + \beta^n \left[r(i,a,j) + \exp\left(-\gamma t(i,a,j)\right) J^{\infty}(j) \right];$$

Step C, policy is improved, i.e., for each $i \in \mathcal{S}$, select $\mu'(i) \in argmax_{a \in \mathcal{A}(i)} Q^{\infty}(i,a)$, where $Q^{\infty}(.,.)$ denotes the final Q-value obtained in Step B2.

Acknowledgements The paper has benefitted significantly from responding to suggestions from the Associate Editor and the reviewers. The authors thank the first reviewer for revisions leading to performance comparisons with CAP, the second reviewer for the Robbins-Monro analysis, and the third reviewer for the added numerical details in the first computational test.

References

1. Bertsekas, D.P.: Dynamic Programming and Optimal Control, 4th edn. Athena, Belmont (2012)
2. Bertsekas, D.P.: Feature-based aggregation and deep reinforcement learning: a survey and some new implementations. IEEE/CAA J Autom Sin **6**(1), 1–31 (2018)
3. Bertsekas, D.P.: Reinforcement Learning and Optimal Control. Athena, Belmont (2019)
4. Bertsekas, D.P.: Rollout, Policy Iteration, and Distributed Reinforcement Learning. Athena, Belmont (2021)
5. Bertsekas, D.P., Castanon, D.A.: Rollout algorithms for stochastic scheduling problems. J Heuristics **5**(1), 89–108 (1999)
6. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena, Belmont (1996)
7. Bradtke, S.J., Duff. M.: Reinforcement learning methods for continuous-time Markov decision problems. In Advances in Neural Information Processing Systems 7, MIT Press, Cambridge (1995)
8. Busoniu, L., Babuska, R., De Schutter, B., Ernst, D.: Reinforcement Learning and Dynamic Programming Using Function Approximators. CRC Press, Boca Raton (2010)
9. Cao, X.R.: Stochastic Learning and Optimization: A Sensitivity-Based View. Springer, Berlin (2007)
10. Chang, H.S., Fu, M.C., Hu, J., Marcus, S.I.: Simulation-Based Algorithms for Markov Decision Processes, 2nd edn. Springer, NY (2013)
11. Chang, H.S., Lee, H.-G., Fu, M.C., Marcus, S.: Evolutionary policy iteration for solving Markov decision processes. IEEE Trans. Autom. Control **50**(11), 1804–1808 (2005)
12. de la Torre, L.E., Dolinskaya, I.S., Smilowitz, K.R.: Disaster relief routing: integrating research and practice. Socioecon. Plann. Sci. **46**(1), 88–97 (2012)
13. FEMA. 2022-2016 Strategic plan. https://www.fema.gov/about/strategic-plan, (2023)

14. Fern, A., Yoon, S., Givan, R.: Approximate policy iteration with a policy language bias: solving relational Markov decision processes. J Artif Intell Res **25**, 75–118 (2006)
15. Fraioli, G., Gosavi, A., Sneed, L.H.: Strategic implications for civil infrastructure and logistical support systems in postearthquake disaster management: the case of St. Louis. IEEE Eng Manag Rev **49**(1), 165–173 (2020)
16. Ghosh, S., Gosavi, A.: A semi-Markov model for post-earthquake emergency response in a smart city. Control Theory Technol **15**(1), 13–25 (2017)
17. Gosavi, A.: A reinforcement learning algorithm based on policy iteration for average reward: empirical results with yield management and convergence analysis. Mach. Learn. **55**, 5–29 (2004)
18. Gosavi, A.: Boundedness of iterates in *Q*-learning. Syst. Control Lett. **55**, 347–349 (2006)
19. Gosavi, A.: On step-sizes, stochastic paths, and survival probabilities in reinforcement learning. In Proceedings of the winter simulation conference. IEEE, (2008)
20. Gosavi, A.: Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning, 2nd edn. Springer, NY (2015)
21. Gosavi, A., Fraioli, G., Sneed, L.H., Tasker, N.: Discrete-event-based simulation model for performance evaluation of post-earthquake restoration in a smart city. IEEE Trans. Eng. Manag. **67**(3), 582–592 (2019)
22. Hoffman, M., de Freitas, N.: Inference Strategies for Solving Semi-Markov Decision Processes. In Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions, pp. 82–96. IGI Global, Pennsylvania (2012)
23. Howard, R.: Dynamic Programming and Markov Processes. MIT Press, Cambridge (1960)
24. Matsui, T., Goto, T., Izumi, K., Chen, Y.: Compound reinforcement learning: theory and an application to finance. In European workshop on reinforcement learning, pp. 321–332. Springer (2011)
25. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.C., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
26. Puterman, M.L.: Markov Decision Processes. Wiley, NY (1994)
27. Puterman, M.L., Shin, M.C.: Modified policy iteration for discounted Markov decision problems. Manage. Sci. **24**, 1127–1137 (1978)
28. Scherrer, B.: Performance bounds for λ policy iteration and application to the game of tetris. J. Mach. Learn. Res. **14**(4), (2013)
29. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
30. Sutton, R., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. The MIT Press, Cambridge (2018)
31. Tadepalli, P., Ok, D.: Model-based average reward reinforcement learning algorithms. Artif. Intell. **100**, 177–224 (1998)
32. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double *Q*-learning. In Proceedings of AAAI conference on Art. Intel., vol. 30, No. (1) (2016)
33. van Nunen, J.A.E.E.: A set of successive approximation methods for discounted Markovian decision problems. Z. Operat. Res. **20**, 203–208 (1976)
34. van Seijen, H., Whiteson, S., van Hasselt, H., Wiering, M.: Exploiting best-match equations for efficient reinforcement learning. J. Mach. Learn. Res. **12**, 2045–2094 (2011)
35. Yoshida, W., Ishii, S.: Model-based reinforcement learning: a computational model and an fMRI study. Neurocomputing **63**, 253–269 (2005)

Authors and Affiliations

A. Gosavi[1] · L. H. Sneed[2] · L. A. Spearing[3]

✉ A. Gosavi
gosavia@mst.edu

L. H. Sneed
lhsneed@uic.edu

L. A. Spearing
spearing@uic.edu

[1] Missouri University of Science and Technology, 210 EMAN Bldg, Rolla, MO 65409, USA

[2] University of Illinois, Chicago, EIB 218, 929 West Taylor Street, Chicago, IL 60607, USA

[3] University of Illinois, Chicago, ERF 3069, 842 West Taylor Street, Chicago, IL 60607, USA