
Discrete Event Dynamic Systems: Theory and Applications, 15, 169–197, 2005

Basic Ideas for Event-Based Optimization of Markov Systems

XI-REN CAO* eecao@ee.ust.hk
Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Abstract. The goal of this paper is two-fold: First, we present a sensitivity point of view on the optimization of Markov systems. We show that Markov decision processes (MDPs) and the policy-gradient approach, or perturbation analysis (PA), can be derived easily from two fundamental sensitivity formulas, and such formulas can be flexibly constructed, by first principles, with performance potentials as building blocks. Second, with this sensitivity view we propose an event-based optimization approach, including the event-based sensitivity analysis and event-based policy iteration. This approach utilizes the special feature of a system characterized by events and illustrates how the potentials can be aggregated using the special feature and how the aggregated potential can be used in policy iteration. Compared with the traditional MDP approach, the event-based approach has its advantages: the number of aggregated potentials may scale to the system size despite that the number of states grows exponentially in the system size, this reduces the policy space and saves computation; the approach does not require actions at different states to be independent; and it utilizes the special feature of a system and does not need to know the exact transition probability matrix. The main ideas of the approach are illustrated by an admission control problem.

Keywords: perturbation analysis, Markov decision processes (MDPs), POMDPs, performance potentials, policy iteration, policy gradients, aggregation

1. Introduction

The research of this paper is a continuation of the recent research on performance optimization of discrete event dynamic systems with a sensitivity point of view (Cao, 1998; Cao and Chen, 1997; Cao, 2000, 2004a; Cao and Guo, in press; Cao and Wan, 1998). We present the results in a self-contained manner. The research is motivated by a number of previously established results: First, performance optimization of Markov systems are based on two fundamental sensitivity formulas, one for performance difference and the other for performance derivatives. Policy iteration in Markov decision processes (MDPs) can be developed easily from the performance difference formula (Cao, 1998, 2000; Cao and Guo, in press), and gradient-based optimization (perturbation analysis (PA) or policy gradient) is based on the performance derivative formula (Cao, 1998, 2000; Cao and Chen, 1997; Cao and Guo, in press; Cao and Wan, 1998). Second, both sensitivity formulas can be constructed, by first principles, using performance potentials as building blocks (Cao, 2004a); such construction is intuitive, flexible, and therefore can utilize the special feature of the system. Third, sample-path-based algorithms can be developed for estimating potentials or performance derivatives, and

* Supported in part by a grant from Hong Kong UGC.

for implementing policy iteration and policy-gradient based optimization (Baxter and Bartlett, 2001; Baxter et al., 2001; Cao, 1999; Cao and Wan, 1998; Chong and Ramadge, 1994; Cooper et al., 2003; Fang and Cao, 2004; Marbach and Tsitsiklis, 2001).

However, the standard Markov model-based formulation suffers from a number of drawbacks. First and foremost, the state space is usually too large for practical problems. That is, the number of potentials to be calculated or estimated is too large for most problems. Second, the generally applicable Markov model does not reflect any special structure of a particular problem. Thus, it is not clear whether and how potentials can be aggregated to save computation by exploring the special structure of the system. The third issue is related to policy iteration: it requires the actions at different states be chosen independently (we will call it the *independent-action assumption*). In many practical problems, however, these actions may have to be correlated; the standard policy iteration cannot handle such problems properly.

The sensitivity point of view and the flexible construction of the sensitivity formulas provide us a new perspective to explore alternative approaches for performance optimization of systems with some special features. In this paper, we propose to formulate the optimization problem based on "events" rather than on states. The approach is called *event-based optimization*. In a real world system, a physical event that happens at a particular time instant can be characterized by the state transition at that instant; e.g., if a customer arrives to a network at a particular instant, then the population of the network increases by one at that instant. Therefore, an event is defined as a set of state transitions. Furthermore, in many systems actions can be taken only when some events occur; e.g., in the admission control problem, actions can be taken only when a customer arrives. Thus, policies are defined on events rather than on states. The performance sensitivity formulas are constructed for event-based policies and optimization can be implemented based on these sensitivity formulas, in a way similar to the standard MDPs. This approach utilizes the special features of a system captured by the logical relations among different types of events. The potentials associated with an event can be aggregated, and computation is reduced. The independent-action assumption is not required because the same action can be chosen at many different states corresponding to the same event. In aggregating the potentials, we may use the special system structure (e.g., the queueing structure), and the explicit form of the transition probabilities of the underlying Markov system may not be needed.

The main concept can be clearly explained by the admission control problem in communication. Logically, the process of accepting or rejecting an arrival customer consists of three phases, which can be formulated as three types of events: First, a new customer arrives; this corresponds to a set of state transitions on the sample paths; this set of transitions forms an event which is observable and is called an observable event. After the customer arrives, an action (accept or reject) is taken, which partially determines the state transitions. The set of transitions determined by the action is called a controllable event because we may control the probabilities of the actions. Finally, the customer chooses its own destination in the network (when accepted) or leaves it (when rejected). The set of state transitions corresponding to this phase forms an event called the natural transition event, because the transitions are purely determined by the nature. These three phases have a logic order in timing but they happen simultaneously in the Markov model.

Associated with the observable event is some information about the network; a policy determines an action based on the information contained in the observable event, or in a history of the observable events. The optimization problem becomes to choose an event-based policy that maximizes the performance. As in the standard MDPs, the solution is based on the performance sensitivity formulas derived for any two policies in the event-based policy space, using the construction method with potentials as building blocks.

In Section 2, we briefly review the performance potentials, the two performance sensitivity (derivative and difference) formulas, and the construction method; with construction, we can derive the sensitivity formulas for many problems flexibly using the potentials as building blocks. In Section 3, we provide an overview for learning and optimization from a sensitivity point of view. We emphasize a fundamental fact: By observing and analyzing a system's behavior under a policy, one can in general obtain only the local information around that policy in the policy space, including the performance gradients. Policy iteration, on the other hand, depends heavily on the form of the performance difference formulas. Therefore, learning and optimization follow directly from the two basic performance sensitivity formulas: performance derivatives and performance differences. These two sections serve as the base for the event-based optimization proposed in Section 4. We use the admission control problem to present the main ideas. In Section 4.1, we define an event in a Markov system as a set of state transitions that satisfy some common properties. In Section 4.2, we classify events into three types: the observable, controllable, and natural transition events. A policy chooses an action based on the information contained in the observable events, and the action controls the probabilities of the controllable events, and the nature finalizes the state transition. In Section 4.3, we derive the performance sensitivity formulas for the admission control problem using the construction method; the event structure is utilized in potential aggregation. In Section 4.4, we show how the aggregated potentials can be estimated on a sample path without estimating potentials for every state. In Section 4.5, we show that with the sensitivity formulas derived in Section 4.3, the gradient-based optimization and event-based policy iteration can be developed for this admission control problem. In Section 5, we discuss the possible extensions of the approach.

