A Real-Time Computational Learning Model for Sequential Decision-Making Problems Under Uncertainty

Andreas A. Malikopoulos
e-mail: amalik@umich.edu

Panos Y. Papalambros
e-mail: pyp@umich.edu

Dennis N. Assanis
e-mail: assanis@umich.edu

Department of Mechanical Engineering,
University of Michigan,
Ann Arbor, MI 48109

Modeling dynamic systems incurring stochastic disturbances for deriving a control policy is a ubiquitous task in engineering. However, in some instances obtaining a model of a system may be impractical or impossible. Alternative approaches have been developed using a simulation-based stochastic framework, in which the system interacts with its environment in real time and obtains information that can be processed to produce an optimal control policy. In this context, the problem of developing a policy for controlling the system's behavior is formulated as a sequential decision-making problem under uncertainty. This paper considers the problem of deriving a control policy for a dynamic system with unknown dynamics in real time, formulated as a sequential decision-making under uncertainty. The evolution of the system is modeled as a controlled Markov chain. A new state-space representation model and a learning mechanism are proposed that can be used to improve system performance over time. The major difference between the existing methods and the proposed learning model is that the latter utilizes an evaluation function, which considers the expected cost that can be achieved by state transitions forward in time. The model allows decision-making based on gradually enhanced knowledge of system response as it transitions from one state to another, in conjunction with actions taken at each state. The proposed model is demonstrated on the single cart-pole balancing problem and a vehicle cruise-control problem. [DOI: 10.1115/1.3117200]

1 Introduction

Deriving a control policy for dynamic systems is an off-line process in which various methods from control theory are utilized. These methods aim to determine the policy that satisfies the system's physical constraints while optimizing specific performance criteria. A challenging task in this process is to derive a mathematical model of the system's dynamics that can adequately predict the response of the physical system to all anticipated inputs. Exact modeling of complex engineering systems, however, may be infeasible or expensive. Viable alternative methods have been developed enabling the real-time implementation of control policies for systems when an accurate model is not available. In this framework, the system interacts with its environment and obtains information enabling it to improve its future performance by means of a cost (or reward) associated with control actions taken. This interaction portrays the learning process conveyed by the progressive enhancement of the system's "knowledge" regarding the control policy that minimizes (maximizes) the accumulated cost (reward) with respect to the system's operating point (state). The environment is assumed to be nondeterministic; namely, taking the same action in the same state on two different stages, the system may transit to a different state and receive a dissimilar cost (reward) in the subsequent stage. Consequently, the problem of developing a policy for controlling the system's behavior is formulated as a sequential decision-making problem under uncertainty.

Dynamic programming (DP) has been widely employed as the principal method for analysis of sequential decision-making problems [1]. Algorithms, such as value and policy iteration, have been extensively utilized in solving deterministic and stochastic optimal control problems, Markov and semi-Markov decision problems, min-max control problems, and sequential games. However, the computational complexity of these algorithms in some occasions may be prohibitive and can grow intractably with the size of the problem and its related data, referred to as the DP "curse of dimensionality" [2]. In addition, DP algorithms require the realization of the conditional probabilities of state transitions and the associated cost, implying a priori knowledge of the system dynamics.

Alternative approaches for solving sequential decision-making problems under uncertainty have been primarily developed in the field of reinforcement learning (RL) [3,4]. RL has aimed to provide simulation-based algorithms, founded on DP, for learning control policies of complex systems, where exact modeling is infeasible or expensive [5]. A major influence on research leading to current RL algorithms has been Samuel's method [6,7], used to modify a heuristic evaluation function for deriving optimal board positions in the game of checkers. In this algorithm, Samuel represented the evaluation function as a weighted sum of numerical features and adjusted the weights based on an error derived from comparing evaluations of current and predicted board positions. This approach was refined and extended by Sutton [8,9] to introduce a class of incremental learning algorithms, temporal difference (TD). TD algorithms are specialized for deriving policies for incompletely known systems, using past experience to predict their future behavior. Watkins [10,11] extended Sutton's TD algorithms and developed an algorithm for systems to learn how to act optimally in controlled Markov domains by explicitly utilizing the theory of DP. A strong condition implicit in the convergence of Q-learning to a control policy is that the sequence of stages that forms the basis of learning must include an infinite number of stages for each initial state and action. However, Q-learning is considered the most popular and efficient model-free learning al-

Contributed by the Dynamic Systems, Measurement, and Control Division of ASME for publication in the JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL. Manuscript received March 18, 2008; final manuscript received February 4, 2009 published online May 20, 2009. Assoc. Editor: Santosh Devasia. Paper presented at the 2007 ASME International Mechanical Engineering Congress (IMECE2007), Seattle, WA, November 10–16, 2007.

gorithm in deriving control policies in Markov domains [12]. Schwartz [13] explored the potential of adapting Q-learning to an average-reward framework with his R-learning algorithm; Bertsekas and Tsitsiklis [3] presented a similar to Q-learning average-reward algorithm. Mahadevan [14] surveyed reinforcement-learning average-reward algorithms and showed that these algorithms do not always produce bias-optimal control policies.

Although many of these algorithms are eventually guaranteed to find suboptimal policies in sequential decision-making problems under uncertainty, their use of the accumulated data acquired over the learning process is inefficient, and they require a significant amount of experience to achieve good performance [12]. This requirement arises due to the formation of these algorithms in deriving control policies without learning the system dynamics *en route*, that is, they do not solve the system identification problem simultaneously.

Algorithms for computing control policies by learning the models are especially appealing in applications in which real-world experience is considered expensive. Sutton's Dyna architecture [15,16] exploits strategies, which simultaneously utilize experience in building the model and adjust the derived policy. Prioritized sweeping [17] and Queue-Dyna [18] are similar methods concentrating on the interesting subspaces of the state-action space. Barto et al. [19] developed another method, called real-time dynamic programming (RTDP), referring to the cases in which concurrently executing DP and control processes influence one another. RTDP focuses the computational effort on the state-subspace that the system is most likely to occupy. However, these methods are specific to problems in which the system needs to achieve particular "goal" states.

In this paper, we consider the problem of deriving a control policy for a dynamic system with unknown dynamics formulated as a sequential decision-making problem under uncertainty. The goal is to adequately control an unknown system, when its performance can be completely measured, by learning the system dynamics in real time. A state-space representation model and a learning mechanism are proposed that can be used to improve system performance over time for its entire operating domain [20]. The major difference between the existing methods and the proposed learning model is that the latter utilizes an evaluation function, which considers the expected cost that can be achieved by state transitions forward in time. The model accumulates gradually enhanced knowledge of system response as it transitions from one state to another, in conjunction with actions taken at each state.

The remainder of the paper proceeds as follows: Section 2 provides the mathematical framework of sequential decision-making problems under uncertainty. Section 3 introduces the predictive optimal decision-making (POD) model and the learning mechanism that can be utilized to improve system performance over time. The proposed model is demonstrated on the single cart-pole balancing problem in Sec. 4 and on a vehicle cruise-control problem in Sec. 5. Conclusions are presented in Sec. 6.

