Annual Reviews in Control 33 (2009) 11–24

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Stochastic learning and optimization—A sensitivity-based approach[☆]

Xi-Ren Cao

Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China

ARTICLE INFO

Article history:
Received 1 December 2008
Accepted 11 March 2009
Available online 16 April 2009

Keywords:
Perturbation analysis
Markov decision processes
Reinforcement learning
Stochastic control
Performance potentials
Event-based optimization

ABSTRACT

We introduce a sensitivity-based view to the area of learning and optimization of stochastic dynamic systems. We show that this sensitivity-based view provides a unified framework for many different disciplines in this area, including perturbation analysis, Markov decision processes, reinforcement learning, identification and adaptive control, and singular stochastic control; and that this unified framework applies to both the discrete event dynamic systems and continuous-time continuous-state systems. Many results in these disciplines can be simply derived and intuitively explained by using two performance sensitivity formulas. In addition, we show that this sensitivity-based view leads to new results and opens up new directions for future research. For example, the n th bias optimality of Markov processes has been established and the event-based optimization may be developed; this approach has computational and other advantages over the state-based approaches.

1. Introduction

Performance optimization plays an important role in the design and operation of modern engineering and economic systems in many areas, including communications (Internet and wireless networks), manufacturing, logistics, robotics, bio-informatics, and finance. Most such systems are too complicated to be analyzed, or the parameters of the system models cannot be easily obtained. Therefore, learning techniques have to be applied.

Learning and optimization of stochastic systems is a multi-disciplinary area that has attracted wide attention from researchers in many disciplines, including control systems, operations research, computer science, and financial engineering. Many research areas share a common goal: to make the "best decision" to optimize a system's performance. These areas include perturbation analysis (PA) in discrete event dynamic systems (DEDSs) (Cao, 1994; Cassandras & Lafortune, 1999; Ho & Cao, 1991), Markov decision processes (MDPs) in operations research (Bertsekas, 1995, 2001, 2007; Puterman, 1994; Veinott, 1969), reinforcement learning (RL) in computer science (Sutton & Barto, 1998), neuro-dynamic programming (NDP) (Bertsekas & Tsitsiklis, 1996; Werbos, 1992), identification, and adaptive control (I&AC) in control systems (Åström & Wittenmark, 1989), and portfolio management in financial engineering (Oksendal & Sulem, 2007).

Different disciplines take different perspectives and have different formulations for the problems with the same goal. In this paper, we introduce a sensitivity point of view to the area of learning and optimization, which provides a unified framework for the different disciplines, including PA, MDPs, RL, and I&AC. This sensitivity-based view unifies the optimization theories of both the discrete event dynamic systems (DEDS) and continuous-time continuous-state (CTCS) systems. We show that many results in these disciplines in both DEDS and CTCS systems can be derived simply and explained clearly and intuitively from two performance sensitivity (difference and derivative) formulas. In addition, we show that with this sensitivity-based view, new results and approaches such as the n th bias optimality of Markov processes and the event-based optimization can be developed (Cao, 2007).

2. An overview of learning and optimization

The goal of learning and optimization is to make the "best" decisions to optimize, or to improve, the performance of a system based on the information obtained by observing and analyzing the system's behavior. A system's behavior is usually represented by a model, or by the sample paths (also called *trajectories*) of the system. A *sample path* is a record of the operation history of a system.

In this paper, we mainly discuss stochastic dynamic systems. A dynamic system evolves as time passes by. It is generally easier to explain the ideas with a discrete time and finite state model, which is assumed in most part of the paper, except for the part where we discuss CTCS systems. In addition to its dynamic nature, a stochastic system is always subject to random influences caused by noise or other uncertainties.

[☆] This paper was based on a plenary talk at the 17th World Congress of International Federation of Automatic Control, Seoul, Korea, July 2008, and was submitted to IFAC journal *Annual Review in Control*. This work is supported by a grant from Hong Kong UGC (No. 610806).

1367-5788/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.arcontrol.2009.03.003

2.1. States, actions, and observations

To study the system behavior, we need to describe precisely the system's status. A system's status at any time $l = 0, 1, \ldots$ can be represented by a quantity called the system's *state* at time l, denoted as X_l, $l = 0, 1, \ldots$. The *state space* (i.e., the set of all states) is denoted as $\mathcal{S}$, and for simplicity, we assume it to be finite and denote it as $\mathcal{S} = \{1, 2, \ldots, S\}$. A *sample path* of a system is a record of state history denoted as $\boldsymbol{X} = \{X_0, X_1, \ldots\}$. In stochastic dynamic systems, X_l, $l = 0, 1, \ldots$, are random variables (may be multi-dimensional random vectors). A system's dynamic behavior is then represented by its sample paths. We denote a "finite-length" sample path as $\boldsymbol{X}_l := \{X_0, X_1, \ldots, X_l\}$.

In optimization problems, at any time l, we can apply an *action*, denoted as $A_l \in \mathcal{A}$, $l = 0, 1, \ldots$, to the system, where $\mathcal{A}$ is an action space. In this paper, we assume that $\mathcal{A}$ contains a finite number of actions, but in general it may contain infinitely many actions, or may even be a continuous space. The actions $A_0, A_1, \ldots$ may affect the evolution of the system. Because the actions affect the system behavior, the operation history of a system should include the actions. Let $\boldsymbol{A}_{l-1} := \{A_0, A_1, \ldots, A_{l-1}\}$ denote an action history with a finite length and $\boldsymbol{A} := \{A_0, A_1, \ldots\}$ denote an infinitely long action history. Taking the actions into consideration, we denote a sample path as $\boldsymbol{H} := (\boldsymbol{X}, \boldsymbol{A})$, or $\boldsymbol{H}_l := (\boldsymbol{X}_l, \boldsymbol{A}_{l-1})$.

In many cases, the system's state cannot be exactly observed, and we can only observe a random variable Y_l at time l that is related to X_l, $l = 0, 1, \ldots$. The observation history is denoted as $\boldsymbol{Y} := \{Y_0, Y_1, \ldots\}$, or $\boldsymbol{Y}_l := \{Y_0, Y_1, \ldots, Y_l\}$. In such cases, we say that the system is partially observable. The information history up to time l is $\boldsymbol{H}_l := (\boldsymbol{Y}_l, \boldsymbol{A}_{l-1})$. When $Y_l = X_l$, for all $l = 0, 1, \ldots$, we say that the system is completely observable. In such cases, we have $\boldsymbol{H}_l = (\boldsymbol{X}_l, \boldsymbol{A}_{l-1})$. Note that even for partially observable systems, we reserve the word "sample path" for $\boldsymbol{H}_l = (\boldsymbol{X}_l, \boldsymbol{A}_{l-1})$, or $\boldsymbol{H} = (\boldsymbol{X}, \boldsymbol{A})$, and we call $\boldsymbol{H}_l := (\boldsymbol{Y}_l, \boldsymbol{A}_{l-1})$ and $\boldsymbol{H} = (\boldsymbol{Y}, \boldsymbol{A})$ information histories.

2.2. Rewards and performance measures

Associated with each sample path $\boldsymbol{H}_L = (\boldsymbol{X}_L, \boldsymbol{A}_{L-1})$, there is a *reward* denoted as $\eta_L(\boldsymbol{H}_L)$. Because the states $\boldsymbol{X}_L$ and the actions $\boldsymbol{A}_{L-1}$ are generally random, $\eta_L(\boldsymbol{H}_L)$ is usually a random variable. For finite-length problems, $\eta_L(\boldsymbol{H}_L)$ represents the reward (total, discounted, or average, etc.) received when the system is going through the sample path $\boldsymbol{H}_L$. The *performance measure* η (or simply called the *performance*) is defined as the mean of the sample-path-based rewards

$$\eta = E[\eta_L(\boldsymbol{H}_L)], \tag{1}$$

where "E" denotes the expectation. For sample paths with infinitely long lengths, the *performance measure* η is defined as the limit of the mean rewards

$$\eta = \lim_{L \to \infty} E[\eta_L(\boldsymbol{H}_L)], \tag{2}$$

in which we assume that both the expectation and the limit exist. In many cases, $\eta_L(\boldsymbol{H}_L)$ represents the average reward per step received by the system during the operation, and η is called the long-run average performance.

2.3. The learning and optimization problem

A general description of the learning and optimization problem is illustrated by Fig. 1. In the figure, the shaded area represents a stochastic dynamic system. The system is essentially a black box and it can only interact with the outside through its inputs and outputs. The inputs provide a vehicle to intervene or to control the operation of the system, and/or to affect the reward of the operation. The inputs are usually the actions taken that will affect the future evolution of the system. In some cases, an input may simply control the system operation modes, or tune the values of system parameters, etc. In this terminology, setting different values for system parameters is viewed as taking different actions. It is usually assumed that the available actions are known to us (e.g., we know that we can accept or reject a packet in a communication system, or we can tune the rate of a transmission line to θ megabit/second). The outputs provide a window for observing the system. That is, the outputs are the observations Y_l, $l = 0, 1, \ldots$. Associated with every system, there is a performance measure η, which depends on the inputs.

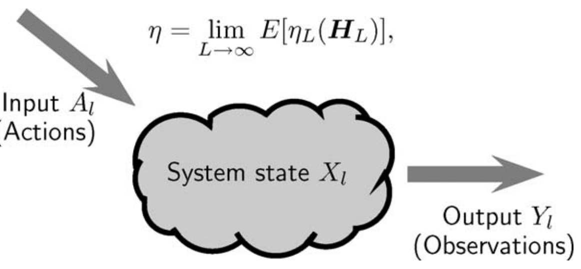
$\eta = \lim_{L \to \infty} E[\eta_L(\boldsymbol{H}_L)]$,
Input A_l (Actions)
System state X_l
Output Y_l (Observations)

Fig. 1. A model of learning and optimization.

The goal of an optimization problem is to answer the following question: *Based on the information we know about the system at any particular time, i.e., the output history learned from observation and the input (action) history, what action should we take at that time so that we can obtain the best possible system performance?*

2.4. Policies

The information history $\boldsymbol{H}_l = \{\boldsymbol{Y}_l, \boldsymbol{A}_{l-1}\}$, with $\boldsymbol{Y}_l = \{Y_0, Y_1, \ldots, Y_l\}$ being the observation history and $\boldsymbol{A}_{l-1} = \{A_0, A_1, \ldots, A_{l-1}\}$ being the action history (with $\boldsymbol{A}_{-1} := \emptyset$), represents all the information available at time l before an action is taken at l, $l = 0, 1, \ldots$. Based on this information, an action can be chosen by following some rules, called a *policy*, denoted as $d_l : A_l = d_l(\boldsymbol{H}_l), A_l \in \mathcal{A}$. (This is called a *deterministic policy*.) The set of all policies is called a *policy space* and is denoted as $\mathcal{D}$.

The optimization problem now becomes to find a policy that achieves the maximum of the system performance. Such a policy is called an *optimal policy*. When the number of policies is finite, such optimal policies always exist and may not be unique.