There are a number of advantages of the event-based optimization. First, potentials can be aggregated by exploiting the event-based system structure, and sample-path-based estimation algorithms can be developed for the aggregated potentials. Furthermore, as shown in Section 4.4, estimating an aggregated potential on a sample path requires the same computation and achieves the same accuracy as estimating the potential of a state. This may significantly save computation and reduce the number of potentials to be estimated in the learning process. Despite the fact that the number of states usually grows exponentially with respect to the system size, the number of aggregated potentials depends on the number of observable events, which may scale to the system size. Thus, the event-based optimization saves considerably computations in performance optimization. Second, the approach applies to many practical problems where actions depend on events, not states; such problems do not fit well the standard MDP formulation, because the same action may be taken when the same event is observed, which may correspond to many different states. This violates the independent-action assumption. Third, the construction of the sensitivity formulas can be carried out by using the special structure

of the system (e.g., the queueing structure), and the performance sensitivity formulas thus obtained can be expressed in terms of the structural parameters rather than in the transition probabilities of the underlying Markov process. This provides structural insights and avoids the tedious effort associated with large state spaces. Fourth, for systems where the events requiring actions happen rarely, the approach is easy to implement on a sample path and is efficient. Finally, this approach may be applied to a number of subjects such as multilevel (hierarchical) control, state and time aggregation (Cao et al., 2002), options (Barto and Mahadevan, 2003), singular perturbation, and partially observed MDPs, etc, by formulating different events to capture the different features of these problems. Thus, it provides a unified framework to these different areas and opens up new research topics.

The limitation of the approach is that the aggregated potentials in the performance difference formula may depend on both policies under comparison; this may prevent the aggregated potentials from being used in policy iteration. It is shown in Section 4.3 that under some special conditions (including the admission control example studied in this paper), the aggregated potentials depend only on the original policy and can be estimated on a single sample path. In such cases, event-based policy iteration algorithms can be developed. In this regards, the approach clearly indicates whether in policy iteration the aggregated potentials can be estimated on the original sample path; and if not, why. It is clear, however, in performance derivative analysis, the aggregated potentials always depend only on the original policy. Therefore, in general, performance gradient-based optimization (with events) is more applicable than the event-based policy iteration.

2. Performance potentials and performance sensitivities

In this and the next sections, we briefly summarize the relevant results in sensitivity analysis with the standard MDP formulation that are scattered in a number of previous papers. We first review the main concepts in performance optimization, the performance potentials and the perturbation realization factors, in Section 2.1, and then review the two sensitivity formulas in Section 2.2, which is followed by a brief introduction to the construction method in Section 2.3.

Consider an ergodic (irreducible and aperiodic) finite Markov chain $\mathbf{X}$ on a state space $\mathcal{S} = \{1, 2, \cdots, S\}$ with transition probability matrix $P = [p(i, j)] \in [\,0, 1]^{S \times S}$. Denote its sample path as $\mathbf{X} = \{X_l : l \geq 0\}$. Let $\pi = (\pi(1), \ldots \pi(S))$ be the (row) vector representing its steady-state probabilities, and $f = (f(1), f(2), \cdots, f(S))^T$ be the (column) reward (cost) vector, where "T" represents transpose. We have $Pe = e$ and $\pi e = 1$ where $e = (1, 1, \cdots, 1)^T$ is an S-dimensional vector whose components are all equal to 1. The steady-state probability flow balance equation is $\pi = \pi P$. The performance measure is the long-run average defined as

$$\eta = \pi f = \sum_{i=1}^{\mathcal{S}} \pi(i) f(i) = \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} f(X_l), \quad w.p.1. \tag{1}$$

The last equation holds because of the ergodicity.

2.1. *Performance potentials*

The main concept in performance optimization is the *performance potential*, or simply the *potential*. Roughly speaking, the performance potential at state i, denoted as $g(i)$, $i \in \mathcal{S}$, measures the "potential" contribution of state i to the long-run average performance η. Intuitively, from (1), we can use something like $E\{\Sigma_{l=0}^{L-1} f(X_l) \,\big|\, X_0 = i\}$ to measure the average "potential" contribution of the current state i to the performance η. However, for ergodic chains, this sum goes to infinity as $L \to \infty$. As we will see, just like the potential energy in physics, the performance potentials are relative and we can subtract a constant from every components. Therefore, we define

$$g(i) = \lim_{L\to\infty} E\left\{\sum_{l=0}^{L-1} [f(X_l) - \eta] | X_0 = i\right\}. \tag{2}$$

It is well known that (2) is finite for ergodic chain. From (2), via a standard dynamic programming argument, it is easy to see that

$$\begin{aligned} g(i) &= \textit{contribution at the current state } i \\ &\quad + \textit{expected long term "potential" contribution of the next state} \\ &= (f(i) - \eta) + \sum_{j\in S} p(i,j) g(j). \end{aligned} \tag{3}$$

Writing (3) in a matrix, we obtain the *Poisson equation*:

$$(I - P)g + e\eta = f, \tag{4}$$

where $g = (g(1), \cdots, g(M))^T$ is the potential vector (for more details, see (Cao, 1998; Cao and Chen, 1997; Cao and Wan, 1998; Cao et al., 1996)). The solution to (4) is only up to an additive constant; i.e., if g is a solution to (4), then so is $g + ce$. (2) is the solution to (4) that satisfies $\pi g = 0$. For simplicity, we will refer to any solution to (4) as a potential (vector).