2 Problem Formulation

A large class of sequential decision-making problems under uncertainty can be modeled as a Markov decision process (MDP) [21]. MDP provides the mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of the decision maker. Decisions are made at points of time referred to as decision epochs, and the time domain can be either discrete or continuous. The focus of this paper is on discrete-time decision-making problems.

The Markov decision process model consists of five elements: (a) a decision maker (controller), (b) system states, (c) control actions, (d) the transition probability matrix, and (e) the transition cost matrix. In this framework, the decision maker is faced with the problem of influencing system behavior as it evolves over time, by selecting control actions. The objective of the decision maker is to select the control policy, which causes the system to perform optimally with respect to some predetermined optimization criterion. Control actions must anticipate costs associated with future system states-actions. The system's operation is treated as a controlled stochastic process $\{s_k, k=0,1,2,\ldots\}$ that visits the states of a finite state-space $\mathcal{S}$.

At each decision epoch, the system occupies a state $s_k=i$ from the finite set of all possible system states $\mathcal{S}$.

$$\mathcal{S}=\{1,2,\ldots,N\}, \quad N \in \mathbb{N} \tag{1}$$

In this state $s_k=i \in \mathcal{S}$, the decision maker has a set of allowable actions available, $a_k \in A(s_k), A(s_k) \subseteq \mathcal{A}$, where $\mathcal{A}$ is the finite action space.

$$\mathcal{A}=\cup_{s_k \in \mathcal{S}} A(s_k) \tag{2}$$

The decision-making process occurs at each of a sequence of decision epochs $k=0,1,2,\ldots,M, M \in \mathbb{N}$. At each epoch, the decision maker observes a system's state $s_k=i \in \mathcal{S}$, and executes an action $a_k \in A(s_k)$, from the feasible set of actions $A(s_k) \subseteq \mathcal{A}$ at this state. At the next epoch, the system transits to the state $s_{k+1}=j \in \mathcal{S}$ imposed by the conditional probabilities $p(s_{k+1}=j|s_k=i,a_k)$, designated by the transition probability matrix $\mathbf{P}(\cdot,\cdot)$. The conditional probabilities of $\mathbf{P}(\cdot,\cdot)$, $p: \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$, satisfy the following constraint:

$$\sum_{j=1}^{N} p(s_{k+1}=j|s_k=i,a_k)=1 \tag{3}$$

Following this state transition, the decision maker receives a cost associated with the action a_k, $R(s_{k+1}=j|s_k=i,a_k)$, and $R: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ as imposed by the transition cost matrix $\mathbf{R}(\cdot,\cdot)$. The states of a MDP possess the Markov property, stating that the conditional probability distribution of future states of the process depends only on the current state and not on any past states, i.e., it is conditionally independent of the past states (the path of the process) given the present state. Mathematically, the Markov property states that

$$p(s_{k+1}|s_k,s_{k-1},\ldots,s_0)=p(s_{k+1}|s_k) \tag{4}$$

We seek a finite sequence of functions $\pi=\{\mu_0,\mu_1,\ldots,\mu_{M-1}\}$, defined as a control policy, which minimizes the total cost over M decision epochs. The functions μ_k specify the control $a_k=\mu(s_k)$ that will be chosen when at kth decision epoch the state is s_k. Consequently, the total cost corresponding to a policy $\pi=\{\mu_0,\mu_1,\ldots,\mu_{M-1}\}$ and initial state s_0 is given by

$$J^{\pi}(s_0)=\sum_{k=0}^{M-1} R(s_{k+1}|s_k,\mu(s_k)) \tag{5}$$

At each initial state s_0 and π, there is a corresponding sequence of control actions $a_0,a_1,\ldots,a_{M-1}$, where $a_k=\mu(s_k)$. The accumulated cost $J_{\pi}(s_0)$ is a random variable since $\{s_k, k \geq 0\}$ and $\{a_k, k \geq 0\}$ are random variables. Hence the expected accumulated cost of a Markov policy is given by

$$J^{\pi}(s_0)=\underset{\substack{s_k \in \mathcal{S} \\ \mu_k \in A(s_k)}}{E}\left\{\sum_{k=0}^{M-1} R_k(s_{k+1}=j|s_k=i,\mu_k(s_k))\right\} \tag{6}$$

$J^{\pi}(s_0)$ can be readily evaluated in terms of the transition probability matrix, namely,

$$J^{\pi}(s_0)=\sum_{k=1}^{M-1}\sum_{j \in S} P(s_{k+1}=j|s_k=i,\mu_k(s_k)) \cdot R(s_{k+1}=j|s_k=i,\mu_k(s_k)) \tag{7}$$

Consequently, the optimal policy $\pi^*=\{\mu_0^*,\mu_1^*,\ldots,\mu_M^*\}$ for the M-decision epoch sequence is

$$\pi^* = \arg\min_{\pi \in \Pi} J^{\pi^*}(s_0) \tag{8}$$

In this paper, we consider the problem of deriving a control policy for a dynamic system with unknown dynamics, namely, no prior information regarding the values of the transition probability and cost matrices is provided. The problem is formulated as a Markov decision process. The goal is to adequately control an unknown system when its performance can be completely measured in real time. This problem involves two major subproblems: (a) the system identification problem and (b) the stochastic control problem. The first is exploitation of the information acquired from the system output to identify its behavior, that is, how a state representation can be built by observing the system's state transitions. The second is the assessment of the system output with respect to alternative control policies, and selecting those that optimize specified performance criteria.

The POD computational model proposed in this paper is suitable for solving the system identification problem in real time. To address the stochastic control problem, the lookahead control algorithm POSCA [22] is employed that assigns at each state the control actions that minimize the transition cost of the next two decision epochs. The accumulated cost resulting from the control policy of POSCA is bounded by the accumulated cost of the optimal minimax control policy of DP with probability 1 [23]. POSCA is especially appealing for real-time implementation when the time between decision epochs is small. In this situation, the controller needs to select control actions quickly and there is not enough time to search for an optimal policy for a relatively distant future.

3 Predictive Optimal Decision-Making Model

The POD computational model consists of a new state-space system representation that accumulates gradually enhanced knowledge of the system's transition from each state to another in conjunction with actions taken for each state. This knowledge is expressed in terms of an expected evaluation function associated with each state. The major difference between the proposed learning method and the existing RL methods is that the latter consists of evaluation functions attempting to successively approximate Eq. (6). These evaluation functions assign to each state the total cost expected to accumulate over time starting from a given state when a policy π is employed. The proposed learning model, on the contrary, utilizes an evaluation function, which considers the expected cost that can be achieved by state transitions forward in time. This approach is especially appealing to learning engineering systems in which the initial state is not fixed [22,24,25], and recursive updates of the evaluation functions to approximate Eq. (6) demand a huge number of iterations to achieve the desired system performance.