If we have a mathematical model for the system in Fig. 1, the optimal policies might be found analytically; in many cases, however, a mathematical model may not exist, and we need to observe and analyze the sample paths of the system to determine the performance and/or to make improvement decisions. This is called "*learning*". In engineering applications, at the design stage, sample paths can only be obtained by simulation following a system model; and while a system is operating, its sample paths can also be obtained by direct observation. If learning and optimization is implemented by simulation, the approach is called a *simulation-based* approach. With simulation, we may even let the system operate under policies that are not feasible in a real system. For real systems, performance optimization (or improvement) decisions can be made through learning the system behavior by observing and analyzing its sample paths while the system is operating without interruption; we call such an approach an *on-line* approach.

2.5. The Markov model

The optimization problem formulated with the black-box system (its structure is completely unknown) shown in Fig. 1 is too difficult to solve (see discussions in the next section). To develop specific optimization approaches, we need to introduce some structures into the system model. Perhaps the most widely used model for stochastic dynamic systems is the *Markov model*.

The word "*state*" is used in a strict sense in the Markov model (úinlar, 1975). This means that *given the current state* X_l, *the system's future behavior* $\{X_{l+1}, X_{l+2}, \dots\}$ *is independent of its past history* $\{X_0, X_1, \dots, X_{l-1}\}$, *for all* $l = 1, 2, \dots$. This is called the *Markov property*, and a stochastic process $\boldsymbol{X} = \{X_0, X_1, \dots\}$ satisfying Markov property is called a *Markov chain*. Intuitively, a state in a Markov chain completely captures the system's current status in regard to its future evolution.

The evolution of a Markov chain is determined by its *transition probability matrix* $P = [p(j|i)]_{i,j=1}^{S}$, where $p(j|i) = \mathcal{P}(X_{l+1} = j | X_l = i)$ is the transition probability that the system moves to state j at time $l+1$ when it is in state i at time l. We assume that the system is *homogenous* so $p(j|i)$, $i, j \in \mathcal{S}$, do not depend on l. A (homogenous) Markov chain is called *irreducible*, if starting from any state i, the system can reach any other state $j \in \mathcal{S}$ in a finite number of steps. A Markov chain with a finite number of states is called *ergodic*, if it is irreducible and aperiodic (úinlar, 1975). For an ergodic Markov chain, the *steady-state probabilities*

$$\pi(i) = \lim_{l \to \infty} \mathcal{P}(X_l = i | X_0 = j), \quad i, j \in \mathcal{S},$$

exist, which do not depend on the initial state $j \in \mathcal{S}$. Let $\pi := (\pi(1), \dots, \pi(S))$ denote the (row) vector of the steady-state probabilities. Then we have

$$\pi = \pi P, \quad \pi e = 1, \tag{3}$$

where $e = (1, 1, \dots, 1)^T$ is a vector with all components being 1, and the superscript "T" denotes the transpose.

With the Markov model, the actions control the transition probabilities of the state process. If action $\alpha \in \mathcal{A}$ is taken at time l (i.e., $A_l = \alpha$), then the transition probabilities at time l are denoted as $p^{\alpha}(X_{l+1}|X_l)$, $X_l, X_{l+1} \in \mathcal{S}$, $l = 0, 1, \dots$. With the Markov model, we further assume that there is a reward function denoted as $f(i, \alpha)$, $i \in \mathcal{S}, \alpha \in \mathcal{A}$. At time l, if the system is in state i, i.e., $X_l = i$, and action $A_l = \alpha \in \mathcal{A}$ is taken, then the system receives a reward of $f(i, \alpha)$. With the reward function, many performance measure can be defined, including the discounted reward, total reward, etc. We mainly discuss the *long-run average reward* defined as

$$\eta = \lim_{L \to \infty} \frac{1}{L} E\left\{ \sum_{l=0}^{L-1} f(X_l, A_l) \middle| X_0 = i \right\}. \tag{4}$$

If the state process is an ergodic Markov chain (depending on A_l and $p^{\alpha}(X_{l+1}|X_l)$), the long-run average reward does not depend on the initial state and we have

$$\eta := \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} f(X_l, A_l), \qquad w.p.1.$$

In this paper, for simplicity, if not mentioned otherwise, we assume that the state X_l, $l = 0, 1, \dots$, can be observed exactly. The information history becomes $\boldsymbol{H}_l = (\boldsymbol{X}_l, \boldsymbol{A}_{l-1})$, and a policy becomes $A_l = d_l(\boldsymbol{X}_l, \boldsymbol{A}_{l-1})$, $A_l \in \mathcal{A}$. Because of the Markov property, if a state process is Markov, the current state X_l contains all the information in the system's history in regard to its future behavior. We may expect that in many cases a policy depending on only X_l may do as well as a policy depending on the entire history $\boldsymbol{H}_l = (\boldsymbol{X}_l, \boldsymbol{A}_{l-1})$ for controlling the system's future behavior. Therefore, we may only consider the policies $A_l = d_l(X_l)$, $l = 0, 1, \dots$.

2.6. Stationary and randomized policies

A policy $A_l = d_l(X_l)$, $X_l \in \mathcal{S}$, $A_l \in \mathcal{A}$, $l = 0, 1, \dots$, is called a *stationary policy* if it does not depend on time l; such a policy is denoted as $A = d(X), X \in \mathcal{S}$, which is a mapping from the state space $\mathcal{S}$ to the action space $\mathcal{A}$. The action $d(i), i \in \mathcal{S}$, controls the transition probabilities of state i. With a stationary policy d, the transition probabilities when the state is $i \in \mathcal{S}$ are denoted as $p^{d(i)}(j|i)$, $j \in \mathcal{S}$. The system under policy $d(X)$ is Markov, and the corresponding transition matrix is denoted as $P^d := [p^{d(i)}(j|i)]$. The reward function can be expressed as a column vector $f^d = (f(1, d(1)), \dots, f(S, d(S)))^T$. The effect of a policy d to the system can be completely described by (P^d, f^d); therefore, we may refer to a policy as $d := (P^d, f^d)$. In addition, we will simply use $d = (P, f)$ as a generic notation for a policy. It is known (Puterman, 1994) that there exists a stationary policy that is optimal.

A (stationary) *randomized policy* $\nu = d(X)$ assigns a distribution ν over the action space $\mathcal{A}$ for every state $X = i \in \mathcal{S}$; it is a mapping from the state space $\mathcal{S}$ to the space of the distribution functions over the action space. For example, suppose that $\mathcal{A} = \{\alpha_1, \alpha_2, \dots, \alpha_M\}$. For any state $i \in \mathcal{S}$, a randomized policy assigns a distribution $\nu = (p_1(i), p_2(i), \dots, p_M(i))$ on $\mathcal{A}$, with $\sum_{k=1}^{M} p_k(i) = 1$. When the system state is i, we take action α_k with probability $p_k(i)$, $k = 1, 2, \dots, M$, $i \in \mathcal{S}$. A deterministic policy is a special case of a randomized policy ν where $p_k(i) = 1$ for some $k \in \{1, 2, \dots, M\}$, with k depending on i, $i \in \mathcal{S}$.

3. Fundamental limitations and search methods

An optimization problem is to find an optimal policy in a given policy space $\mathcal{D}$. First, we observe that even for a small problem, the policy space is too large for us to handle. For example, for a (small) system with $S = 100$ states and $M = 2$ actions available in each state, the number of stationary policies is $M^S = 2^{100} \approx 10^{30}$! With the fastest PC (10 GHz) currently available to count the policies at a speed of 1 policy/Hz (i.e., 10G policies/second), it requires 3×10^{12} years to finish the counting!

3.1. Learning

To develop efficient algorithms for performance optimization, we need to explore the special features of a system. This process is called *learning*. For dynamic systems, learning may involve observing and analyzing a sample path of a system to obtain necessary information; this is in the normal sense of the word "learning", as it is used in research areas such as *reinforcement learning*. Simulation-based and on-line optimization approaches are based on learning from sample paths. On the other hand, we may also analytically study the behavior of a system under a policy to learn how to improve the system performance. This is done with a mathematical model of the system. In a wide sense, we shall also call this analytical process "learning" (to learn something about the system performance under other policies from its behavior under the current policy).

3.2. Fundamental limitations

Obviously, the task of learning and optimization is complicated and we are facing a vast forest and wish to find a path in it to reach our destination at the top of a peak. It is wise to pause for a short while and take an overview of the forest from the outside to see which directions may possibly lead us to our goal quickly. Indeed, we are constrained by some philosophical and logical facts that significantly limit what we can do. These facts are simple and intuitively obvious, yet they provide general principles that chart the paths in our journey of developing learning and optimization theories and methodologies. Because of the importance as well as the simplicity of these facts, we state them as the "fundamental limitations":

The Fundamental Limitations of Learning and Optimization

A. A system can be run and/or studied under only one policy at a time.
B. If no structural information of the system is available, by learning from the behavior of a system under one policy, we cannot obtain any information about the performance of other policies.
C. We can only compare two policies at a time.

These simple rules describe the boundaries in developing learning and optimization approaches. First of all, if there is no structural information about the system, from the fundamental limitations A and B, we need to observe/analyze the system under each policy, with a model or by simulating or operating the system under the policy, to analytically compute or to estimate its performance. In such cases, the search methods are the only approaches for optimization. On the other hand, if we have some knowledge about the system structure, we may infer some information about the performance of the system under other polices while analyzing its behavior under one policy. More efficient approaches may be developed to identify optimal policies.

3.3. The search methods

The only search method that guarantees to give an optimal policy under any circumstance is the exhaustive search, in which the performance of every individual policy is estimated or computed. From the fundamental limitation C, for M policies we need to make $M-1$ pairwise comparisons. This exhaustive search method is stated as follows:

Exhaustive Search Given M policies d_i, $i=1,2,\ldots,M$. Let η^{d_i} be the performance of policy d_i, $i=1,2,\ldots,M$.

```
i. Set $\tilde{d} := d_1$, and $\tilde{\eta} := \eta^{d_1}$;
ii. For $i := 2$ to $M$, do: if $\eta^{d_i} > \tilde{\eta}$ then set $\tilde{d} := d_i$ and $\tilde{\eta} := \eta^{d_i}$.
```

The algorithm outputs an optimal policy and the optimal performance. In the algorithm, we may randomly order the policies. However, in many problems the number of policies increases exponentially with respect to the number of states. Therefore, exhaustive search, which requires computing and comparing the performance of every policy, is not computationally feasible for most practical problems. A variation is the blind random search, in which we randomly choose a limited number of policies to be evaluated and compared.

Moreover, if there is no additional information about the mapping $\eta^d : \mathcal{D} \to \mathcal{R}$ (such as its shape, or the continuity if the policy space $\mathcal{D}$ is continuous, or other similar properties regarding how the performance η^d, $d \in \mathcal{D}$, distribute over $\mathcal{D}$, etc.), any optimization scheme is no better than blind (random) search. This result is formulated as the "No Free Lunch Theorem", see (Ho, Zhao, & Pepyne, 2003).

Various search methods have been developed. These methods may work better than the blind random search when policy space $\mathcal{D}$ and/or the mapping $\eta^d : \mathcal{D} \to \mathcal{R}$ have some special features (e.g., policies near a good policy are also good). Among them are simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983), genetic algorithms (Srinivas & Patnaik, 1994), and the recently proposed cross-entropy method (Rubinstein & Kroese, 2004), model reference adaptive search (Hu, Fu, & Marcus, 2007), and nested partition method (Shi & Olafsson, 2000).