From (2), we can choose a large integer L and obtain the approximation

$$g(i) \approx E\left\{\sum_{l=0}^{L-1} [f(X_l) - \eta] \,\big|\, X_0 = i\right\}.$$

Moreover, since potentials are relative, we can remove the constant term $L\eta$ in the above expression and simply use the approximation

$$g(i) \approx E\left\{\sum_{l=0}^{L-1} [f(X_l)] | X_0 = i\right\}. \tag{5}$$

That is, we can average the sum of the reward function f in L transitions after visiting state i to obtain an approximate of $g(i)$. Thus, the potentials can be estimated on sample paths.

Since potentials are relative, sometimes it is easier to consider the differences of the potentials at two different states. Thus, we define

$$d(i, j) = g(j) - g(i), \quad i, j \in \mathcal{S}, \tag{6}$$

which is called the *perturbation realization factor* in PA (Cao and Chen, 1997; Cao et al., 1996). From the meaning of potentials, it is clear that $d(i, j)$ measures the effect of a perturbation from state i to state j on the long-run performance η. Let $D = [d(i, j)]$ be the *performance realization matrix*. We have $D^T = -D$ and $D = eg^T - ge^T$. By the same reasoning as (3), or from (3) directly, we can obtain

$$d(i, j) = f(j) - f(i) + \sum_{i' \in \mathcal{S}} \sum_{j' \in \mathcal{S}} p(i, i')p(j, j')d(i', j'). \tag{7}$$

In a matrix form, this is the Lyapunov equation

$$D - PDP^T = F,$$

with $F = ef^T - fe^T$. (This discrete-time version of Lyapunov equation was first derived in (Cao et al., 1996); for Lyapunov equation for continuous-time Markov processes, see (Cao and Chen, 1997). It is to be emphasized that despite the non-intuitive but elegant forms, the Poisson and the Lyapunov equations (3) and (7) flow from first principles of a Markov system.

With the realization factors, we can obtain many finite versions of the potentials. For example, following the same idea as for the potential energy in physics, we can choose any state $i^* \in \mathcal{S}$ and define $g(i^*) = 0$. Then we have

$$g(i) = d(i^*, i) + g(i^*) = d(i^*, i), \quad i \in \mathcal{S}.$$

To estimate $d(i,j)$, we consider two sample paths of the Markov chain, denoted as $\mathbf{X} = \{X_l, l \geq 0\}$ and $\mathbf{X}' = \{X'_l, l \geq 0\}$; they follow the same transition probability matrix P but start with two different initial states $X_0 = i$ and $X'_0 = j$. Define $L_{i,j} = \min\{k : k \geq 0, X_k = X'_k\}$. At $L_{i,j}$, the two sample paths "merge" together for the first time (see Figure 1). From $L_{i,j}$ on, the two sample paths behave statistically similar because both of them follow the same transition probability matrix. More precisely, By the strong Markov property, we have

$$\lim_{L \to \infty} E\left\{ \sum_{L_{i,j}}^{L-1} \left[f(X'_l) - f(X_l) \right] \middle| X_{L_{i,j}} = X'_{L_{i,j}} \right\} = 0.$$

Then from (2) we have

$$d(i,j) = \lim_{L \to \infty} E\left\{ \sum_{l=0}^{L-1} \left[f(X'_l) - f(X_l) \right] \middle| X_0 = i, X'_0 = j \right\} \tag{8}$$

$$= E\left\{ \sum_{l=0}^{L_{i,j}} \left[f(X'_l) - f(X_l) \right] \middle| X_0 = i, X'_0 = j \right\}. \tag{9}$$

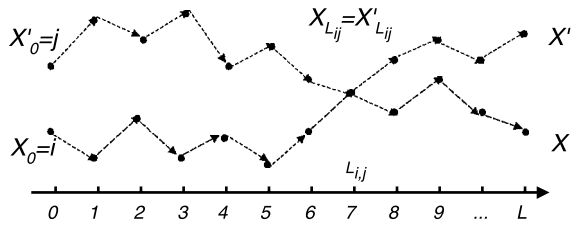

Figure 1. Estimating $d(i, j)$.

With (9), sample-path-based algorithms for estimating $d(i,j)$ can be developed. Compared with (5), the mean length of the sample paths in (9) is finite. Equation (8) helps us to understand the construction approach for the performance difference formula presented in Section 2.3. Other algorithms also exist (see, e.g., (Cao and Wan, 1998)).

2.2. *Performance sensitivity formulas*

Performance sensitivity formulas can be easily derived from the Poisson equation (4). Let P' be another irreducible transition probability matrix on the same state space, and π', f' and η' be the steady-state probability, the performance function, and the long-run average performance measure for the system associated with P'. Then $\eta' = \pi' f'$. Set $Q = P' - P = [q(i,j)]$ and $h = f' - f$. We have $Qe = 0$. Multiplying both sides of (4) on the left with π', we obtain the performance difference equation (Cao and Chen, 1997; Cao et al., 1996).

$$\eta' - \eta = \pi'(Qg + h). \tag{10}$$

Now, suppose that P changes to $P(\delta) = P + \delta Q = \delta P' + (1 - \delta)P$, and f changes to $f(\delta) = f + \delta h$, with $\delta \in (0,1)$. This corresponds to a randomized policy which applies policy P' with probability δ and policy P with probability $1 - \delta$. The performance measure will change to $\eta(\delta)$. The derivative of η in the direction of Q is denoted as $\frac{d\eta(\delta)}{d\delta}$. Taking $P(\delta)$ as the P' in (10), we have $\eta(\delta) - \eta = \pi(\delta)(\delta Qg + \delta h)$. Letting $\delta \to 0$, we get the performance derivative equation

$$\frac{d\eta}{d\delta} = \pi(Qg + h). \tag{11}$$

Since $Qe = 0$, both (11) and (10) hold for $g' = g + ce$ for any constant c. This verifies again that potentials are determined only up to an additive constant.

The extension of (11) to the case where P and f depend arbitrarily on any parameter θ (denoted as $P(\theta)$ and $f(\theta)$ with $P(0) = P, f(0) = f$) is straightforward. Replacing Q in (11) with $\left(\frac{dP}{d\theta}\right)\big|_{\theta=0}$ and h with $\left(\frac{df}{d\theta}\right)\big|_{\theta=0}$, we have

$$\frac{d\eta}{d\theta}\big|_{\theta=0} = \pi\left\{\left(\frac{dP}{d\theta}\right)g + \left(\frac{df}{d\theta}\right)\right\}_{\theta=0}. \tag{12}$$

Therefore, without loss of generality, we shall mainly discuss the case with a linear function $P(\delta) = P + \delta Q$ as in (11).