3.1 POD State-Space Representation. In our analysis, we consider dynamic systems incurring stochastic disturbances in stationary environment, that is, we assume that the Markov chain is homogeneous. The proposed state-space representation defines the POD domain $\tilde{\mathcal{S}}$, which is implemented by a mapping $\mathcal{H}$ from the Cartesian product of the finite state space and action space of the Markov chain $\{s_k, k \geq 0\}$.

$$\mathcal{H}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \tilde{\mathcal{S}} \tag{9}$$

where $\mathcal{S}=\{1,2,\ldots,N\}, N \in \mathbb{N}$ denotes the Markov state space, and $\mathcal{A}=\cup_{s_k \in \mathcal{S}} A(s_k), \forall\, s_k=i \in \mathcal{S}$ stands for the finite action space. Each state of the POD domain represents a Markov state transition from $s_k=i \in \mathcal{S}$ to $s_{k+1}=j \in \mathcal{S}$ for all $k \geq 0$, as illustrated in Fig. 1, that is,

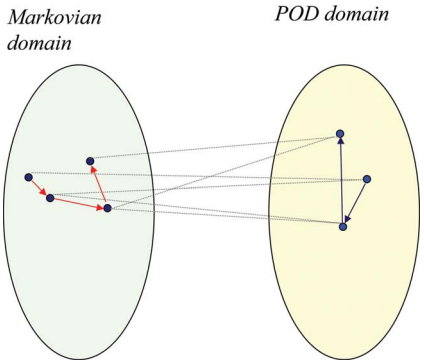

Fig. 1 Construction of the POD domain

$$\tilde{\mathcal{S}} := \left\{ \tilde{s}_{k+1}^{ij} | \tilde{s}_{k+1}^{ij} \equiv s_k = i \underset{\mu(s_k) \in A(s_k)}{\rightarrow} s_{k+1} = j, \sum_{j=1}^{N} p(s_{k+1}=j | s_k = i, a_k) = 1, N = |\mathcal{S}| \right\}$$

$$\forall i, j \in \mathcal{S}, \quad \forall\, \mu(s_k) \in A(s_k) \tag{10}$$

DEFINITION 3.1. The mapping $\mathcal{H}$ generates an indexed family of subsets $\tilde{\mathcal{S}}_i$ for each Markov state $s_k=i \in \mathcal{S}$, defined as predictive representation nodes (PRNs). Each PRN is constituted by the set of POD states $\tilde{s}_{k+1}^{ij} \in \tilde{\mathcal{S}}_i$ representing the state transitions from the state $s_k=i \in \mathcal{S}$ to all other Markov states

$$\tilde{\mathcal{S}}_i := \{ \tilde{s}_{k+1}^{ij} | s_k = i \underset{\mu(s_k) \in A(s_k)}{\rightarrow} s_{k+1} = j, \forall\, j \in \mathcal{S} \} \tag{11}$$

PRNs partition the POD domain insofar as the POD underlying structure captures the state transitions in the Markov domain as depicted in Fig. 2, namely,

$$\tilde{\mathcal{S}} = \cup_{\tilde{s}_k^{ij} \in \tilde{\mathcal{S}}_i} \tilde{\mathcal{S}}_i \tag{12}$$

with

$$\cap_{\tilde{s}_k^{ij} \in \tilde{\mathcal{S}}_i} \tilde{\mathcal{S}}_i = \emptyset \tag{13}$$

PRNs, constituting the fundamental aspect of the POD state representation, provide an assessment of the Markov state transitions along with the actions executed at each state. This assessment aims to establish a necessary embedded property of the new state representation so as to consider the potential transitions that can occur in subsequent decision epochs. The assessment is expressed by means of the PRN value $\bar{R}_i(\tilde{s}_{k+1}^{ij} | \mu(s_i))$, which accounts for the minimum expected cost that can be achieved by transitions occurring inside a PRN.

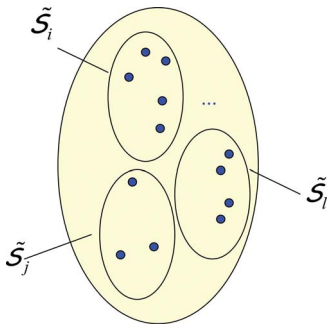

Fig. 2 Partition of POD through the PRNs

DEFINITION 3.2. The PRN value $\bar{R}_i((\tilde{s}_{k+1}^{ij}|\mu(s_i))$ is defined as

$$\bar{R}_i(\tilde{s}_{k+1}^{ij}|\mu(s_k=i)) := \min_{\mu(s_k)\in A} E_{s_{k+1}\in\mathcal{S}}\left\{\sum_{j=1}^{N} R(s_{k+1}=j|s_k=i,\mu(s_k))\right\} \tag{14}$$

$$\forall \tilde{s}_{k+1}^{ij} \in \tilde{\mathcal{S}},\ \forall\, i,j \in \mathcal{S},\ \forall\ \mu(s_k) \in A(s_k) \quad \text{and} \quad N=|\mathcal{S}|$$

The PRN value is exploited by POD state representation as an evaluation metric to estimate the subsequent Markov state transitions. The estimation property is founded on the assessment of POD states by means of an expected evaluation function $R_{\mathrm{PRN}}^{i}(\tilde{s}_{k+1}^{ij},\mu(s_k))$ defined as

$$R_{\mathrm{PRN}}^{i}(\tilde{s}_{k+1}^{ij},\mu(s_k)) := E_{s_{k+1}\in\mathcal{S}}\{R(s_{k+1}=j|s_k=i,\mu(s_k))\} + \bar{R}_j(\tilde{s}_{k+2}^{jm}|\mu(s_{k+1})) \tag{15}$$

$$\forall \tilde{s}_{k+2}^{jm} \in \tilde{\mathcal{S}},\ \forall\, i,j,m \in \mathcal{S},\ \forall\ \mu(s_k),\ \in A(s_k),\ \forall\ \mu(s_{k+1}) \in A(s_{k+1})$$

Consequently, employing the POD evaluation function through Eq. (15), each POD state $\tilde{s}_{k+1}^{ij} \in \tilde{\mathcal{S}}_i$ is comprised of an overall cost corresponding to (a) the expected cost of transiting from state $s_k=i$ to $s_{k+1}=j$ (implying also the transition from the PRN $\tilde{\mathcal{S}}_i$ to $\tilde{\mathcal{S}}_j$) and (b) the minimum expected cost when transiting from $s_{k+1}=j$ to any other Markov state at $k+2$ (transition occurring into $\tilde{\mathcal{S}}_j$).

3.2 Real-Time Self-Learning System Identification. While the system interacts with its environment, the POD model learns the system dynamics in terms of the Markov state transitions. The POD state representation attempts to provide a process in realizing the sequences of state transitions that occurred in the Markov domain, as infused in PRNs. The different sequences of the Markov state transitions are captured by the POD states and evaluated through the expected evaluation functions given in Eq. (15). Consequently, the lowest value of the expected evaluation function at each POD state essentially estimates the subsequent Markov state transitions with respect to the actions taken.