The recently developed *"Ordinal Optimization"* approach deals with the trade-off between accuracy and efficiency of random search. It proposes an interesting idea of a "soft goal" and opens up a new perspective for optimization. The main ideas are two folds: First, the search algorithm depends on the comparison of the performance of two policies, $\eta^{d_i} > \tilde{\eta}$. It is important to note that to verify this relationship we may not need to obtain the exact values of the performance of these two policies. For example, if the performance of two policies is quite different, then we may need only run a short simulation for each policy to verify this relationship with a high accuracy. Second, we may not need to sample and compare all the policies (impossible); it can be shown that even we sample only a small set of policies, we are able to get a "good enough" policy with a reasonably large probability. See (Ho et al., 2003; Ho, Zhao, & Jia, 2007) for details.

In summary, if we have no information about the system dynamics or structure, by observing/analyzing the system under one policy, we cannot know any information about the system's performance under other policies. Search methods, which requires us to know only the performance of the system under each policy, are the only approaches for optimization. Exhaustive search is not computationally feasible for most practical problems. Ordinary optimization searches for a good enough policy with significantly reduced computation. Other search methods may work better than random search if the performance does distribute "nicely" over the policy space.

If we know something about the structure/dynamics of the system, we may obtain some information about the performance of the system under other polices while analyzing its behavior under one policy. These additional information may lead to optimization approaches that are more efficient than the search methods. This is the focus of the rest of the paper.

4. A sensitivity-based view of learning and optimization

To develop more efficient approaches than the search methods, we need to explore the special feature of a system. Naturally, we wish to develop approaches that require as little structural information and can be applied to as many systems as possible. The question is *"HOW"*. The fundamental limitations also provide us with some hints.

4.1. Performance gradient

As indicated by the fundamental limitations A and B, if we analyze a system's behavior under one policy, we can hardly know its behavior under other policies. It is natural to believe that if two policies are "close" to each other, then the system under these two policies may behave similarly. If this is the case, when we are analyzing a system under a policy, it might be easier to "predict" the system behavior under a "close" policy and to calculate its performance than to do the same for a policy that is "far away". In other words, to predict the performance for a "close" policy may require as little knowledge about the system structure as possible.

If a policy space can be characterized by a continuous parameter θ, then two policies are "close" if their corresponding values for θ are close. Such a policy space is called a *continuous policy space*. For example, for Markov systems, policies correspond to transition probability matrices. Therefore, two policies can be viewed as "close" if their transition probability matrices are close (entry-by-entry). In modeling manufacturing or communication networks, policies may be characterized by production rates or transmission rates. Two policies are close if their corresponding rates are close. In randomized policies, the distributions $(p_1, p_2, \ldots, p_M)$ over the action space $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_M\}$ are continuous variables. Two randomized policies are close if their corresponding distribution functions are close.

Therefore, a reasonable step towards developing efficient and generally applicable approaches is to look at a "neighborhood" of a policy. The neighborhood must be small enough, so that the behavior of the system under the policies in this neighborhood of the policy can be predicted with as little knowledge about the system structure as possible. In mathematical terms, "small enough" is precisely described by the word "infinitesimal". When the performance of the policies in an infinitesimal neighborhood of a policy is known, we can further get the gradient of the performance in the policy space at this policy.

We may summarize the above discussion by the following statement:

Statement A:

With some knowledge about the system structure under different policies, by studying the behavior of a system under one policy, we can determine the performance of the system under the policies in a small neighborhood of this policy; i.e., determine the performance gradient.

The prediction of the performance for other (neighboring) policies while analyzing the system under one policy can be done analytically, if we can describe the structure mathematically (usually based on a model) and know the values of its parameters. However, in many cases, we always start by analyzing a sample path of the system. This is because

1. A sample path clearly illustrates the system dynamics, and sample-path-based analysis stimulates intuitive thinking.
2. In many practical problems, the size of the problem is too large for any analytical solution, or we may have only partial information about the system; for example, in some cases, we may only know the structure of the system but do not know the values of its parameters, and in some other cases, we know the values of the parameters, but the system structure is too complicated to model. Sample-path-based algorithms may be implemented easily even with these constraints, as long as there is a sample path available.

Of course, the results obtained by the sample-path-based approach can also be expressed in an analytical form.

4.2. Perturbation analysis (PA)

PA estimates the performance derivatives with respect to system parameters by analyzing a single sample path of a stochastic dynamic system.

Because PA emphasizes the dynamic nature of a stochastic system with discrete states, such a system is also called a *discrete event dynamic system* (DEDS) (Cassandras & Lafortune, 1999; Ho & Cao, 1991). PA was proposed in the late 1970s and early 1980s (Cao, 1994; Cassandras & Lafortune, 1999; Ho & Cao, 1991). The early work on PA focused on queueing systems. Later, the basic principles of PA were extended to Markov systems with both discrete- and continuous-time models (Cao, 2007).

The basic principles of PA are: We start with a given sample path of a system with parameter θ. Suppose that θ changes to $\theta + \Delta\theta$, $\Delta\theta << 1$. This small change in the parameter θ induces a series of perturbations on the sample path. The average effect of each single perturbation on the system performance can be measured by a fundamental quantity called a *perturbation*

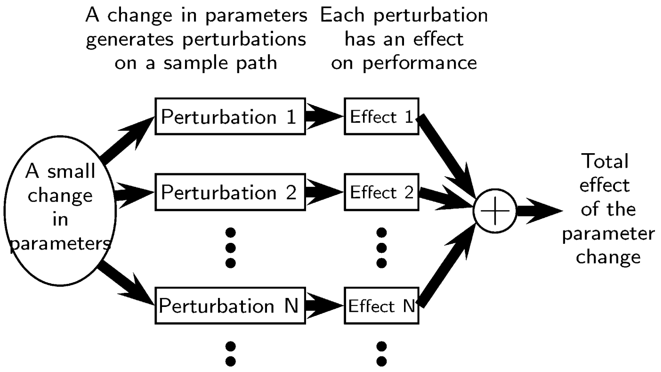
A change in parameters generates perturbations on a sample path
Each perturbation has an effect on performance
A small change in parameters
Perturbation 1
Perturbation 2
Perturbation N
Effect 1
Effect 2
Effect N
Total effect of the parameter change

Fig. 2. The basic principles of perturbation analysis.

realization factor, which can be estimated by observing and analyzing the given sample path of the system with parameter θ. Finally, the effect of the small change $\Delta\theta$ in the system parameter θ on the system performance equals the sum of the effects of all the perturbations (i.e., all the perturbation realization factors) induced by the parameter change on a sample path. It is important to note that the realization factor can be estimated on a sample path with parameter θ, and no information about the system with $\theta + \Delta\theta$ is needed. This means that the effect of a small change $\Delta\theta$, and hence the performance derivative with respect to θ, can be obtained on a sample path with parameter θ.

These basic principles are illustrated in Fig. 2. The important step in this approach is to determine the average effect of a single perturbation, i.e., the realization factor.

The PA principles illustrated in Fig. 2 can be applied to estimate the performance derivatives with respect to the transition probabilities of a Markov system. In this approach, the behavior of the black box in Fig. 1 is described by a Markov model with transition probability matrix P and the performance measure η is defined in (4). We assume that the states X_l are observable, i.e.; $Y_l = X_l$, $l = 0, 1, \ldots$. In this model, a policy d corresponds to a transition probability matrix denoted as P^d. We wish to get the performance gradients around a policy P^d in the policy space by analyzing the system's behavior under this policy P^d. For simplicity, we assume that the reward function is the same for all policies.

Let P^h be another policy, and let $\Delta P = P^h - P^d$. Define $P_\delta = P^d + \delta(\Delta P) = (1-\delta)P^d + \delta P^h$, $0 \le \delta \le 1$. P_δ is a randomized policy. With policy P_δ, in any state $k \in \mathcal{S}$ the system moves according to $p^{h(k)}(j|k)$, $j \in \mathcal{S}$, with probability δ, and moves according to $p^{d(k)}(j|k)$, $j \in \mathcal{S}$, with probability $1-\delta$. Let π_δ and η_δ be the steady-state probability and the performance measure associated with P_δ. We have $P_0 = P^d$, $P_1 = P^h$ and $\eta_0 = \eta^d$. The performance derivative at policy P^d along the direction ΔP (from P^d to P^h) is

$$\frac{\mathrm{d}\eta_\delta}{\mathrm{d}\delta}\Big|_{\delta=0} = \lim_{\delta\to 0}\frac{\eta_\delta - \eta}{\delta}.$$

Different P^h s correspond to different directional derivatives in the policy space.

The performance derivatives are obtained by predicting how the system would behave if we slightly perturb the transition probability matrix from P^d to P_δ, $\delta << 1$. Small changes in P^d induce a series of perturbations on a sample path of P^d. A perturbation on a sample path is a "jump" from one state i to another state j, $i, j \in \mathcal{S}$ (i.e., at some time l, the Markov chain with P^d was in state i, $X_l = i$, however, because of the slight change in the transition probabilities, the Markov chain with P_δ is in state $X_l = j$).

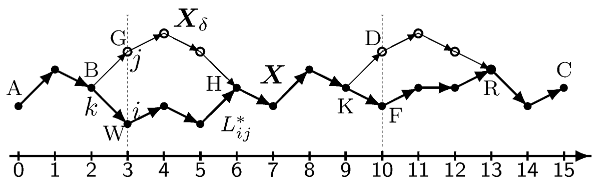

Fig. 3. The original and perturbed sample paths.

In Fig. 3, the path with thick arrows denotes an "original" sample path $\boldsymbol{X}$ with the transition probability matrix P^d. Suppose that P^d changes to P_δ, with $\delta << 1$, the sample path will change to another one denoted as $\boldsymbol{X}_\delta$. Because $\delta << 1$, we can expect that the changes are very few. That is, most part of $\boldsymbol{X}$ and $\boldsymbol{X}_\delta$ are the same, which are shown as segments $A-B$, $H-K$, and $R-C$. (Because of the limit in space, Fig. 3 is not drawn proportionally; in fact these segments should be very long, much longer than shown in the figure.) Occasionally, these two paths $\boldsymbol{X}$ and $\boldsymbol{X}_\delta$ are different; as shown in the figure, at $l=3$ there is a jump from state i to state j. The figure illustrates another jump at $l=10$. Starting from point G, $\boldsymbol{X}_\delta$ are different from $\boldsymbol{X}$. Because of the ergodicity, both paths $\boldsymbol{X}$ and $\boldsymbol{X}_\delta$ merge together at some point denoted as $l=L^*_{ij}=6$. Because $\delta << 1$, $\boldsymbol{X}_\delta$ in $G-H$, which follows P_δ, behaves the same as if following P^d. (We ignore the possibility that in $G-H$, there is another jump, since such a probability is of $o(\delta)$.)