For simplicity, we assume $h = f' - f = 0$. Thus, (10) and (11) become

$$\eta' - \eta = \pi' Q g, \tag{13}$$

and

$$\frac{d\eta}{d\delta} = \pi Q g. \tag{14}$$

We will see that the above two simple formulas (13) and (14) are the most fundamental equations in learning and optimization, and many results can be explained clearly and derived concisely from these two equations.

2.3. Constructing the sensitivity formulas with potentials as building blocks

Although (13) and (14) can be easily derived analytically, the analytical derivation has two weaknesses: first, it does not provide a clear intuitive meaning of these two equations and the potentials; second, this approach may not apply to many systems that do not fit the standard MDP formulation, because the corresponding Poisson equation may not exist. In this section, we will show that various performance sensitivity formulas, including (13) and (14), can be constructed by first principle using performance potentials as building blocks. This construction approach provides the intuition behind the formulas, and with this method we may derive sensitivity formulas for systems that do not fit the standard formulation. As we will see later, new optimization approaches can be developed based on these sensitivity formulas thus constructed.

The construction approach was discussed in detail in (Cao, 2004a), and we will review the main concepts here. With the PA terminology, We start with a "perturbed" sample path with transition probability matrix P' (Path A–C in Figure 2). At every time instant on the perturbed path, we determine whether the state transition would be different if it followed transition probability matrix P, instead of P'; the transitions with P and P' are assumed to be independent. The figure illustrates that the transitions in segments A–D, K–E, M–G, and N–C happen to be the same for both P and P'. In other words, these segments can be viewed as a part of either an original path (with P) or a perturbed one (with P'). However, at Points D, E, and G, the transitions following P and P' are different. For example, at Point D, the perturbed system (with P') transits to state j', while the original system (with P) transits to state j. We say that a "jump" from j to j' occurs at $l = 2$. There is a jump from v to v' at $l = 5$ and a jump from j to j' again at $l = 8$ in Figure 2. Following the idea in (Cao, 2004), after each jump, we add an auxiliary path (D–R–B, E–F, and G–H) that follows the original transition probability matrix P. Based

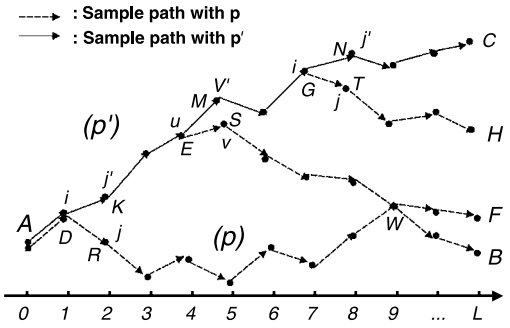

Figure 2. The performance difference of two policies.

on the construction, Paths A–R–B, K–E–F, M–G–H, and N–C can be viewed as original sample paths (following P). Let

$$F_L = \sum_{l=0}^{L-1} f(X_l), \quad F'_L = \sum_{l=0}^{L-1} f(X'_l),$$

where X_l and X_l' are the states at time l on the original path A–R–B and the perturbed path A–C, respectively. We also denote, for example,

$$F_{K-F} = \sum_{l=0}^{L-1} f\left(X_l^{K-F}\right),$$

with X_l^{K-F} being the state on Path K–F at time l. Similar notations are used for other paths. We have

$$\begin{aligned} \Delta F_L &:= F'_L - F_L = F_{A-C} - F_{A-R-B} \\ &= \{F_{A-C} - F_{A-H}\} + \{F_{A-H} - F_{A-E-F}\} + \{F_{A-E-F} - F_{A-R-B}\} \\ &= \{F_{N-C} - F_{T-H}\} + \{F_{M-H} - F_{S-F}\} + \{F_{K-F} - F_{R-B}\}. \end{aligned}$$

All the paths N–C, T–H, M–H, S–F, K–F, and R–B on the most-right side of the above equation follow the original transition probability matrix P. From (8), as $L \to \infty$, the average of $F_{K-F} - F_{R-B}$ goes to $d(j, j')$, the average of $F_{M-H} - F_{S-F}$ goes to $d(v, v')$, etc. In other words, each jump from j to j' contributes on the average $d(j, j')$ to the difference ΔF_L.

Next, after the system visits state i, $i \in \mathcal{S}$, the probability of such a jump from j to j' is $p(i, j)p'(i, j')$. Therefore, the number of such jumps on the L transitions is roughly

$$L\pi'(i)p(i, j)p(i, j').$$

Thus, on the average for a very large L the effect of all these jumps is

$$E(\Delta F_L) = E(F'_L) - E(F_L) \approx \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \sum_{j' \in \mathcal{S}} \{L\pi'(i)p(i, j)p'(i, j')d(j, j')\}. \tag{15}$$

Note that $\eta = \lim_{L\to\infty} E(F_L)/L$ and $d(j, j') = g(j') - g(j)$. When $L\to\infty$, the above approximation becomes accurate. We have

$$\begin{aligned}
\Delta\eta &= \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{j'\in\mathcal{S}}\left\{\pi'(i)p(i,j)p'(i,j')[g(j') - g(j)]\right\} \\
&= \sum_{i\in\mathcal{S}}\pi'(i)\left\{\sum_{j\in\mathcal{S}}\sum_{j'\in\mathcal{S}}p(i,j)p'(i,j')g(j') - \sum_{j\in\mathcal{S}}\sum_{j'\in\mathcal{S}}p(i,j)p'(i,j')g(j)\right\} \\
&= \sum_{i\in\mathcal{S}}\pi'(i)\left\{\sum_{j'\in\mathcal{S}}p'(i,j')g(j') - \sum_{j\in\mathcal{S}}p(i,j)g(j)\right\} \\
&= \sum_{i\in\mathcal{S}}\pi'(i)\left\{\sum_{j\in\mathcal{S}}[p'(i,j) - p(i,j)]g(i)\right\} = \sum_{i\in\mathcal{S}}\pi'(i)\left\{\sum_{j\in\mathcal{S}}[q(i,j)]g(j)\right\}.
\end{aligned}$$

This leads to (13).