The learning performance is closely related to the exploration-exploitation strategy of the action space. More precisely, the decision maker has to exploit what is already known regarding the correlation involving the admissible state-action pairs that minimize the costs and also to explore those actions that have not yet been tried for these pairs to assess whether these actions may result in lower costs. A balance between an exhaustive exploration of the environment and the exploitation of the learned policy is fundamental to reach nearly optimal solutions in a few decision epochs and, thus, to enhance the learning performance. This exploration-exploitation dilemma has been extensively reported in the literature. Iwata et al. [26] proposed a model-based learning method extending Q-learning and introducing two separated functions based on statistics and on information by applying exploration and exploitation strategies. Ishii et al. [27] developed a model-based reinforcement- learning method utilizing a balance parameter, controlled through a variation of action rewards and perception of environmental change. Chan-Geon and Sung-Bong [28] proposed an exploration-exploitation policy in Q-learning consisting of an auxiliary Markov process and the original Markov process. Miyazaki and Yamamura [29] developed a unified learning system realizing the tradeoff between exploration and exploitation. Hernandez-Aguirre et al. [30] analyzed the problem of exploration-exploitation in the context of the approximately correct framework and studied whether it is possible to set bounds on the complexity of the exploration needed to achieve a fixed approximation error over the action value function with a given probability.

An exhaustive exploration of the environment is necessary to evade premature convergence on a suboptimal solution even if this may result in both sacrificing the system's performance in the short run and increasing the learning time. In our case it is assumed that, for any state $s_k=i \in \mathcal{S}$, all actions of the feasible action set $\mu(s_k) \in A(s_k)$ are selected by the decision maker at least once. At the early decision epochs and until full exploration of the action set $A(s_k)$ occurs, the mapping from the states to probabilities of selecting the actions is constant; namely, the actions for each state are selected randomly with the same probability.

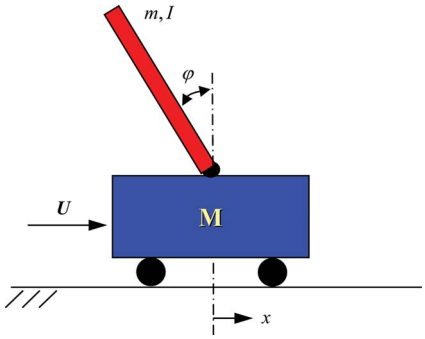

Fig. 3 The inverted pendulum

$$p(\mu(s_k)|s_k) = \frac{1}{|A(s_k)|}, \quad \forall\ \mu(s_k) \in A(s_k), \quad \forall\ s_k \in \mathcal{S} \tag{16}$$

The POD state representation attempts to provide an efficient process in realizing the state transitions that occurred in the Markov domain. The different sequences of the state transitions are captured by the POD states and evaluated through the expected evaluation functions given in Eq. (15). Consequently, the lowest value of the expected evaluation function at each PRN essentially predicts the Markov state transition that will occur in the future. As the process is stochastic, however, it is still necessary for the controller to build a decision-making learning mechanism of how to select actions.

The idea behind for such a mechanism is founded on the theory of stochastic control problems with unknown disturbance distribution, also known as games against nature. The decision-making mechanism is modeled as a stochastic game between the decision maker (controller) and an "opponent" (environment). Each POD state $\tilde{s}_{k+1}^{ij} \in \tilde{\mathcal{S}}_i$ corresponds to a completed game that started at the Markov state $s_k=i \in \mathcal{S}$ and ended up at $s_{k+1}=j \in \mathcal{S}$. At state $s_k=i$, the decision maker has a set of strategies (control actions) $\mu(s_k) \in A(s_k)$ available to play. Similarly, the environment's set of strategies are the Markov states $\mathcal{S}=\{1,2,\ldots,N\}, N \in \mathbb{N}$. During the learning process of the POD model, this game has been played insofar as the decision maker forms a belief about the environment's behavior by fully exploring all available strategies, $\mu(s_k) \in A(s_k)$. This property arises when the state representation converges to the stationary distribution of the Markov chain [23,31]. Consequently, at state $s_k=i \in \mathcal{S}$, the decision maker can select those control actions by means of the PRN expected evaluation functions, $R_{\mathrm{PRN}}^{i}(\tilde{s}_{k+1}^{ij},\mu(s_k))$. However, to handle the uncertainty of this prediction, the decision maker seeks a policy $\pi^* \in \Pi$ through POSCA, which guarantees the best performance in the worst possible situation, namely,

$$\pi^*(s_k) = \arg\min_{\bar{\mu}_k(s_k)\in A(s_k)} \{\max_{s_{k+1}\in\mathcal{S}} [R_{\mathrm{PRN}}^{i}(\tilde{s}_{k+1}^{ij},\mu(s_k))]\} \tag{17}$$

4 Application I: The Single Cart-Pole Balancing Problem

The overall performance of the POD real-time learning model is evaluated on the basis of its application to the inverted pendulum balancing problem. The inverted pendulum involves a pendulum hinged to the top of a wheeled cart as illustrated in Fig. 3. The

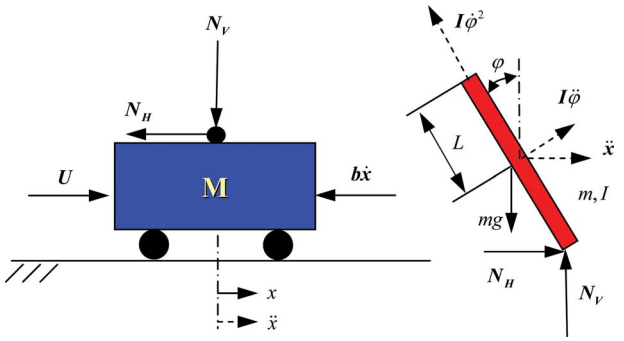

Fig. 4 Free body diagram of the system

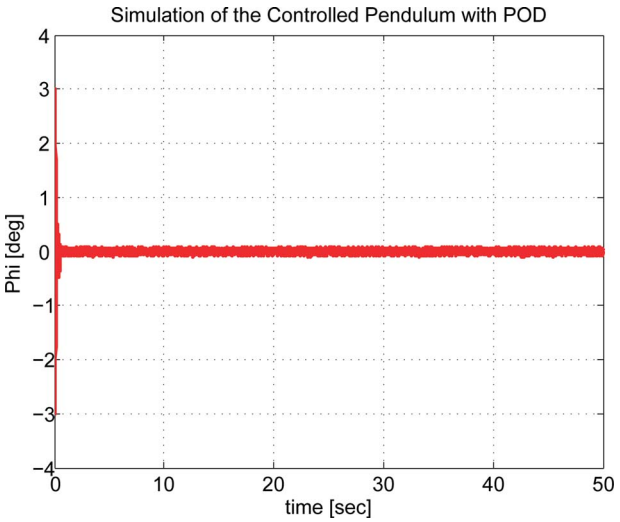

Fig. 5 Simulation of the system after learning the balance control policy with POD for different initial conditions

objective of POD is to balance the pendulum having no prior knowledge about the system dynamics utilizing only real-time measurements.