Next, from the figure, it is clear that the average effect of the jump at $l=3$ on the infinite sum in the system performance η^d in (4) can be measured by the *perturbation realization factor*, denoted as (use X'_l for the state of $\boldsymbol{X}_\delta$ at time l)

$$\gamma^d(i,j) := E\left\{\sum_{l=0}^{L^*_{ij}}[f(X'_l) - f(X_l)]|X'_0 = j, X_0 = i\right\}$$

It can be shown that

$$\gamma^d(i,j) = g^d(j) - g^d(i), \quad \forall i,j \in \mathcal{S},$$

where $g^d(i)$ is called the *performance potential* (or simply the *potential*) of state i (Cao, 2007).

The performance potential is the main concept of performance optimization of Markov systems. Intuitively, the performance potential of state i, $g(i)$, of a policy P measures the "potential" contribution of state i to the long-run average reward η in (4). It is defined on a sample path of P as

$$g(i) := E\left\{\sum_{l=0}^{\infty}[f(X_l) - \eta]|X_0 = i\right\}. \tag{5}$$

(We have omitted the A_l that appeared in (4) since we assume that f does not depend on the actions; in addition, we subtract a constant η in each term to make the sum finite.) From (5), we can easily derive

$$\begin{aligned} g(i) &= \text{contribution of the current state } i \\ &\quad + \text{expected long-run `` potential'' contribution of the next state} \\ &= (f(i) - \eta) + \sum_{j\in\mathcal{S}} p(j|i)g(j). \end{aligned}$$

This can be written in a matrix form called the *Poisson equation*:

$$(I-P)g + \eta e = f, \tag{6}$$

where $g = (g(1), \ldots, g(S))^T$ is the potential vector.

From the definition of $g(i)$ in (5), we can see that the effect of a jump from state i to j on the long-run average reward (4) can be measured by $\gamma(i,j) = g(j) - g(i)$. Finally, the effect of a small (infinitesimal) change in a Markov chain's transition probability matrix (from P^d to P_δ) on the long-run average reward (4) can be decomposed into the sum of the effects of all the single perturbations (jumps on a sample path) induced by the change on a sample path. (Fig. 3 illustrates two such perturbations.) With these principles, we can intuitively derive the formulas for the performance derivative along any direction ΔP (from P^d to any P^h) in the policy space:

$$\frac{d\eta_\delta}{d\delta}|_{\delta=0} = \pi^d(\Delta P)g^d = \pi^d(P^h - P^d)g^d. \tag{7}$$

For a more rigorous analysis, see Cao (2007). (This formula can be also easily derived from the Poisson equation; however, the PA principles provide a clear and intuitive explanation for potentials and the derivative formula, and it can be easily extended to other non-standard problems for which the Poisson equation may not exist.)

From (7), knowing the steady-state probability π^d and the potential g^d of policy P^d, we can obtain the directional derivative $\frac{d\eta_\delta}{d\delta}|_{\delta=0}$ along any direction (ΔP) pointing to any given policy P^h from P^d. This is illustrated in Fig. 4. The potentials in (7) can be estimated (or "learned") from a sample path of the Markov chain under policy P^d. Optimization can be carried out using the performance derivatives together with stochastic approximation (Marbach & Tsitsiklis, 2001), or other gradient-based methods (Cao, 2007; Ho & Cao, 1991). It is explained in the next subsection that the potentials also play a key role in policy iteration in MDPs.

The extension of (7) to the case where the transition matrix depends arbitrarily on any parameter θ (denoted as P_θ with $P_0 = P$) is straightforward. Replacing ΔP in (7) with $(dP_\theta/d\theta)|_{\theta=0}$, we have

$$\frac{d\eta_\theta}{d\theta}|_{\theta=0} = \pi\frac{dP_\theta}{d\theta}|_{\theta=0}g,$$

where π and g are associated with $P = P_0$ (i.e., $\theta = 0$). Therefore, without loss of generality, we need only to discuss the linear case (7).

Now, suppose that policy P^h has a different reward function f^h. Let $\Delta P = P^h - P^d$ and $\Delta f = f^h - f^d$. Define $P_\delta = P^d + \delta(\Delta P)$ and $f_\delta = f^d + \delta(\Delta f)$, $0 \le \delta \le 1$. Then the directional derivative from (P, f) to (P_δ, f_δ) is

$$\frac{d\eta_\delta}{d\delta}|_{\delta=0} = \pi^d[(\Delta P)g^d + \Delta f]. \tag{8}$$

The potentials $g(i)$, $i = 1, 2, \ldots, S$, can be obtained by solving the Poisson equation (6), analytically or numerically, or can be estimated based on a sample path by using (5). The expectation can be approximated by average or by applying stochastic approximation on a sample path (Bertsekas, 1995, 2001, 2007; Bertsekas & Tsitsiklis, 1996; Cao, 2007). The performance derivatives can then be calculated by (8).

Furthermore, the derivatives (i.e., the term $\pi^d\Delta Pg^d$ in (7)) can be estimated as a whole without estimating each potential for

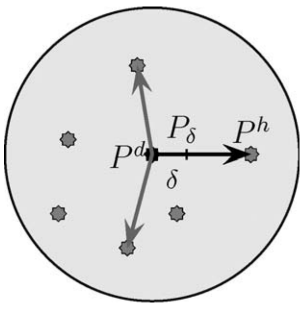

Fig. 4. The directional derivatives along any direction.

every state (cf. we may estimate $\eta^d = \pi^d f$ as a whole without estimating each component $\pi^d(i)$ for every $i \in \mathcal{S}$). This can be implemented on line without disturbing the operation of a system; efficient algorithms have been developed (Baxter & Bartlett, 2001; Baxter, Bartlett, & Weaver, 2001; Cao, 2005, 2007; Cao & Wan, 1998).

In summary, there are a number of advantages of PA: It can estimate performance derivatives along all directions based on a single sample path of a Markov chain. It can estimate derivatives along any direction on line as a whole, and the "curse of dimensionality" issue does not appear; furthermore, the approach applies to any policy space or subspace with constraints. However, PA-based approaches may reach a local optimal point.

In addition to PA of Markov systems, efficient algorithms were developed by PA principles for queueing systems (Cao, 1994; Ho & Cao, 1991); these algorithms utilize the special "coupling" feature among servers to determine the effect of a single perturbation. Recently, fluid model of queueing systems was introduced into PA, which provides good approximations (Cassandras, Sun, Panayiotou, & Wardi, 2003).

4.3. Performance differences

The gradient method does not apply to discrete policy spaces. For discrete policy spaces, we need to compare the performance of different policies that may not be close to each other.

The fundamental limitation C implies more than it seems on the surface. It says that all we can do in terms of optimization is based on a simple comparison of two policies. In other words, if we cannot compare two policies, then we have no way to do optimization. Furthermore, in many cases, we may even emphasize that the performance difference formula contains almost all the information about what we can do in performance optimization. This simple philosophical point guides the direction of our research in optimization: *We should always start with developing a formula for the difference of the performance measure of any two policies and then to investigate what we can learn from this performance difference formula.* In many cases, it is not difficult to derive such a difference formula for a particular problem, yet the insights provided and the results thus obtained can be remarkable.

How much we can get from the performance difference formula depends on the system structure. So far, the best result is that with some assumptions such as the independent-action assumption in Markov decision processes, by analyzing the system's behavior under one policy, we can find another policy that performs better, if such a policy exists (see the discussion below).

We may summarize the above discussion by the following statement:

Statement B:

With some assumptions on the system structure, by studying the behavior of a system under one policy, we can find a policy that performs better, if such a policy exists.

4.4. Markov decision processes

It has been shown (Cao, 2007) that the MDP theory (see, e.g., (Bertsekas, 1995, 2001, 2007; Puterman, 1994; Veinott, 1969)) can be developed based on the performance difference formula. MDPs use the Markov model and policies defined in Section 2; in addition, it assumes that the action at different states $d(i), i \in \mathcal{S}$, can be chosen independently (the *independent-action assumption*). For any policy d, we have transition probability matrix P^d, reward vector f^d, steady-state probability π^d, and performance η^d. The goal of MDPs is to find a policy $\widehat{d}$ such that its performance is the best among all policies.

Policy iteration is one of the main solutions to MDPs. Its basic principle is the same as Statement B: *With the independent-action assumption, by analyzing the behavior of the system under one policy, we can always find another policy under which the system performs better, if such a policy exists.* This can be shown by following the performance difference formula.

Consider two policies (P^d, f^d) and (P^h, f^h) with steady-state probabilities π^d, π^h and performance η^d and η^h, respectively. Let g^d be the potential of policy (P^d, f^d). Left-multiplying both side of the Poisson equation (6)$(I - P^d)g^d + \eta^d e = f^d$ with π^h, we obtain the performance difference formula

$$\eta^h - \eta^d = \pi^h[(\Delta P)g^d + \Delta f], \tag{9}$$

where $\Delta P = P^h - P^d$ and $\Delta f = f^h - f^d$. This equation can be also derived with a sample-path-based argument intuitively by first principles (Cao, 2007). The sample-path-based argument provides a clear intuition that can be extended to problems where the Poisson equation does not exist.

In (8), both π^d and g^d can be obtained from analyzing the system with policy (P^d, f^d). Thus, from (8), we can obtain the directional derivatives at P^d along any given direction $\Delta P = P^h - P^d$ without analyzing the system under (P^h, f^h). However, to obtain the performance difference with (9), we need to solve for both π^h and g^d. This is the same as a comparison in exhaustive search because we need to analyze both systems to compare the performance of the two systems. Thus, (9) saves nothing if we wish to get the exact value of the difference $\eta^d - \eta^h$.

Fortunately, all is not lost. The particular factorized form of (9) can be utilized. In fact, the updating procedure in policy iteration is based on (9) and the following simple fact: $\pi^h > 0$ (i.e., $\pi^h(i) > 0$ for all $i \in \mathcal{S}$) for any ergodic P^h. Thus, for any given P^d, if we can find a P^h such that $(\Delta P)g^d + \Delta f = (P^h g^d + f^h) - (P^d g^d + f^d) \geq 0$ with at least one positive component, then $\eta^h > \eta^d$. In particular, there is no need to solve for π^h in the procedure. Conventionally, in state i we choose the action that maximizes the ith component of $P^h g^d + f^h$ as $h(i)$; i.e., we choose $h(i)$, $i \in \mathcal{S}$, such that

$$\begin{aligned}&\sum_{j=1}^{S} p^{h(i)}(j|i)g^d(j) + f(i, h(i))\\ &\quad = \max\left\{\sum_{j=1}^{S} p^{\alpha}(j|i)g^d(j) + f(i,\alpha) : \alpha \in \mathcal{A}(i)\right\},\end{aligned} \tag{10}$$

where $\mathcal{A}(i) \subseteq \mathcal{A}$ is the set of actions available for state $i \in \mathcal{S}$. In words, we choose the action such that the expected potential after the next transition with this action is maximized. Let h be the policy determined by (10). We have $\eta^h > \eta^d$ if P^d is not the optimal policy; however, h may not be optimal even if (10) holds.

The above discussion leads to the following policy iteration algorithm:

(1) Guess an initial policy d_0, set $k = 0$.
(2) (Policy evaluation) Obtain the potential g^{d_k} by solving the Poisson equation $(I - P^{d_k})g^{d_k} + \eta^{d_k} e = f^{d_k}$.
(3) (Policy improvement) Choose

$$d_{k+1} \in \arg\left\{\max{}_{d \in \mathcal{D}}[f^d + P^d g^{d_k}]\right\}, \tag{11}$$

component-wisely (i.e., to determine an action for each state). If in state i, action $d_k(i)$ attains the maximum, set $d_{k+1}(i) = d_k(i)$.
(4) If $d_{k+1} = d_k$, stop; otherwise, set $k := k + 1$ and go to step 2.