Figure 2 shows that a sample path of any policy P' (A–C in Figure 2) can be decomposed into the sum of a sample path of any other policy P (A–R–B) and the segments representing the performance potentials of P (K–E–F, M–G–H, etc). This property holds for other systems with non-standard formulations. In (Cao, 2004a), we also obtained the sensitivity formulas for two Markov chains with different state spaces using this construction method.

3. A sensitivity point of view for performance optimization

In this section, we provide a sensitivity point of view to the area of learning and optimization with the standard MDP formulation; we show that the two sensitivity formulas (10) and (11) (or (13) and (14)) form the basis for performance optimization. The event-based optimization approach introduced later in this paper follows with this sensitivity view and is in parallel to those results for the standard MDP formulation.

3.1. Markov decision processes and policy iteration

Equations (13) and (14) can be applied to any two Markov chains defined in the same state space with no additional requirements. A popular formulation of optimization problem is the Markov decision process (MDP). The main approach to MDPs, policy iteration, follows directly from (13).

In an MDP (Bertsekas, 1995; Puterman, 1994), there is an action space $\mathcal{A}$ consisting of all (finite) available actions. If the system is at state i, $i \in \mathcal{S}$, an action $\alpha \in \mathcal{A}_i$ can be taken and applied to the system, where $\mathcal{A}_i \subseteq \mathcal{A}$ is the set of actions that are available at state $i \in \mathcal{S}$. The action determines the state transition probabilities. When action α is taken at state i, the state transition probabilities are denoted as $p^{\alpha}(i, j), j \in \mathcal{S}$. The reward that the system receives when it is at state i with action α is $f(i, \alpha)$. A deterministic and stationary policy is a mapping from $\mathcal{S}$ to $\mathcal{A}$, denoted as $\mathcal{L} : \alpha = \mathcal{L}(i)$, that determines the action taken at state i. Therefore, if policy $\mathcal{L}$ is adopted, the transition probability matrix

is $P^{\mathcal{L}} = \left[p^{\mathcal{L}(i)}(i,j)\right]_{i,j=1}^{\mathcal{S}}$. We assume that the system is ergodic under all policies. The long-run average system performance under policy $\mathcal{L}$ is defined as

$$\eta^{\mathcal{L}} = \lim_{L\to\infty}\left\{\frac{1}{L}\sum_{l=0}^{L-1} f(X_l, \mathcal{L}(X_l))\right\}, \quad w.p.1. \tag{16}$$

The goal of the MDP is to find a policy $\mathcal{L}^*$ such that its performance is the maximum among all policies. Since a policy corresponds to a transition matrix, we sometimes refer to a transition matrix as a policy.

In the standard MDP formulation, actions can be chosen independently at each state. Thus, there are $\prod_{i\in\mathcal{S}}|\mathcal{A}_i|$ ($|\mathcal{A}_i|$: number of policies in $\mathcal{A}_i$) policies in the policy space. This structural condition on the policy space is crucial for the policy iteration approach: it enables us to find a "better" policy in the policy space, if such a policy exists, by only analyzing the current policy. This updating procedure is based on the performance difference equation (13). Indeed, because $\pi' > 0$ component-wisely for any P', if we can find a P' such that $Qg = (P' - P)g \geq 0$, component-wisely, with at least one positive component, then we have $\eta' > \eta$; i.e., P' is a better policy. It is always possible to find such a P' (if it exists) in the policy space of an MDP because we can choose actions independently at any state (see (Cao, 1998) for details, the results were extended to the multi-chain case in (Cao and Guo, in press)). This "independent-action" assumption is crucial for policy iteration. In this procedure, we do not need to solve for π' for any transition probability matrix P'.

3.2. *Learning and optimization*

In a performance optimization problem, we have a Markov system which may run under many transition probability matrices, called policies. Therefore, we have a policy space consisting of discrete points as policies (illustrated as starts in an oval in Figure 3). The performance (16) depends on the policy. Our goal is to find a policy which attains the best performance (say the largest reward) in the policy space.

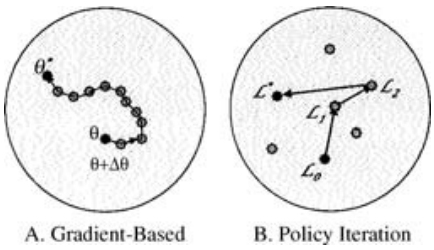

Figure 3. Two types of learning and optimization approaches.

In general, we may not require the actions at different states be chosen independently, which is the case in the MDP formulation. In this general form, actions at different states may be co-related; this is the case for many practical problems, in which the transition probability matrices in the policy space can be parameterized with a number of parameters.

There are two basic approaches for performance optimization of Markov systems:

1. In the general setting and for parameterized systems, performance gradients can be obtained by (11) (or equivalently (14), δ is a parameter representing randomized policies). Gradient-type optimization algorithms can be developed using the performance derivative formula (11).

2. With the independent-action assumption in the MDP formulation, policy iteration algorithms can be developed using the performance difference equation (13), as explained in Section 3.1.

The performance potential g plays a crucial role in both performance derivative-and difference formulas, which, as shown above, are the basis for the gradient-based and policy iteration-based optimization approaches. If we know the exact form and values of the transition probability matrices, it is possible to solve the Poisson equation (4) to obtain g and then apply the above optimization schemes. However, in many practical problems, the size of the Poisson equation may be too large for it to be solved analytically, or we may only have a partial information of the transition probability matrices; for example, in some cases we may know only the structure of the system but do not know the values of the system parameters, or in some other cases, we know the values of parameters but the system structure is too complicated and it is difficult to construct the transition probability matrices. In all such cases, we need to "learn" from the system behavior to determine which policy performs better. Thus, learning and optimization are closely related.