Realizing the balance control policy of a single inverted pendulum without a priori knowledge of the system's model has been extensively reported in the literature for the evaluation of learning algorithms. Anderson [32] implemented a neural network reinforcement-learning method to generate successful action sequences. Two neural networks having a similar structure were employed to learn two functions: (a) an action function mapping the current state into control actions, and (b) an evaluation action mapping the current state into an evaluation of that state. These two networks were trained utilizing reinforcement learning by evaluating the performance of the network and were compared with real-time measurements. Williams and Matsuoka [33] proposed a learning architecture for training a neural network controller to provide the appropriate control force to balance the inverted pendulum. One network for the identification of the plant dynamics and one for the controller were employed. Zhidong et al. [34] implemented a "neural-fuzzy BOXES" control system by neural networks and utilized reinforcement learning for the training. Jeen-Shing and McLaren [35] proposed a defuzzification method incorporating a genetic algorithm to learn the defuzzification factors. Mustapha and Lachiver [36] developed an actor-critic reinforcement-learning algorithm represented by two adaptive neural-fuzzy systems. Si and Wang [37] proposed a generic online learning control system similar to Anderson's utilizing neural networks and evaluated it through its application to both a single and double cart-pole balancing problems. The system utilizes two neural networks and employs the action-dependent heuristic dynamic programming to adapt the weights of the networks.

In the implementation of the POD on the single inverted pendulum presented here, two major variations are considered: (a) a single look-up table-based representation is employed for the controller to develop the mapping from the system's Markov states to optimal actions, and (b) two system's state variables are selected to represent the Markov state. The latter introduces uncertainty and thus a conditional probability distribution associating the state transitions with respect to the actions taken. Consequently, the POD method is evaluated in deriving the optimal policy (balance control policy) in a sequential decision-making problem under uncertainty.

The governing equations, derived from the free body diagram of the system, shown in Fig. 4, are as follows:

$$(M+m)\ddot{x} + b\dot{x} + mL\ddot{\varphi}\cos\varphi - mL\dot{\varphi}^2\sin\varphi = U \tag{18}$$

$$mL\ddot{x}\cos\varphi + (I+mL^2)\ddot{\varphi} + mgL\sin\varphi = 0 \tag{19}$$

where $M=0.5$ kg, $m=0.2$ kg, $b=0.1$ N s/m, $I=0.006$ kg m^2, $g=9.81$ m/s^2, and $L=0.3$ m.

The goal of the learning controller is to realize in real time the force U of a fixed magnitude to be applied either to the right or the left direction so that the pendulum stands balanced when released from any angle φ between 3 deg and -3 deg. The system is simulated by numerically solving the nonlinear differential equations (18) and (19) employing the explicit Runge–Kutta method with a time step of $\tau=0.02$ s. The simulation is conducted by observing the system's states and executing actions (control force U) with a sample rate $k=0.02$ s (50 Hz). This sample rate defines a sequence of decision-making epochs, $k=0,1,2,\ldots,M, M\in\mathbb{N}$.

The system is fully specified by four state variables: (a) the position of the cart on the track x, (b) the cart velocity $\dot{x}$, (c) the pendulum's angle with respect to the vertical position φ, and (d) the angular velocity $\dot{\varphi}$. However, to incorporate uncertainty, the Markov states are selected to be only the pair of the pendulum's angle and angular velocity, namely, the finite state-space $\mathcal{S}$ in Eq. (1) is defined as

$$\mathcal{S} = \{(\varphi, \dot{\varphi})\} \tag{20}$$

Consequently, at state $s_k=i\in\mathcal{S}$ and executing a control force value $U(s_k)$, the system will end up at state $s_{k+1}=j\in\mathcal{S}$ with a conditional probability $p(s_{k+1}=j|s_k=i,U(s_k))$. The control force $U(s_k)$ selects values from the finite set $\mathcal{A}$, defined as

$$\mathcal{A} = A(s_k) = [-3N, 3N] \tag{21}$$

The decision-making process occurs at each of the sequence of epochs $k=0,1,2,\ldots,M, M\in\mathbb{N}$. At each decision epoch, the learning controller observes the system's state $s_k\in\mathcal{S}$ and executes a control force value $U(s_k)\in A(s_k)$. At the next decision epoch, the system transits to another state $s_{k+1}\in\mathcal{S}$ imposed by the conditional probabilities $p(s_{k+1}|s_k,U(s_k))$ and receives a numerical cost (the pendulum's angle φ).

The inverted pendulum is simulated repeatedly for different initial angles φ between 3 deg and -3 deg. The simulation lasts for 50 s and each complete simulation defines one iteration. If at any instant during the simulation, the pendulum's angle φ becomes greater than 3 deg or less than -3 deg, this constitutes a failure, denoted by stating that there was one iteration associated with a failure. If, however, no failure occurs during the simulation, this is denoted by stating that there was one iteration associated with no failure.

After completing the learning process, the controller employing the POD learning model realizes the balance control policy of the pendulum, as illustrated in Fig. 5. In some instances, however, the system's response demonstrates some overshoots or delays during the transient period, shown in Fig. 6. This can be handled by a denser parametrization of the state-space or adding a penalty in

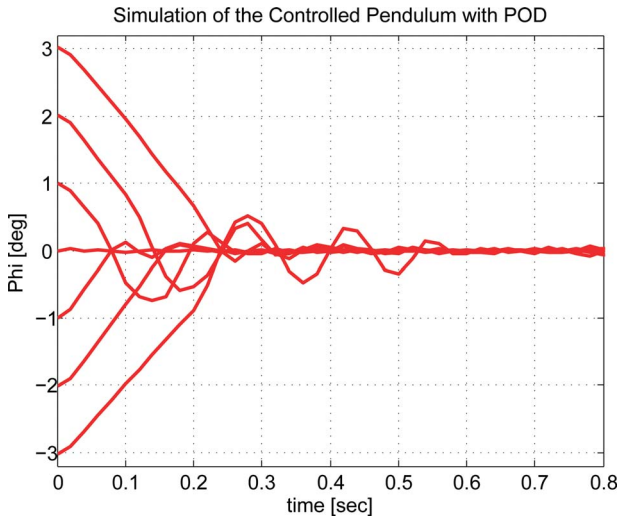

Fig. 6 Simulation of the system after learning the balance control policy with POD for different initial conditions (zoom in)

long transient responses. The efficiency of the POD learning method in deriving the optimal balance control policy that stabilizes the system is illustrated in Fig. 7. It is noted that after POD realizes the optimal policy in 749 failures, as the number of iterations continues no further failures occur.