In the algorithm, we need to choose actions independently at each state. It is easy to prove that this algorithm outputs an optimal policy (Puterman, 1994; Cao, 2007). The fundamental quantity in (9) is the performance potential. From a learning point of view, we need to analyze the behavior of a system under one policy to "learn" its potential of each state to determine how to make the system perform better. Potential is equivalent to the bias or the relative cost in the MDP literature, up to an additive constant. We use the word "potential" because of its physical meaning. Roughly speaking, the performance potential of a state i, $i \in \mathcal{S}$, measures the "potential" contribution of the state i to the system performance; the difference between the potentials of two states measures the effect of a jump (perturbation) from one state to another state on the system performance; and to improve the performance, in any state we should choose the action that leads to the best expected potential with this action (i.e., the largest $P^h g^d + f^h$ in (9)).

The optimality equation follows easily from the performance difference formula (9): a policy $\widehat{d}$ is optimal if and only if

$$\eta^{\widehat{d}} e + g^{\widehat{d}} = \max_{d \in \mathcal{D}} \{ f^d + P^d g^{\widehat{d}} \}. \qquad (12)$$

The only difference between (9) and (8) is that π^d in (8) is replaced by π^h in (9). This leads to an interesting observation: *policy iteration in MDPs in fact chooses the policy with the steepest directional derivative as the policy in the next iteration.*

4.5. A complete theory for MDPs with the long-run average criteria

As shown above, with the performance difference formula (9), the policy iteration procedure for ergodic chains can be derived simply and intuitively. This sensitivity-based approach also applies to multi-chain Markov systems, systems with absorbing states, and problems with other performance criteria such as the discounted performance and even the bias. The idea that many results can be derived simply from the performance difference formulas is further verified by the recently proposed approach with the n th-bias optimality (Cao, 2007). Essentially, starting with the performance difference formulas, we can develop a simple and direct approach to derive the results that are equivalent to the sensitive discount optimality for multi-chain Markov systems with long-run average criteria (Puterman, 1994; Veinott, 1969); and no discounting is needed. In this approach, a new concept called the n th-bias (and the n th bias optimality) is introduced.

The main results about the n th-bias optimality are as follows. The transition probability matrix of a multi-chain takes the following *canonical form*:

$$P = \begin{bmatrix} P_1 & 0 & 0 & \cdots & \cdot & 0 \\ 0 & P_2 & 0 & \cdots & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & P_m & 0 \\ R_1 & R_2 & R_3 & \cdots & R_m & R_{m+1} \end{bmatrix}, \qquad (13)$$

where $P_1, P_2, \ldots, P_m$ are all irreducible square matrices. This form indicates that the state space of a Markov chain consists of m closed subsets of recurrent states; each subset corresponds to one of the sub-matrices P_k, $k = 1, 2, \ldots, m$. The states corresponding to the last row, $R_1, R_2, \ldots, R_{m+1}$, are transient.

In this formulation, a policy is still denoted as (P, f), where f is the reward function. The long-run average reward is called the 0th *bias*, which is defined as a vector g_0 with components

$$g_0(i) := \eta(i) = \lim_{L \to \infty} \frac{1}{L} E\left\{ \sum_{l=0}^{L-1} f(X_l) | X_0 = i \right\}, \quad i \in \mathcal{S},$$

where $\{X_l, l = 0, 1, \ldots\}$ is a sample path of the Markov chain with P. The performance depends on the initial state i. The *bias* or the *1st bias* is denoted as $g_1 := g$, its ith component is

$$g_1(i) := g(i) = \sum_{l=0}^{\infty} E[f(X_l) - \eta(i) | X_0 = i].$$

The n th *bias*, $n > 1$, of policy (P, f) is defined as a vector g_n whose ith component is

$$g_n(i) = -\sum_{l=0}^{\infty} E[g_{n-1}(X_l) | X_0 = i], \quad n > 1.$$

The n th bias, $n \ge 0$, associated with a policy $d \in \mathcal{D}$ (with (P^d, f^d)) is denoted as g_n^d.

A policy $\widehat{d}$ is said to be *gain* (0th *bias*) *optimal* if

$$g_0^{\widehat{d}} \ge g_0^d, \quad \text{for all } d \in \mathcal{D}.$$

Let $\mathcal{D}_0$ be the set of all gain-optimal policies. A policy $\widehat{d}$ is said to be n th-*bias optimal*, $n > 0$, if $\widehat{d} \in \mathcal{D}_{n-1}$ and

$$g_n^{\widehat{d}} \ge g_n^d, \qquad \text{for all} \qquad d \in \mathcal{D}_{n-1}, \quad n > 0.$$

Let $\mathcal{D}_n$ be the set of all n th-bias optimal policies in $\mathcal{D}_{n-1}$, $n > 0$.

The sets $\mathcal{D}, \mathcal{D}_0, \mathcal{D}_1, \ldots$, are illustrated in Fig. 5. Our goal is to find an n th bias optimal policy in $\mathcal{D}_n$, $n = 0, 1, \ldots$. Following the sensitivity-based view, we start with the performance difference formulas for any two n th bias optimal policies, $n = 0, 1, \ldots$; these formulas can be easily derived. They are (Cao, 2007)

$$g_0^h - g_0^d = (P^h)^* [(f^h + P^h g_1^d) - (f^d + P^d g_1^d)] + [(P^h)^* - I] g_0^d, \qquad (14)$$

where for any policy P, we define

$$P^* = \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} P^l.$$

If $g_0^h = g_0^d$, then

$$g_1^h - g_1^d = (P^h)^* (P^h - P^d) g_2^d + \sum_{k=0}^{\infty} \{ (P^h)^k [(f^h + P^h g_1^d) - (f^d + P^d g_1^d)] \}. \qquad (15)$$

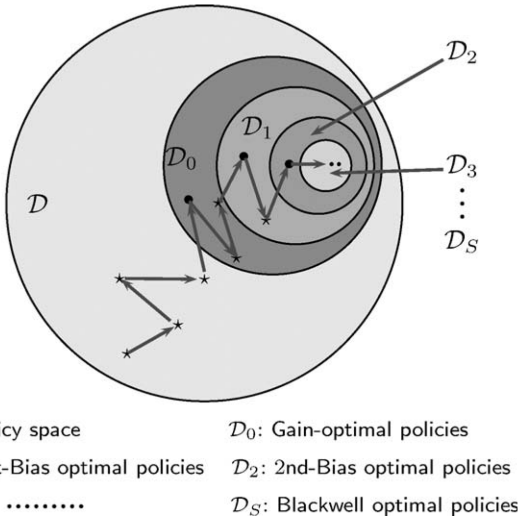
$\mathcal{D}_2$
$\mathcal{D}_1$
$\mathcal{D}_0$
$\mathcal{D}_3$
$\mathcal{D}$
$\mathcal{D}_S$
$\mathcal{D}$: Policy space
$\mathcal{D}_0$: Gain-optimal policies
$\mathcal{D}_1$: 1st-Bias optimal policies
$\mathcal{D}_2$: 2nd-Bias optimal policies
$\mathcal{D}_S$: Blackwell optimal policies

Fig. 5. Policy iteration for n th-bias and blackwell optimal policies.

If $g_n^h = g_n^d$ for a particular $n \geq 1$, then

$$g_{n+1}^h - g_{n+1}^d = (P^h)^*(P^h - P^d)g_{n+2}^d + \sum_{k=0}^{\infty}\{(P^h)^k(P^h - P^d)g_{n+1}^d\}. \quad (16)$$

Indeed, all the following results can be obtained by simply exploring and manipulating the special structures of these performance difference formulas. For details, see (Cao, 2007)

(1) Choose any policy $d_0 \in \mathcal{D}$ as the initial policy. Applying the policy iteration algorithm, we may obtain a gain (0th bias) optimal policy $\widehat{d}_0 \in \mathcal{D}_0$.
(2) Staring from any n th bias optimal policy $\widehat{d}_n \in \mathcal{D}_n$, $n = 0, 1 \ldots$, applying a similar policy iteration algorithm we may obtain an $(n+1)$ th bias optimal policy $\widehat{d}_{n+1} \in \mathcal{D}_{n+1}$.
(3) If a policy is S th bias optimal, with S being the number of states, it is also n th bias optimal for all $n > S$; i.e., $\mathcal{D}_S = \mathcal{D}_{S+1} = \mathcal{D}_{S+2} = \ldots$.
(4) An S th bias optimal policy is a *Blackwell* optimal policy.
(5) The optimality equations for n th bias optimal polices, both necessary and sufficient, can be derived from the performance difference formulas (14)–(16).

The sensitivity-based view provides a unified approach to all these MDP-types of optimization problems; and the basic principle behind this approach is surprisingly simple and clear: all these results can be derived simply by a comparison of the performance of any two policies.

4.6. Reinforcement learning (RL)

The fundamental model for systems in RL is also the Markov chain. While MDP is basically an analytical approach, which assumes that all the parameters are known, RL is a simulation-based (or in some cases, on-line) learning approach. Simulation produces a sample path of a system and can be carried out by following the system structure (e.g., the queueing structure) with out knowing the state transition probability matrix.

If we know enough information about the transition probabilities to implement policy iteration with potentials, we need only to "learn", or to estimate, the potentials g^{d_k} for all the states from a sample path of the system under one policy d_k and then update the policies iteratively according to (11). In this sense, any estimation-based or on-line approach for estimating potentials belongs to RL. In this regard, many efficient RL algorithms, such as TD(λ) (Bertsekas, 1995, 2001, 2007; Sutton & Barto, 1998), and approximate approaches, such as *neuro-dynamic programming* (Bertsekas, 1995, 2001, 2007; Bertsekas & Tsitsiklis, 1996), have been developed.

If we do not know exactly the transition probabilities, we cannot implement policy iteration even if we know the potentials. In this case, we need to learn the system behavior for all state-action pairs. Basically, in state i, we need to try all the actions in $\mathcal{A}(i)$ in order to get enough information for comparison. Therefore, this type of RL approach (e.g., Q-learning) requires a sample path that visits all the state-action pairs.

In such approaches, we consider a variant of the potential $g^d(i)$, called the *Q-factor* of a state-action pair (i, α), denoted as $Q^d(i, \alpha)$ for any $i \in \mathcal{S}$ and $\alpha \in \mathcal{A}(i)$. $Q^d(i, \alpha)$ is defined as the average potential of state i if action $\alpha \in \mathcal{A}(i)$ (not necessarily $d(i)$) is taken at a particular time and the rest of the Markov chain is run under a policy d:

$$Q^d(i, \alpha) = \sum_{j=1}^{S} p^{\alpha}(j|i)g^d(j) + f(i, \alpha) - \eta^d, \quad \alpha \in \mathcal{A}(i).$$

With this definition, (10) becomes

$$Q^d(i, h(i)) = \max_{\alpha \in \mathcal{A}(i)}\{Q^d(i, \alpha)\}$$

Thus, we may implement policy iteration by choosing the action that leads to the largest $Q^d(i, \alpha)$ in state i as $h(i)$ in the improved policy h.