By "learning," we usually mean to observe a system's behavior and to analyze the information obtained through observation. Typically, while learning, a system is run under one (may be a randomized) policy in the policy space, and all information is contained in a sample path of the Markov system. Therefore, learning is sometimes called a sample-path-based approach. For both the gradient-based and polity iteration-based approaches, learning is essentially estimating the potentials g. In many gradient-based algorithms, the derivatives $\frac{d\eta}{d\delta}$ in (14) are estimated directly without estimating every component of g (Baxter and Bartlett, 2001; Baxter et al., 2001; Cao, 2004b; Cao and Wan, 1998). (A general principle for sample-path-based gradient algorithms for Markov systems is summarized in (Cao, 2004b), algorithms for performance gradients and optimization for special problems are proposed in (Baxter and Bartlett, 2001; Baxter et al., 2001; Cao and Wan, 1998; Chong and Ramadge, 1994; Ho and Cao, 1983; Marbach and Tsitsiklis, 2001; Suri and Leung, 1989); for sample-path-based policy iteration algorithms, see (Cao, 1999; Cooper et al., 2003; Fang and Cao, 2004). With g or $\frac{d\eta}{d\delta}$ estimated, an improvement decision can be made; note that such a decision is made by purely observing and analyzing a system's behavior under the current policy without intervening its operation. Thus, it is also called on-line optimization.

It should be noted that there is a slight difference between the on-line approach and the simulation-based approaches. In simulation, one may simulate a sample path that contains all the possible state-action pairs. Such an approach cannot be implemented on real systems because perturbing the system to deviate from the current policy for learning is usually not allowed. The approaches we discuss in this paper are of on-line nature.

One simple but fundamental fact about learning and optimization is that, by learning and/or analyzing a system behavior under a policy, we can only obtain the "local" information around that policy in the policy space. Indeed, by learning from the behavior of a system under a policy, in general we can not expect to obtain the performance of the system under other policies. This simple philosophical point puts fundamental limits on learning and optimization. A similar fact is the "No Free Lunch Theorem" (Ho et al., 2003), which assets that if there is no additional structural information, any optimization scheme is no better than blind searching.

As shown in Section 2.3, the performance derivative formula (14) can be interpreted by perturbation analysis (PA). PA focused on queueing-type of systems in its early days (Cao, 1994; Ho and Cao, 1983, 1991) and was extended to Markov system in (Cao and Chen, 1997; Cao et al., 1996). In PA, the perturbation realization factor $d(i,j)$ in (9) measures the effect of a perturbation on a sample path (an artificial "jump") from state i to j on the long-run average performance η, and the effect of any small (infinitesimal) change in the transition probabilities, represented by $Q\delta$, can be decomposed into the sum of the effects of a series of such jumps on a sample path. Therefore, by applying the PA principles to a sample path of a Markov system under a policy, we can obtain both the performance for that policy and the performance derivative along any given direction in the policy space specified by Q. (This is equivalent to obtaining the performance of the policies in an infinitesimal neighborhood of the policy in the policy space). These are the "local" information learned on a sample path, which leads to the first type of optimization approach, the gradient-based optimization approach shown in Figure 3A. We start from a policy with parameter θ and determine the performance gradient with PA, then change θ along the direction of the gradient, and learn again until the optimal value θ^* is reached. This approach can be used together with stochastic approximation techniques when the gradient estimates contain random noise (Marbach and Tsitsiklis, 2001).

The second type of optimization approaches, i.e., policy iteration, applies to the policy spaces satisfying the independent-action assumption. As shown in Figure 3B, we start from any policy $\mathcal{L}_0$, analyze its behavior and find a better policy $\mathcal{L}_1$, then learn from $\mathcal{L}_1$ and find a better policy $\mathcal{L}_2$, and so on until the best policy $\mathcal{L}^*$ is reached. As discussed in Section 3.1, policy iteration is based on the performance difference formula (13). In fact, as pointed out above, by learning and/or analyzing a system's behavior under one policy, it is not possible to know the exact value of the difference of the performance of two policies. In this sense, (13) is no better than the simplest formula $\eta' - \eta = (\pi' - \pi)f$, since in (13) we need to obtain both π' and g, which are associated with the two policies P' and P, respectively. The only merit of (13) lies in its particular form: because $\pi' > 0$ component-wisely, thus if $(P' - P)g \geq 0$ then we know $\eta' \geq \eta$ without the need to know the exact value of π'. This allows us to find better policies P' if the actions at different

states can be chosen independently. Therefore, the second (policy-iteration-based) approach relies on the particular form of the performance difference formula and applies to the policy spaces satisfying the independent-action assumption. On-line optimization algorithms are developed in (Fang and Cao, 2004).

In summary, by learning from the behavior of a system under one policy, we can only get the local information in an infinitesimal neighborhood of the policy in the policy space. The performance derivative obtained by PA can be used in gradient-based optimization. Policy iteration relies on the particular form of the performance difference formula (13) (see Cao and Guo, in press) for the multi-chain case) and works only when the actions at different states can be chosen independently. These two approaches are based on the two fundamental sensitivity formulas: performance derivative (14) and performance difference (13).

4. Event-based optimization: The main concepts

In many problems, the special feature related to performance changes can be characterized by "events." Such special features can be utilized in constructing the performance sensitivity formulas to carry out potential aggregation. Based on these formulas, similar approaches like the gradient-based optimization and policy iteration may be developed. In the remains of this paper, we will use an example in communication to illustrate the main ideas and results.

Consider the admission control problem in a communication system modelled as a variant of an open Jackson network (Dijk, 1993). The network consists of M servers; the service time of Server i is exponentially distributed with mean $1/\mu_i$, $i = 1, 2, \cdots, M$. After being served at Server i, a customer will join the queue at Server j with probability q_{ij}, and will leave the network with probability q_{i0}, $\sum_{j=0}^{M} q_{ij} = 1$, $i, j = 1, 2, \cdots, M$. Let n_i be the number of customers at Server i, and $n = \sum_{i=1}^{M} n_i$ be the number of all customers in (or the population of) the system. The customers arrive to 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 $b(n)$ and will be rejected with probability $1-b(n)$. The system has a capacity of N; i.e., $b(N) = 0$; thus, an arriving customer finding N customers in the system will be dropped. An admitted customer will join queue i with probability q_{0i}, $\sum_{i=1}^{M} q_{0i} = 1$. For simplicity, we assume $q_{ii} = 0$ for all i. As explained later, this assumption is not restrictive.