5 Application II: Vehicle Cruise Control

The POD model is applied here to a vehicle cruise-control problem. Cruise control automatically regulates the vehicle's longitudinal velocity by suitably adjusting the gas pedal position. A vehicle cruise-control system is activated by the driver who desires to maintain a constant speed in long highway driving. The driver activates the cruise controller while driving at a particular speed, which is then recorded as the desired or set-point speed to be maintained by the controller. The main goal in designing a cruise-control algorithm is to maintain vehicle speed smoothly but accurately, even under large variation of plant parameters (e.g., the vehicle's varying mass in terms of the number of passengers) and road grade. In the case of passenger cars, however, vehicle mass may change noticeably but is within a small range. Therefore, powertrain behavior might not vary significantly.

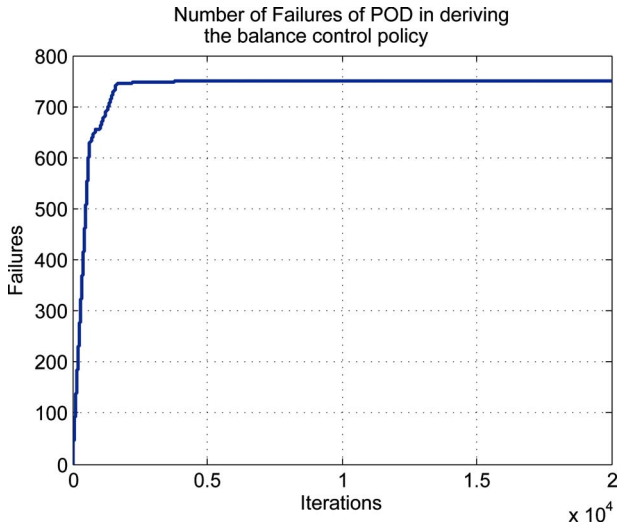

Fig. 7 Number of failures until POD derives the balance control policy

The objective of the POD learning cruise controller is to realize in real time the control policy (gas pedal position) that maintains the vehicle speed as set by the driver under a great range of different road grades. Implementing learning vehicle cruise controllers has been addressed previously employing learning and active control approaches. Zhang et al. [38] implemented learning control based on pattern recognition to regulate in real time the parameters of a proportional, integral, and derivative (PID) cruise controller. Shahdi and Shouraki [39] proposed an active learning method to extract the driver's behavior and to derive control rules for a cruise-control system. However, no attempt has been reported in implementing a learning automotive vehicle cruise controller utilizing the principle of reinforcement learning, i.e., enabling the controller to improve its performance over time by learning from its own failures through a reinforcement signal from the external environment, and thus, attempting to improve future performance.

The software package ENDYNA by TESIS [40], suitable for real-time simulation of internal combustion engines, is used to evaluate the performance of the POD learning cruise controller. The software simulates the longitudinal vehicle dynamics with a highly variable drive train including the modules of starter, brake, clutch, converter, and transmission. In the driving mode the engine is operated by means of the usual vehicle control elements just as a driver would do. In addition, a mechanical parking lock and the uphill grade can be set. The driver model is designed to operate the vehicle at given speed profiles (driving cycles). It actuates the starter, accelerator, clutch, and brake pedals according to the profile specification, and also shifts gears. In this example, an existing vehicle model is selected representing a midsize passenger car carrying a four-cylinder 1.9 l turbocharged diesel engine.

When activated, the learning cruise controller bypasses the driver model and takes over the vehicle's cruising. The Markov states are defined to be the pair of the transmission gear and the difference between the desired and actual vehicle speeds ΔV, namely,

$$\mathcal{S} = \{(\text{gear}, \Delta V)\} \tag{22}$$

The control actions correspond to the gas pedal position and can take values from the feasible set $\mathcal{A}$, defined as

$$\mathcal{A} = A(s_k) = [0, 0.7] \tag{23}$$

To incorporate uncertainty the vehicle is simulated in a great range of different road grades from 0 deg to 10 deg. Consequently, at state $s_k \in \mathcal{S}$ and executing a control action (gas pedal position), the system transits to another state $s_{k+1} \in \mathcal{S}$ with a conditional probability $p(s_{k+1}|s_k, a_k(s_k))$, since the acceleration capability of a vehicle varies at different road grades. As a consequence of this state transition the system receives a numerical cost (difference between the desired and actual vehicle speeds).

After completing four simulations of each road grade, the POD cruise controller realizes the control policy (gas pedal position) to maintain the vehicle's speed at the desired set point. The vehicle model was initiated from zero speed. The driver model, following the driving cycle, accelerated the vehicle up to 40 mph and at 10 s the POD cruise controller was activated. The desired and actual vehicle speeds for three different road grades as well as the gas pedal rates of the POD controller are illustrated in Fig. 8. The small discrepancy between the desired and actual vehicle speeds before the cruise-controller activation is due to the steady-state error of the driver's model. However, since the desired driving cycle set the vehicle's speed to be at 40 mph, when the POD cruise controller is activated helps to correct this error and, afterwards, maintains the vehicle's actual speed at the set point. The accelerator pedal position is at different values because in the case of road grades 2 deg and 6 deg the selected transmission gear is 2, shown in Fig. 9, while in the case of road grade 10 deg the selected transmission gear is 1. Hence, at different selected gears,

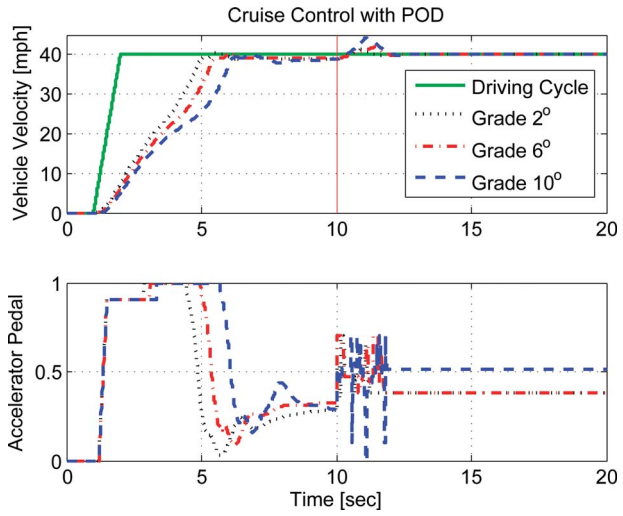

Fig. 8 Vehicle speed and accelerator pedal rate for different road grades by self-learning cruise control with POD