Sample-path-based algorithms may be developed to estimate Q-factors. This leads to the Q-factor-based policy iteration, which can be used when the Markov chain's transition probability matrix is unknown. This approach in fact estimates the combined effect of the transition probabilities $p^{\alpha}(j|i)$ and the potentials $g^d(j)$ together without estimating these items separately.

In the approach, we need a sample path that visits all the state-action pairs. However, with a deterministic policy d, only the state-action pairs $(i, d(i))$, $i \in \mathcal{S}$, are visited. This issue may be resolved by introducing, with a small probability, other actions into the system as follows: in any state i, we apply action $d(i)$ with probability $1 - \epsilon$ and any other action $\alpha \in \mathcal{A}(i)$ randomly with an equal probability $\epsilon/(|\mathcal{A}(i)| - 1)$, $0 < \epsilon < < 1$, where $|\mathcal{A}(i)|$ denotes the number of actions in set $\mathcal{A}(i)$. We denote such a policy as d_{ϵ}.

In recent years, performance gradient-based optimization has attracted more and more attention from the RL community. Sample-path-based algorithms can be developed for performance gradients (Baxter & Bartlett, 2001; Baxter et al., 2001; Cao, 2005, 2007; Cao & Wan, 1998) these algorithms are based on the performance derivative formula (7).

In summary, the RL approach focuses on algorithms estimating potentials and its variant Q-factors, or the potentials and Q-factors for optimal policies, and the performance gradients.

4.7. Identification and adaptive control

Identification and adaptive control are well-developed areas. In adaptive control theory, system dynamics are modeled by differential or difference equations that determine the system structure. With such a mathematical model, elegant analysis can be carried out, leading to widely deployed adaptive control algorithms. When the system parameters are unknown and/or time varying, they need to be estimated from observations (this is also called system identification), and performance optimization can be achieved by using the adaptive control algorithms with the parameters estimated from observations.

A stochastic system under control, although it has its special structure, can be generally modeled as a Markov process, with the control variables viewed as actions. Consider a (discrete-time) linear stochastic control system modeled as

$$X_{l+1} = AX_l + Bu_l + \xi_l, \qquad l = 0, 1, \ldots, \quad (17)$$

where X_l is the system state at time l, which is usually a random vector, u_l is a vector of control variables, ξ_l is a vector of noise, and A and B are matrices with appropriate dimensions. Apparently, $\boldsymbol{X} = \{X_l, l = 0, 1, \ldots\}$ is a Markov chain and u_l can be viewed as the actions that determine the transition probabilities of $\boldsymbol{X}$, based on the probability distribution function of ξ_l. The problem is how to choose u_l, $l = 0, 1 \ldots$, such that a performance measure is maximized.

Principally, such a problem is amenable to either MDPs, RL, or PA. For example, we can apply policy iteration to find the optimal feedback control policy $u_l = d(X_l)$. Indeed, for the linear stochastic control problem in (17) with a quadratic reward function and a long-run average performance, we can derive, with policy iteration, the famous Riccati optimality equation for optimal policies (Cao, 2007).

When the system parameters are unknown, the basic quantities such as potentials and Q-factors have to be learned from a sample path with various RL algorithms. When the system parameters vary with a slow time scale, the policies have to be updated frequently to keep up with the parameter changes. In this sense, the on-line policy iteration, RL, or PA-based optimization are equivalent to system identification and adaptive control.

Another feature is that with policy iteration, we estimate the potentials and Q-factors, and the system parameters may not need to be estimated. This corresponds to the direct adaptive control in the literature, where the parameters for the optimal control law, instead of those for the system, are identified (Åström & Wittenmark, 1989).

One advantage of the on-line or sample-path-based approach is that, from the learning point of view, principally it applies to both linear and non-linear systems in the same way. The system structure affects only the transition probability matrix. However, determining the transition probability matrix for different control parameters might be a difficult task. There are many works in this direction (Werbos, 1992).

4.8. Continuous-time and continuous-state systems

So far we have shown that the sensitivity-based view provides a unified framework for different disciplines in learning and optimization of stochastic systems, and we mainly discussed the discrete event dynamic systems with a discrete-time and discrete-state model. Our recent on-going research shows that this sensitivity-based view also works well for the stochastic control problem with continuous-time and continuous-state (CTCS) systems, and even for the singular stochastic control problem that models the portfolio management problem in financial engineering.

The simplest model for CTCS dynamic system is by the stochastic differential equation

$$dX(t) = \alpha(X(t))dt + \sigma(X(t))dW(t), \qquad (18)$$

where $W(t)$ is a standard Brownian motion, and $X(t)$ is a CTCS Markov process defined on $t \in [0, \infty)$ and state space $\mathcal{R}$ (the space of real numbers).

In general, we consider a continuous-time Markov process $\boldsymbol{X} = \{X(t), t \in [0, \infty)\}$ with a continuous state space $\mathcal{S} = \mathcal{R}^n$. Let $\mathcal{B}$ be the σ-field of $\mathcal{R}^n$ containing all the Lebesgue measurable sets. Let the system state at time t be $X(t) = x \in \mathcal{R}^n$. The probability that the state at time $t' \geq t$, $X(t')$, lies in a set $B \in \mathcal{B}$ can be denoted as a set function $P_{t,t'}(B|x)$ with $P_{t,t'}(\mathcal{R}^n|x) = \int_{\mathcal{R}^n} P_{t,t'}(dy|x) = 1$ for all $x \in \mathcal{R}^n$. For any $t \in [0, \infty)$ and $t' \geq t$, $P_{t,t'}(B|x)$ is called a *state transition probability function*, which is a function from $\mathcal{R}^n \times \mathcal{B}$ to $[0, 1]$ satisfying the following conditions: for any given $x \in \mathcal{R}^n$, $P_{t,t'}(B|x)$ is a probability measure on $\mathcal{B}$, and for any $B \in \mathcal{B}$, $P_{t,t'}(B|x)$ is a Lebesgue measurable function. Let $\Delta t = t' - t \geq 0$. We write $P_{t,t'}(B|x) = P_{t,t+\Delta t}(B|x)$. For time-homogenous systems, $P_{t,t+\Delta t}(B|x)$ is independent of t and we denote $P_{t,t+\Delta t}(B|x) := P_{\Delta t}(B|x)$.

With this definition, any Lebesgue measurable set is also measurable with any $P_{\Delta t}(B|x)$ for any $x \in \mathcal{R}^n$. Without specifically mentioning, we will assume that all the sets and the functions discussed in this paper are Lebesgue measurable.

Let $P_{\Delta t}(B|x)$ be a transition probability function and $h(x)$ be any function. We define a linear (right) operator on the space of integrable functions, $P_{\Delta t}$: $h \to P_{\Delta t}h$, as follows:

$$(P_{\Delta t}h)(x) := \int_{R^n} h(y) P_{\Delta t}(dy|x).$$

If there is no confusion, we will use a generic notation "P" for operators and the corresponding transition probability functions.

Define $e(x) = 1$ for all $x \in \mathcal{R}^n$. For any transition function P, we have $(Pe)(x) = 1$ for all $x \in \mathcal{R}^n$. Thus, we can write $Pe = e$. Define the n-dimensional identity function I:

$$I(B|x) = \begin{cases} 1 & \text{if } x \in B, \\ 0 & \text{otherwise.} \end{cases}$$

The corresponding operator I is the *identity operator*: $(Ih)(x) = h(x)$, $x \in \mathcal{R}^n$, for any function h; and we have $P_{\Delta t=0}(B|x) = I(B|x)$ for any $x \in \mathcal{R}^n$.

For any probability measure $\nu(B)$ on $\mathcal{B}$, we define another probability measure, denoted as νP, by

$$(\nu P)(B) := \int_{\mathcal{R}^n} \nu(dx) P(B|x), \quad B \in \mathcal{B}.$$

For any probability measure $\nu(B)$, we use $e\nu := e(x)\nu(B)$ to denote a transition probability function which equals $\nu(B)$ for all $x \in \mathcal{R}^n$. A probability measure $\pi(B)$ is called the steady-state probability measure of the transition function $P_{\Delta t}(B|x)$, or its corresponding Markov process, if for all $x \in \mathcal{R}^n$ and $B \in \mathcal{B}$

$$\lim_{\Delta t \to \infty} P_{\Delta t \to \infty}(B|x) = e(x)\pi(B).$$

The long-run average performance is defined as

$$\eta(x) = \lim_{T \to \infty} \frac{1}{T} E\left\{ \int_0^T f(X(t))dt | X(0) = x \right\},$$

where f is the reward function. The performance potential is defined as

$$g(x) := \lim_{T \to \infty} E\left\{ \int_0^T [f(X(t)) - \eta] dt | X(0) = x \right\}.$$

From this, we may derive the Poisson equation

$$-Ag(x) + \eta(x) = f(x),$$

where

$$A := \lim_{\Delta t \to 0} \frac{P_{\Delta t} - I}{\Delta t} \equiv \frac{\partial P_t}{\partial t}|_{t=0}$$

is called the *infinitesimal generator* of the Markov process.

It is well known that for the stochastic process in (18) the infinitesimal generator A has the following property: for any twice differentiable function $f(x)$, it holds

$$Af(x) = \alpha(x) \frac{\partial f}{\partial x}(x) + \frac{1}{2}\sigma^2(x) \frac{\partial^2 f}{\partial x^2}(x). \qquad (19)$$

We have $\pi A = 0$, if the steady-state probability measure π exists. From (19), we can easily verify that for the stochastic process in (18) the steady-state probability density function $\pi(x)$ satisfies the following Fokker–Planck equation:

$$-\frac{\partial}{\partial x}[\alpha(x)\pi(x)] + \frac{1}{2} \frac{\partial^2}{\partial x^2}[\sigma^2(x)\pi(x)] = 0. \qquad (20)$$

Now, we consider two policies and let $\{A', f'\}$ and $\{A, f\}$ be the corresponding infinitesimal generators and reward functions, and η' and η be their long-run average performance, g be the potential of the policy $\{A, f\}$, and π' be the steady-state probability measure of $\{A', f'\}$. We may easily derive the performance difference formula

$$\eta' - \eta = \pi'[(f' + A'g) - (f + Ag)]. \qquad (21)$$

Policy iteration algorithms for an optimal policy can be derived from (21). In addition, we may establish the optimality equation as

$$\max_{u \in U}\{A^u g^{\widehat{u}} + f^u\} = A^{\widehat{u}} g^{\widehat{u}} + f^{\widehat{u}} = \eta^{\widehat{u}}, \qquad (22)$$

in which $u(x)$ (which controls $A(B|x)$ and $f(x)$) denote a policy, U denote the policy space, and $\widehat{u}$ denote an optimal policy. (22) takes the same form as (12). For more details, see a forthcoming paper Cao (2009). Our recent research also indicates that this sensitivity-based approach also applies to the singular stochastic control problems appeared in the portfolio management problem in financial engineering.

Therefore, the sensitivity based view provides a unified framework for both DEDS and CTCS systems. In addition, this sensitivity-based view will bring some new insight to the CTCS problems. For example, the gradient-based learning and event-based optimization (see the discussion in the next section) that were orginally developed for DEDS may lead to new research topics for performance optimization of CTCS systems.