We model the system with the discrete-time Markov chain embedded at the transition epoches. The system state is $\mathbf{n} = (n_1, n_2, \cdots, n_M)$, and the state space is $\mathcal{S} := \{all\, \mathbf{n} : \sum_{i=1}^{M} n_i \leq N\}$. Set $\mathcal{S}_n := \{all\, \mathbf{n} : \sum_{i=1}^{M} n_i = n\}$, $n = 0, 1, \cdots, N$. We have $\mathcal{S} = \cup_{n=0}^{N} \mathcal{S}_n$. In the embedded chain at each epoch there is only one customer transition, because with the continuous time queueing model two transitions occur at the same instant with probability zero.

Note that in this problem we assume that the admission probability depends only on the population n not on the state $\mathbf{n}$. This corresponds to the partially observable Markov decision processes (POMDPs) in which only a partial information about the state is observable.

4.1. The events

[1]An important feature in this problem is that control is applied to the system only when a customer arrives at the network. A customer arrival is described by an *event* in a Markov model. This feature is very common in many problems: actions can only be applied to a system when certain events occur. Such a problem is not a standard MDP, because when an event occurs (a customer arrives), the system can be in many different states, and therefore an action may affect the transition probabilities of many states.

We first formally define an event with a Markov model. The system status is represented by the system state. An *event* is defined as a set of state transitions that satisfy some common properties. We denote a state transition from i to j, $i,j \in \mathcal{S}$, as $\langle i, j\rangle$ and denote the space of all transitions as $\mathcal{E} = \{\emptyset, \langle i,j\rangle : i,j \in \mathcal{S}\}$, with $\emptyset$ being a null element, defined only for logical reasons.

An event is a subset of $\mathcal{E} : e \subseteq \mathcal{E}$. All the set operations apply to events. For any $a, b, c \subset \mathcal{E}$, we can write $c = a \cap b$, $c = a \cup b$, and $c = \bar{a} = \mathcal{E} - a$. Also, we may have $a \subseteq b \subseteq c$, which indicates that if event a happens, then so does b, and so on. A state transition itself is an event, and is sometimes called a single event.

In the admission control problem, a state transition is denoted as $\langle \mathbf{n}, \mathbf{n}'\rangle$, $\mathbf{n}, \mathbf{n}' \in \mathcal{S}$. With the convention $q_{ii} = 0$ for all $i = 1, 2, \cdots, M$, a transition $\langle \mathbf{n}, \mathbf{n}\rangle$ clearly indicates an arriving customer is rejected by the system. For any state $\mathbf{n}$, denote the state after a customer joins Server i as $\mathbf{n}_{+\mathrm{i}} = (n_1, \cdots, n_{i-1}, n_i + 1, n_{\mathrm{i}+1}, \cdots, n_M\}$. Let $a_{n,+}$, $n < N$, be the event representing that an arrival customer is accepted AND there are a total of n customers in the network before the arrival. We have

$$a_{n,+} := \{\langle \mathbf{n}, \mathbf{n}_{+i}\rangle : \mathbf{n} \in \mathcal{S}_n, i = 1, \cdots, M\}.$$

Let $a_{n,-}$, $n \leq N$, be the event representing that an arrival customer is rejected AND there are a total of n customers before the arrival,

$$a_{n,-} := \{\langle \mathbf{n}, \mathbf{n}\rangle : \mathbf{n} \in \mathcal{S}_n\}.$$

The event representing a customer arrival when there are a total of n customers before the arrival is

$$a_n := a_{n,+} \cup a_{n,-},\ n \leq N,$$

with $a_{N,+} = \emptyset$. The event of customer arrivals is

$$a := \cup_{n=0}^{N} a_n.$$

The event of no customer arriving (including internal transitions and customer departures) is

$$b := \mathcal{E} - a. \tag{17}$$

The event that a customer is accepted is

$$a_+ = \cup_{n=0}^{N-1} a_{n,+}.$$

The event of that a customer is rejected is

$$a_- = \cup_{n=0}^{N} a_{n,-}.$$

Furthermore, the event representing an arrival customer joining Server i when there are n customers before the arrival is

$$a_{n,+i} := \{\langle \mathbf{n}, \mathbf{n}_{+i} \rangle, \ \mathbf{n} \in \mathcal{S}_n\};$$

and the event representing an arrival customer joining Server i is

$$a_{+i} = \cup_{n=0}^{N-1} a_{n,+i} = \{\langle \mathbf{n}, \mathbf{n}_{+i} \rangle, \ \mathbf{n} \in \mathcal{S} \text{ and } n < N\}.$$

From the above definitions, if a state transition $e \in a_n \cap a_+ \cap a_{+i}$ (equivalently $e \in a_n \cap a_{n+} \cap a_{n,+i}$), then $e = \langle \mathbf{n}, \mathbf{n}_{+i} \rangle$ with $\sum_{k=1}^{M} n_k = n$, i.e., a customer arrives when the system population is n, is accepted, and joins Server i.

Example: Figure 4 illustrates all the events for a system with $M = 3$ for $n = 1$. There are three states corresponding to $n = 1$: (0, 0, 1), (0, 1, 0), and (1, 0, 0). The top graph in the figure illustrates the events that may happen when the system is in state (0, 0, 1). The three (black) dashed arrows represent event b which contains internal transitions to states (0, 1, 0) and (1, 0, 0) and the customer-departure transition to state (0, 0, 0). The (red) block line represents the arrival event a_1, which contains both $a_{1,+}$ (accept), denoted as the (blue) dot-dashed line, and $a_{1,-}$ (rejection), denoted as the (pink) doted line. Event $a_{1,+}$ contains three events: $a_{1,+1}$, $a_{1,+2}$, and $a_{1,+3}$, denoted as the three (green) thin lines in the figure. Event $a_{1,+i}$ represents that the accepted customer joins Server i, $i = 1,2,3$. The middle and bottom graphes correspond to the events that may happen at states (0, 1, 0) and (1, 0, 0), respectively, in a similar way. The situation for other values of n is similar. Overall, we have $a_+ = \cup_{n=0}^{N-1} a_{n,+}$ and $a_- = \cup_{n=0}^{N} a_{n,-}$, etc. ■

When an event (other than a single event, i.e., a state transition) happens, we may not know the exact state, but we know that the state belongs to a particular subset of the state space. In addition to this partial information about the current state, we also have some knowledge about the state transition at this moment. In the Example, if we know that event a_1 happens, we do not know whether the state is (0, 0), (0, 1, 0), or (0, 0, 1). However, in addition to the partial information about the state (the population is *1*), we do know some partial information about the transition: after the event, the number of customers in the system either increases (accept), or remains the same (reject); it cannot decrease. That is, the next state cannot be (0, 0, 0). If we know that $a_{1,+}$ happens, then the next state cannot be (0, 0, 0) and (0, 0), (0, 1, 0), or (0, 0, 1). Therefore, an observation of an event may contain more information than a partial observation of the system state.