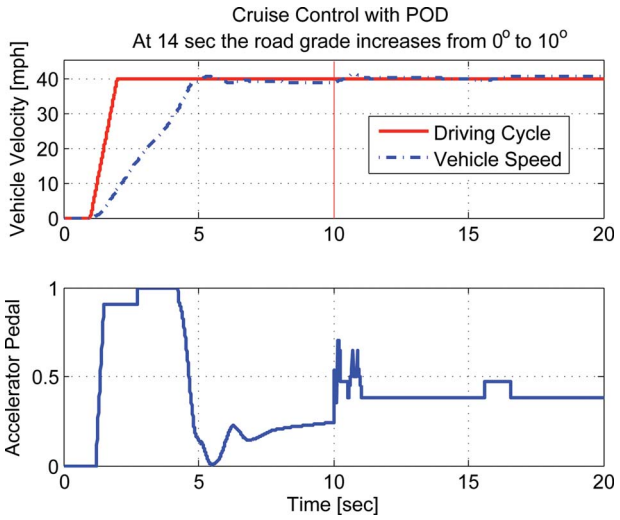

Fig. 10 Vehicle speed and accelerator pedal rate for a road grade increase from 0 deg to 10 deg

the accelerator pedal varies to maintain constant vehicle speed. In Fig. 10, the performance of the POD cruise controller is evaluated in a severe driving scenario where the road grade changes from 0 deg to 10 deg, while the POD cruise controller is active. In this scenario, the POD is activated again at 10 s when the road grade is 0 deg, and at 14 s the road grade becomes 10 deg. The engine speed and the selected transmission gear for this scenario are shown in Fig. 11. While the vehicle is cruising at constant speed and the road grade changes from 0 deg to 10 deg, the vehicle's speed starts decreasing after some time. Once this occurs, the self-learning cruise controller senses the discrepancy between the desired and actual vehicle speeds and commands the accelerator pedal so as to correct the error. Consequently, there is a small time delay in the acceleration pedal command, illustrated in Fig. 10, which depends on vehicle inertia.

6 Concluding Remarks

This paper presented a computational model consisting of a new state-space representation and a learning mechanism capable of solving sequential decision-making problems under uncertainty in real time. The major difference between the existing methods and the proposed learning model is that the latter utilizes an evaluation function, which considers the expected cost that can be achieved by state transitions forward in time. The model aims to build autonomous intelligent systems that can learn to improve their performance over time in stochastic environments. Such systems must be able to sense their environments, estimate the consequences of their actions, and learn in real time the control policy that optimizes a specified objective. The performance of the model in deriving a control policy was evaluated on two applications: (a) the cart-pole balancing problem and (b) a vehicle cruise-controller development. In the first problem, the pendulum was made an autonomous intelligent system capable of realizing the balancing control policy when it was released from any angle between 3 deg and −3 deg. In the second problem, an autonomous cruise controller was implemented that was able to learn to maintain the desired vehicle speed at any road grade between 0 deg and 10 deg.

Future research should investigate the potential of advancing the POD method to accommodate more than one decision maker in sequential decision-making problems under uncertainty, known as multi-agent systems [41]. These problems are found in systems in which many intelligent decision makers (agents) interact with each other. The agents are considered to be autonomous entities. Their interactions can be either cooperative or selfish, i.e., the

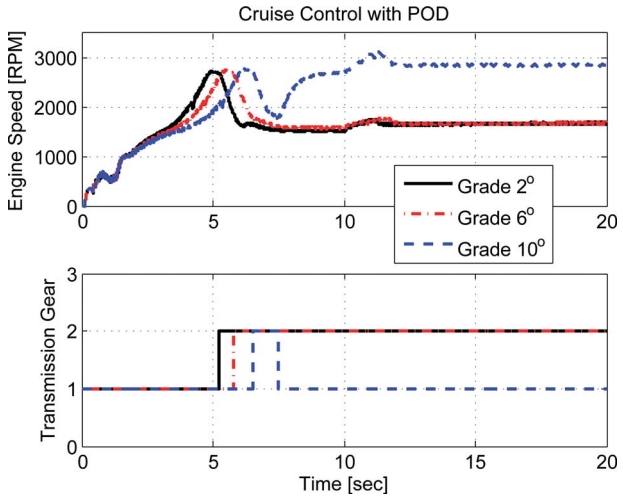

Fig. 9 Engine speed and transmission gear selection for different road grades by self-learning cruise control with POD

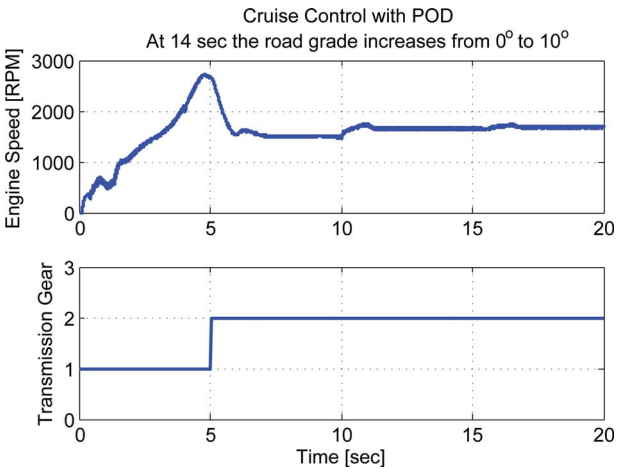

Fig. 11 Engine speed and transmission gear selection for a road grade increase from 0 deg to 10 deg

agents can share a common goal, e.g., control of vehicles operating in platoons to improve throughput on congested highways by allowing groups of vehicles to travel together in a tightly spaced platoon at high speeds. Alternatively, the agents can pursue their own interests.

Acknowledgment

This research was partially supported by the Automotive Research Center (ARC), a U.S. Army Center of Excellence in Modeling and Simulation of Ground Vehicles at the University of Michigan. The engine simulation package ENDYNA was provided by TESIS DYNAware GmbH. This support is gratefully acknowledged.