4.9. A sensitivity-based view of learning and optimization

In summary, the fundamental limitations of learning and optimization sketch out the directions of developing efficient and widely applicable learning and optimization approaches; these approaches may require as little information about the system structure as possible. There are two feasible directions: First, because we can only learn from one policy at a time, we may at most obtain local (in the neighborhood of a policy) information in the policy space; this observation leads to the approaches for estimating performance gradients or derivatives. Second, because we can only compare two policies at a time, we may start with the performance difference formulas of any two policies in developing learning and optimization methods. This observation leads to policy-iteration based approaches and optimality equations. In short, these directions can be characterized by performance derivatives and performance differences, respectively. We say that we take a sensitivity-based view in these approaches (Cao, 2007).

These two directions lead to two types of approaches. The first type of approach is based on *perturbation analysis* (PA). With PA, we can obtain the performance derivatives with respect to the system's parameters. We can develop gradient-based optimization approaches using PA. This approach applies to problems where policy spaces are parameterized with continuous parameters. The basic idea is shown in Fig. 6 A. We first set the parameter θ to be any value and determine the performance gradient at θ with PA. Then we change θ slightly along the direction of the gradient to $\theta+\Delta\theta$ and determine the gradient again at this $\theta+\Delta\theta$. We repeat this procedure until reaching a point $\widehat{\theta}$ at which the performance gradient is zero; this is a local optimal point. The performance gradient can be calculated analytically, or estimated from a sample path. Because the gradient estimates contain noise, stochastic approximation techniques can be used in the optimization procedure (Marbach & Tsitsiklis, 2001).

The second type of learning and optimization approach is based on the comparison of the system performance measures of two

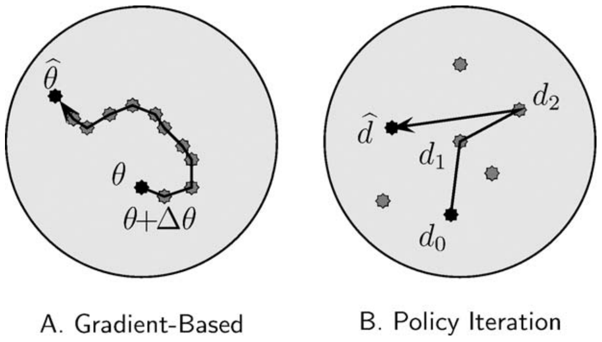

Fig. 6. Two types of optimization approaches.

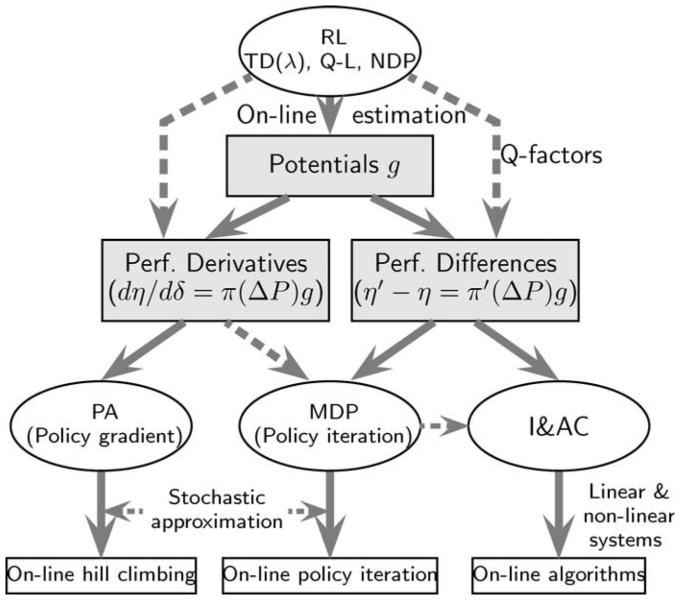

Fig. 7. A map of the learning and optimization world (PA: perturbation analysis, MDP: Markov decision process, RL: reinforcement learning, Q-L: Q-learning, I&AC: identification and adaptive control, and NDP: neuro-dynamic programming).

different policies. The approach strongly depends on the system structure. A well-known result in this direction is: When the actions taken in different states are independent, it may be possible to use the information learned by observing or analyzing the system behavior under the current policy to determine a policy under which the performance of the system is better, if such a policy exists. This leads to the *policy iteration* procedure shown in Fig. 6 B. We start with any policy d_0, learn from its behavior and find a better policy d_1, then learn from d_1 and find a better policy d_2, and so on until the best policy $\widehat{d}$ is reached.

4.10. A map of the learning and optimization world

With a sensitivity point of view, the world of learning and optimization can be illustrated by the map shown in Fig. 7. The central piece of the map is the performance potential. Various RL methods yield sample-path-based estimates for potentials g, or their variant Q-factors, or their values for the optimal policy; the potentials are used as building blocks in constructing the two performance sensitivity formulas; these two formulas form the basis for gradient-based (PA-type) and policy-iteration-type optimization approaches; RL methods can also be developed for directly estimating the performance gradients on sample paths; stochastic approximation techniques can be used to derive efficient optimization algorithms with the sample-path-based gradient estimates, and to derive on-line policy iteration algorithms. Both the gradient-based approaches and policy iteration can be applied to system identification and adaptive control (I&AC) problems, even with non-linear systems. This same map also applies to the CTCS systems.

5. Event-based optimization and potential aggregation

We have introduced a sensitivity-based view of learning and optimization. In the framework, systems are modeled by Markov processes. However, it is well known that the Markov model suffers from the following disadvantages:

(1) The state space and the policy space are too large for most problems.

(2) The MDP policy iteration theory requires the independent action assumption.
(3) The model does not utilize any special feature of the system.

Now, we show that with the sensitivity-based view, we can develop new learning and optimization approaches that utilize the special features of the systems to overcome or alleviate the above difficulties.

One of such approaches is the *event-based optimization*, which can be applied to systems where the actions can be taken only when some events happen.

5.1. The main features of the event-based approach

We first give a simple example to illustrate the ideas.

Example 1: A robot takes a random walk in a five-room maze shown in Fig. 8. The numbers in the parenthesis indicate the rewards that the robot gets in each room. The robot moves from room 0 to the two top rooms 1 or 2 with probability p, and to the two bottom rooms 3 or 4 with probability $q = 1 - p$. There is a traffic light in each passage leading from room 0 to the top or the bottom rooms; if it is red, the robot moves to the left rooms 1 (if it moves to the top) or 3 (if it moves to the bottom), and if it is green, the robot moves to the right room 2 or 4. We may control the probability of the lights being red, α, or being green, $1 - \alpha$. We assume that the two traffic lights must be in the same color; i.e., α is the same for both lights. The system can be modeled by a Markov chain. Fig. 9 illustrates a part of a system's state transition diagram, and Fig. 10 lists the transition probabilities of state 0 when a particular α is chosen.

Let us analyze the structure of the transition diagram. From Fig. 9, if the system moves from state 0 to the two top states, 1 and 2, we need to take the biggest value $\alpha = 1$ to reach state 1 with probability 1 and get a reward of 100; on the contrary, if the system moves from state 0 to the two bottom states, 3 and 4, we need to take the smallest value $\alpha = 0$ to reach state 4 with probability 1 and get a reward of 100. Thus, at state 0, a big α is good for the top, but bad for the bottom, and vice versa. When $p = q = 0.5$, for any α the average reward at the next step is zero. Therefore, the state-based optimal policy may not be very good.

However, the situation improves significantly if we know a bit of information about the state transition. From the structure shown in Fig. 9, the top two transitions, or the bottom two transitions, have similar structural properties. This structure can be captured by aggregating these transitions together and defining two events:

$a := \{\langle 0,1\rangle, \langle 0,2\rangle\}$ and $b := \{\langle 0,3\rangle, \langle 0,4\rangle\}$,

where $\langle i,j\rangle$ denotes a transition from state i to state j, $i, j \in \mathcal{S}$. These two sets of state transitions aggregated into two events are shown as the two ovals, a and b, in Fig. 9; they are also illustrated by the two thick boxes in Fig. 10.

With this formulation, if event a occurs, the system moves to state 1 with probability α and to state 2 with probability $1 - \alpha$; and if event b occurs, the system moves to state 3 with probability α and to state 4 with probability $1 - \alpha$. In the event-based approach, we assume that we can observe the events, not the states; i.e., at any time instant l, we can observe whether $\langle X_l, X_{l+1}\rangle \in a$, or $\langle X_l, X_{l+1}\rangle \in b$, occurs. We need to determine an event-based policy that specifies the probability α given event a or b: $\alpha_a = d(a)$ and $\alpha_b = d(b)$.

From the reward structure shown in Fig. 9, we may design a myopic policy: if a occurs, we choose the largest value, i.e., $\alpha_a = 1$, which leads to state 1 and the reward at the next step is 100; and similarly, if b occurs, we choose the smallest value, i.e., $\alpha_b = 0$, and the state at the next step is 4 and the reward is also 100. In this example, this myopic event-based policy is better than the optimal state-based MDP policy.

This example shows that an optimal event-based policy may be better than an optimal state-based policy; or knowing the event is better than knowing the state. This is because knowing the event implies knowing something about the current transition, which, strictly speaking, contains information about the future (the next state). In addition, we can see that a history-independent event-based policy is good enough in this example.

Many real-world systems fit the event-based formulation.

Example 2: (Admission control) Consider a communication system modeled as a variant of an open network shown in Fig. 11. Packets are called customers in queueing terminology. The network consists of M servers; the customers' service times at server i are identically and independently distributed with an exponential distribution with mean $1/\mu_i$, $i = 1, 2, \ldots, M$. After the completion of its service at server i, a customer will join the queue

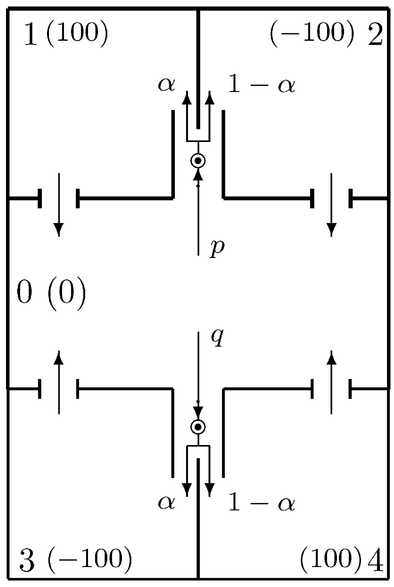

Fig. 8. Random walk of a robot.

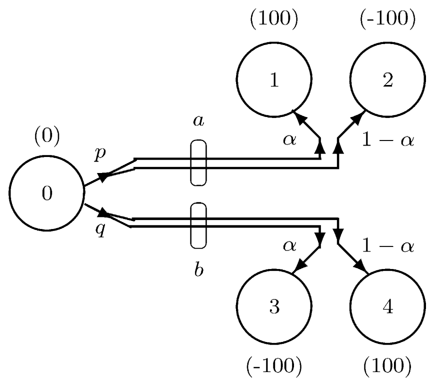

Fig. 9. The state transition diagram of the random walk.