References

[1] Bertsekas, D. P. and Shreve, S. E., 2007, *Stochastic Optimal Control: The Discrete-Time Case*, 1st ed., Athena Scientific, Nashua, NH.
[2] Gosavi, A., 2004, "Reinforcement Learning for Long-Run Average Cost," Eur. J. Oper. Res., **155**, pp. 654–74.
[3] Bertsekas, D. P. and Tsitsiklis, J. N., 1996, "Neuro-Dynamic Programming" (Optimization and Neural Computation Series, 3), 1st ed., Athena Scientific, Nashua, NH.
[4] Sutton, R. S. and Barto, A. G., 1998, "Reinforcement Learning: An Introduction" (Adaptive Computation and Machine Learning), MIT Press, Cambridge, MA.
[5] Borkar, V. S., 2000, "A Learning Algorithm for Discrete-Time Stochastic Control," Probability in the Engineering and Informational Sciences, **14**, pp. 243–258.
[6] Samuel, A. L., 1959, "Some Studies in Machine Learning Using the Game of Checkers," IBM J. Res. Dev., **3**, pp. 210–229.
[7] Samuel, A. L., 1967, "Some Studies in Machine Learning Using the Game of Checkers. II: Recent Progress," IBM J. Res. Develop., **11**, pp. 601–617.
[8] Sutton, R. S., 1984, "Temporal Credit Assignment in Reinforcement Learning," Ph.D. thesis, University of Massachusetts, Amherst, MA.
[9] Sutton, R. S., 1988, "Learning to Predict by the Methods of Temporal Difference," Mach. Learn., **3**, pp. 9–44.
[10] Watkins, C. J., 1989, "Learning From Delayed Rewards," Ph.D. thesis, Kings College, Cambridge, England.
[11] Watkins, C. J. C. H., and Dayan, P., 1992, "Q-Learning," Mach. Learn., **8**, pp. 279–92.
[12] Kaelbling, L. P., Littman, M. L., and Moore, A. W., 1996, "Reinforcement Learning: A Survey," J. Artif. Intell. Res., **4**, pp. 237–285.
[13] Schwartz, A., 1993, "A Reinforcement Learning Method for Maximizing Undiscounted Rewards," Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, pp. 298–305.
[14] Mahadevan, S., 1996, "Average Reward Reinforcement Learning: Foundations, Algorithms, and Empirical Results," Mach. Learn., **22**, pp. 159–195.
[15] Sutton, R. S., 1990, "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming," Proceedings of the Seventh International Conference on Machine Learning, Austin, TX, pp. 216–224.
[16] Sutton, R. S., 1991, "Planning by Incremental Dynamic Programming," Proceedings of the Eighth International Workshop on Machine Learning (ML91), Evanston, IL, pp. 353–357.
[17] Moore, A. W., and Atkinson, C. G., 1993, "Prioritized Sweeping: Reinforcement Learning With Less Data and Less Time," Mach. Learn., **13**, pp. 103–30.
[18] Peng, J., and Williams, R. J., 1993, "Efficient Learning and Planning Within the Dyna Framework," Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, pp. 168–74.
[19] Barto, A. G., Bradtke, S. J., and Singh, S. P., 1995, "Learning to Act Using Real-Time Dynamic Programming," Artif. Intell., **72**, pp. 81–138.
[20] Malikopoulos, A. A., Papalambros, P. Y., and Assanis, D. N., 2007, "A State-Space Representation Model and Learning Algorithm for Real-Time Decision-Making Under Uncertainty," Proceedings of the 2007 ASME International Mechanical Engineering Congress and Exposition, Seattle, WA, Nov. 11–15.
[21] Puterman, M. L., 2005, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2nd rev. ed., Wiley-Interscience, New York.
[22] Malikopoulos, A. A., Papalambros, P. Y., and Assanis, D. N., 2007, "A Learning Algorithm for Optimal Internal Combustion Engine Calibration in Real Time," Proceedings of the ASME 2007 International Design Engineering Technical Conferences Computers and Information in Engineering Conference, Las Vegas, NV, Sept. 4–7.
[23] Malikopoulos, A. A., 2008, "Real-Time, Self-Learning Identification and Stochastic Optimal Control of Advanced Powertrain Systems," Ph.D. thesis, Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI.
[24] Malikopoulos, A. A., Assanis, D. N., and Papalambros, P. Y., 2007, "Real-Time, Self-Learning Optimization of Diesel Engine Calibration," Proceedings of the 2007 Fall Technical Conference of the ASME Internal Combustion Engine Division, Charleston, SC, Oct. 14–17.
[25] Malikopoulos, A. A., Assanis, D. N., and Papalambros, P. Y., 2008, "Optimal Engine Calibration for Individual Driving Styles," Proceedings of the SAE 2008 World Congress and Exhibition, Detroit, MI, Apr. 14–17, SAE Paper No. 2008-01-1367.
[26] Iwata, K., Ito, N., Yamauchi, K., and Ishii, N., 2000, "Combining Exploitation-Based and Exploration-Based Approach in Reinforcement Learning," Proceedings of the Intelligent Data Engineering and Automated–IDEAL 2000, Hong Kong, China, pp. 326–31.
[27] Ishii, S., Yoshida, W., and Yoshimoto, J., 2002, "Control of Exploitation–Exploration Meta-Parameter in Reinforcement Learning," Neural Networks, **15**, pp. 665–87.
[28] Chan-Geon, P., and Sung-Bong, Y., 2003, "Implementation of the Agent Using Universal On-Line Q-Learning by Balancing Exploration and Exploitation in Reinforcement Learning," Journal of KISS: Software and Applications, **30**, pp. 672–80.
[29] Miyazaki, K., and Yamamura, M., 1997, "Marco Polo: A Reinforcement Learning System Considering Tradeoff Exploitation and Exploration Under Markovian Environments," Journal of Japanese Society for Artificial Intelligence, **12**, pp. 78–89.
[30] Hernandez-Aguirre, A., Buckles, B. P., and Martinez-Alcantara, A., 2000, "The Probably Approximately Correct (PAC) Population Size of a Genetic Algorithm," 12th IEEE Internationals Conference on Tools With Artificial Intelligence, pp. 199–202.
[31] Malikopoulos, A. A., 2008, "Convergence Properties of a Computational Learning Model for Unknown Markov Chains," Proceedings of the 2008 ASME Dynamic Systems and Control Conference, Ann Arbor, MI, Oct. 20–22.
[32] Anderson, C. W., 1989, "Learning to Control an Inverted Pendulum Using Neural Networks," IEEE Control Syst. Mag., **9**, pp. 31–7.
[33] Williams, V., and Matsuoka, K., 1991, "Learning to Balance the Inverted Pendulum Using Neural Networks," Proceedings of the 1991 IEEE International Joint Conference on Neural Networks, Singapore, pp. 214–19, Cat. No.91CH3065-0.
[34] Zhidong, D., Zaixing, Z., and Peifa, J., 1995, "A Neural-Fuzzy BOXES Control System With Reinforcement Learning and its Applications to Inverted Pendulum," Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, Vancouver, BC, Canada, pp. 1250–4, Cat. No.95CH3576-7.
[35] Jeen-Shing, W., and McLaren, R., 1997, "A Modified Defuzzifier for Control of the Inverted Pendulum Using Learning," Proceedings of the 1997 Annual Meeting of the North American Fuzzy Information Processing Society–NAFIPS, Syracuse, NY, pp. 118–23, Cat. No. 97TH8297.
[36] Mustapha, S. M., and Lachiver, G., 2000, "A Modified Actor-Critic Reinforcement Learning Algorithm," Proceedings of the 2000 Canadian Conference on Electrical and Computer Engineering, Halifax, NS, Canada, pp. 605–9.
[37] Si, J., and Wang, Y. T., 2001, "On-line Learning Control by Association and Reinforcement," IEEE Trans. Neural Netw., **12**, pp. 264–276.
[38] Zhang, B. S., Leigh, I., and Leigh, J. R., 1995, "Learning Control Based on Pattern Recognition Applied to Vehicle Cruise Control Systems," Proceedings of the American Control Conference, Seattle, WA, pp. 3101–3105.
[39] Shahdi, S. A., and Shouraki, S. B., 2003, "Use of Active Learning Method to Develop an Intelligent Stop and Go Cruise Control," Proceedings of the IASTED International Conference on Intelligent Systems and Control, Salzburg, Austria, pp. 87–90.
[40] TESIS, http://www.tesis.de/en/.
[41] Panait, L., and Luke, S., 2005, "Cooperative Multi-Agent Learning: The State of the Art," Auton. Agents Multi-Agent Syst., **11**, pp. 387–434.