	0	1	2	3	4
0	0	$p\alpha$	$p(1-\alpha)$	$q\alpha$	$q(1-\alpha)$

Event a *Event b*

Fig. 10. The transition probabilities of state 0 when action α is taken in Example 1.

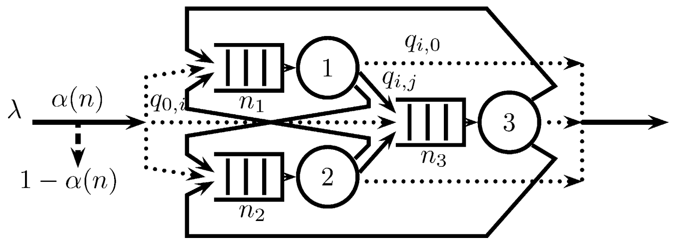

Fig. 11. The admission control problem.

at server j with probability $q_{i,j}$, and will leave the network with probability $q_{i,0}$, $i,j=1,2,\dots,M$. We have $\sum_{j=0}^{M} q_{i,j}=1$, $i=1,2,\dots,M$. Let n_i be the number of customers at server i, and $n=\sum_{i=1}^{M} n_i$ be the population of the system.

The customers arrive at the network in a Poisson process with rate λ. If an arriving customer finds n customers in the network, the customer will be admitted to the system with probability $\alpha(n)$ and will be rejected and leave the system with probability $1-\alpha(n)$, $0\le\alpha(n)\le 1$. We assume that the system has a capacity of N; i.e., $\alpha(N)=0$, or an arriving customer finding N customers in the system will be dropped. An admitted customer will join queue i with probability $q_{0,i}$, $i=1,2,\dots,M$, $\sum_{i=1}^{M} q_{0,i}=1$.

The system can be modeled as a discrete-time Markov chain embedded at the transition times. The system state is $\boldsymbol{n}=(n_1,n_2,\dots,n_M)$. The optimization problem is to find the best admission probabilities $\alpha(n)$, $n=0,1,\dots,N-1$, such that the system performance (discounted, long-run average, etc.) is optimized. In this problem, an action is taken only when a customer arrives at the network; we call it an *event* of a customer arrival (which can be precisely defined as a set of transitions). When a customer arrives, the system can be in many different states $\boldsymbol{n}$'s. Thus, the problem is not a standard MDP, since in this problem an action may affect the transition probabilities of many states. In addition, the decision depends on events, rather than on states.

With the sensitivity-based view, the solutions to the event-based optimization problems can be derived from the performance sensitivity formulas. It is easy to derive the performance difference formula for the random walk example (with "$'$" denoting the quantities for any other policy):

$$\eta'-\eta=\pi'(a)[(\alpha'_a-\alpha_a)g(a)]+\pi'(b)[(\alpha'_b-\alpha_b)g(b)], \tag{23}$$

where $\pi(a)$ and $\pi(b)$ are the steady-state probabilities of events a and b, and

$$g(a)=g(1)-g(2),\quad g(b)=g(3)-g(4), \tag{24}$$

are the potentials of events a and b, which are aggregated from potentials $g(1)$, $g(2)$, and $g(3)$, $g(4)$, respectively, according to the structure of the problem.

For the admission control problem, we have

$$\eta'-\eta=\sum_{n=0}^{N-1}\pi'(n)[(\alpha'(n)-\alpha(n))]d(n), \tag{25}$$

where $\pi(n)$ is the steady-state probability of the event that a customer arrives and finds a population of n, and (let $\boldsymbol{n}_{+i}=(n_1,\dots,n_i+1,\dots n_M)$)

$$\begin{aligned} d(n) &= \frac{1}{\pi(n)}\left\{\sum_{i=1}^{M} q_{0i}\left[\sum_{n_1+\cdots+n_M=n}\pi(\boldsymbol{n})g(\boldsymbol{n}_{+i})\right]\right.\\ &\quad\left.-\left[\sum_{n_1+\cdots+n_M=n}\pi(\boldsymbol{n})g(\boldsymbol{n})\right]\right\}, \end{aligned} \tag{26}$$

is the potential aggregated according to the event structure.

Policy iteration algorithms can be developed from (23) and (25). Furthermore, performance derivative formulas can be easily derived from (23) and (25); therefore, gradient-based optimization algorithms can also be developed. The event-based potentials $g(a)$ and $g(b)$ in (24) and $d(n)$ in (26) can be estimated on a sample path of the systems, and learning algorithms can be developed.

The number of events is usually much smaller than the number of states. In the admission control problem, the number of states grows exponentially with the system size N; however, the number of events, $N+1$, is linear in the system size N.

In summary, the event-based approach has the following advantages:

(1) Events may contain future information and an event-based policy may perform better than state-based policies.
(2) The potentials of events are aggregated from potentials of states; the number of event-based policies may be scale to the system size, and thus the event-based approach may save computation.
(3) With an event-based policy, the same action is taken at different states that correspond to the same event; thus, the event-based approach applies to problems in which the independent-action assumption does not hold.

The world of the event-based learning and optimization can be described by a map similar to Fig. 7, in which the potentials g are replaced by the aggregated potentials of the events, and the performance difference and derivative formulas are replaced by those for the event-based policies. From these formulas, gradient-based approaches, and under some conditions (which are satisfied by the two examples above, but not always so in general!) policy iteration algorithms can be developed (Cao, 2007). The difference and derivative formulas can be "constructed" on a sample path with intuition by using potentials as building blocks (Cao, 2007). Reinforcement learning and other algorithms can also developed. Many of these topics require further study.

Many problems fit the event-based framework (a solution may not be easy, though!). For example, in POMDP, we may define an observation, or a sequence of observations, as an event. Other examples including state and time aggregations, hierarchical control (hybrid systems), options, and singular perturbation. Different events can be defined to capture the special features in these different problems. In this sense, the event-based approach may provide a unified view for possible solutions to these different problems (Cao, 2007).

6. Conclusion

We have shown that the world of learning and optimization of stochastic dynamic systems can be built upon the two performance sensitivity formulas. This sensitivity-based view provides a unified framework for existing approaches for both discrete-time discrete-state systems and continuous-time continuous-state systems, including PA, MDPs, RL, and stochastic control. In addition, this sensitivity-based view brings in new insights and leads to new results such as the n th bias optimality, gradient-based learning, and the time aggregation method. The derivations of these results, old or new, are intuitive clear. The sensitivity-based view also points to new research directions in the area of learning and optimization. For example, the event-based optimization approach, which has many advantages over the state-based approaches, may be developed with this view. Much work needs to be done in that direction.

References

Åström, K. J., & Wittenmark, B. (1989). *Adaptive control.*. Reading, MA: Addison-Wesley.
Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research, 15*, 319–350.

Baxter, J., Bartlett, P. L., & Weaver, L. (2001). Experiments with infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research, 15*, 351–381.

Bertsekas, D. P., & Wittenmark, B. (1995). *Dynamic programming and optimal control, Vols. I and II*. Belmont, MA: Athena Scientific.

Bertsekas, D. P., & Wittenmark, B. (2001). *Dynamic programming and optimal control, Vols. I and II*. Belmont, MA: Athena Scientific.

Bertsekas, D. P., & Wittenmark, B. (2007). *Dynamic programming and optimal control, Vols. I and II*. Belmont, MA: Athena Scientific.

Bertsekas, D. P., & Tsitsiklis, T. N. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

Cao, X. R. (1994). *Realization probabilities: The dynamics of queueing systems*. New York: Springer–Verlag.

Cao, X. R., & Wan, Y. W. (1998). Algorithms for sensitivity analysis of Markov systems through potentials and perturbation realization. *IEEE Transactions on Control System Technology, 6*, 482–494.

Cao, X. R. (2005). A basic formula for online policy gradient algorithms. *IEEE Transactions on Automatic Control, 50*(5), 696–699.

Cao, X. R. (2007). *Stochastic learning and optimization—A sensitivity-based approach*. New York: Springer.

Cao, X. R. (2009). Stochastic control via direct comparison, (submitted, to be appeared).

Cassandras, C. G., & Lafortune, S. (1999). *Introduction to discrete event systems*. Boston: Kluwer Academic Publishers.

Cassandras, C. G., Sun, G., Panayiotou, C. G., & Wardi, Y. (2003). Perturbation analysis and control of two-class stochastic fluid models for communication networks. *IEEE Transactions on Automatic Control, 48*(5), 770–782.

çinlar, E. (1975). *Introduction to stochastic processes*. Englewood Cliffs, NJ: Prentice Hall.

Ho, Y. C., & Cao, X. R. (1991). *Perturbation analysis of discrete-event dynamic systems*. Boston: Kluwer Academic Publisher.

Ho, Y. C., Zhao, Q. C., & Jia, Q. S. (2007). *Ordinal optimization: Soft optimization for hard problems*. New York: Springer.

Ho, Y. C., Zhao, Q. C., & Pepyne, D. (2003). The no free lunch theorem, complexity and computer security. *IEEE Transactions on Automatic Control, 48*, 783–793.

Hu, J. Q., Fu, M. C., & Marcus, S. I. (2007). A model reference adaptive search method for global optimisation. *Operations Research, 55*, 549–568.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Marbach, P., & Tsitsiklis, T. N. (2001). Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control, 46*(2), 191–209.

Oksendal, B., & Sulem, A. (2007). *Applied stochastic control of jump diffusions*. Springer.

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York: Wiley.

Rubinstein, R. Y., & Kroese, D. P. (2004). *The cross-entropy method: A unified approach to combinational optimization, Monte-Carlo simulation and machine learning*. New York: Springer.

Shi, L., & Olafsson, S. (2000). Nested partitions method for global optimisation. *Operations Research, 48*, 390–407.

Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. *Computer, 27*, 17–26.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Veinott, A. F. (1969). Discrete dynamic programming with sensitive discount optimality criteria. *The Annals of Mathematical Statistics, 40*, 1635–1660.

Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. In D. A. White & D. A. Sofge (Eds.), *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches* (pp. 493–525). New York: Van Nostrand Reinhold.

Xi-Ren Cao received the M.S. and Ph.D. degrees from Harvard University, in 1981 and 1984, respectively, where he was a research fellow from 1984 to 1986. He then worked as consultant engineer/engineering manager at Digital Equipment Corporation, USA, until October 1993. Then he joined the Hong Kong University of Science and Technology (HKUST), where he is currently Chair Professor, and Director of the Research Center for Networking.He owns three patents in data- and tele-communications and published three books in the area of performance optimization and discrete event dynamic systems. He received the Outstanding Transactions Paper Award from the IEEE Control System Society in 1987, the Outstanding Publication Award from the Institution of Management Science in 1990, and the Outstanding Service Award from IFAC in 2008. He is a Fellow of IEEE, a Fellow of IFAC, and is/was the Chairman of IEEE Fellow Evaluation Committee of IEEE Control System Society, Editor-in-Chief of Discrete Event Dynamic Systems: Theory and Applications, Associate Editor at Large of IEEE Transactions of Automatic Control, and Board of Governors of IEEE Control Systems Society and on the Technical Board of IFAC. His current research areas include discrete event dynamic systems, stochastic learning and optimization, performance analysis of communication systems, signal processing, and financial engineering.