We have assumed $q_{ii} = 0$ for convenience. This is not restrictive. If $q_{ii} \neq 0$, then a transition $\langle \mathbf{n}, \mathbf{n} \rangle$ corresponds to two situations: a new customer arrives and gets rejected, or a customer returns back to the same server (with probability $q_{ii} \neq 0$). In a real system,

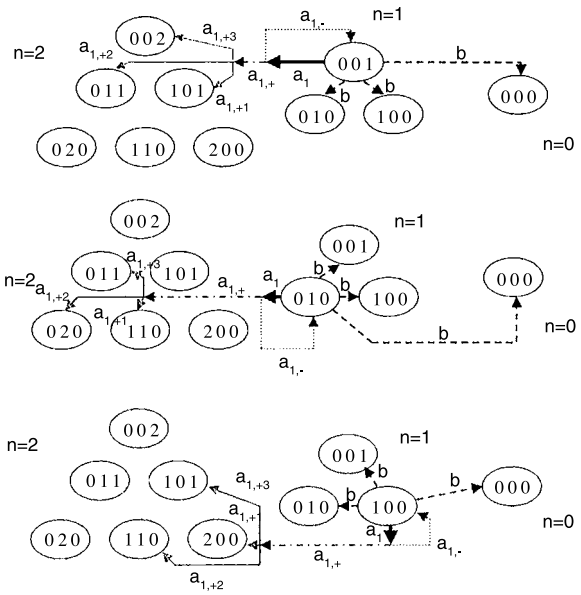

Figure 4. The events in the example with $M = 3$, $n = 1$.

one can observe the difference between these two situations; however, the Markov chain $\mathbf{X}$ does not reflect this difference since the state transition is the same for both situations. From the learning point of view, we need to introduce an additional index to distinguish the two situations. This will make the notations more complicated but does not change the concepts and the results.

Next, let us find out the probability of each event. We first start with the general Markov model. For convention, we say that a single event (a state transition) $e = \langle i, j \rangle$ happens at time l if $X_{l-1} = i$ and $X_l = j$; we denote it as $e_l = \langle i, j \rangle$. Let π_l be the state probability vector at any time instant $l = 0, 1, \cdots$, then $\pi_l = \pi_0 P^l$. The probability distribution π_{l-1} and the transition probability matrix P induce a probability measure on $\mathcal{E}$, denoted as

$$\mathcal{P}(e_l = \langle i,j \rangle) = \pi_{l-1}(i)p(i, j). \tag{18}$$

The steady-state probability of event $\langle i, j \rangle$ is

$$\pi(\langle i,j \rangle) := \pi(i)p(i,j), \tag{19}$$

While the event probability (18) or (19) depends on π_{l-1} or π, the conditional probabilities may only depend on the control parameters. For instance, in the admission control problem, we have

$$\mathcal{P}\big(a_{n,+} | a_n\big) = b(n),$$

and

$$\mathcal{P}(a_{n,-}|a_n) = 1 - b(n).$$

4.2. *Classification of events*

Let us study the logical relation among the events in the admission control problem. In the system, we first observe whether a customer arrives at an instant. If not, we do nothing. Suppose a_1 is observed at an instant, we know that a customer arrives and the system population is *1*. We can either accept the arriving customer, or reject it. That is, we can control the probabilities $b(1)$ and $1 - b(1)$ of the state transition belonging to a_+ or a_- ($a_{1,+}$ or $a_{1,-}$). We call a_n, $n = 0, 1, \cdots, N$, the *observable events* and a_+ and a_- the *controllable events*. Finally, after a customer is accepted, the "nature" determines which server it joins. That is, nature randomly chooses which sub-event a_{+i} or $a_{1,+i}$, $i = 1,2,3$, the transition at this instant belongs to. These events are called *natural transition events*. In summary, the state transition at a customer arrival belongs to three types of events, observable, controllable, and natural transition events, the probabilities of the controllable events can be controlled by actions. The three types of events and the action have a logical order in timing, as shown in Figure 5.

A state transition representing an arriving customer being accepted and joining Server i when the system state is $\mathbf{n}$ can be expressed as

$$\langle \mathbf{n}, \mathbf{n}_{+i} \rangle \in a_n \cap a_+ \cap_{a+i},$$

We can also write

$$\langle \mathbf{n}, \mathbf{n}_{+i} \rangle \in a_n \cap a_{n,+} \cap a_{n,+i}, \tag{20}$$

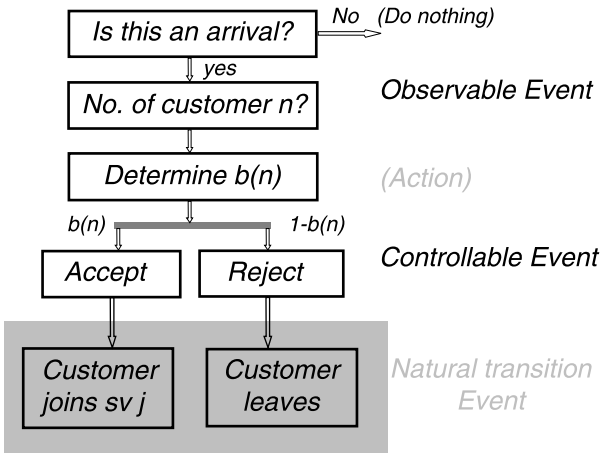

Figure 5. The logic relation among three types of events.

in which $a_{n,+} = a_n \cap a_+$ is a subset of a_n, and $a_{n,+i} = a_{n,+} \cap a_{+i}$ is a subset of $a_{n,+}$. Although (20) looks mathematically redundant, it helps understanding the logic in terms of observation, control, and natural transition. For example, in Figure 4, we have

$$\langle (0,0,1),(0,0,2)\rangle = a_1 \cap a_{1,+} \cap a_{1,+3},$$

An arriving customer being rejected when the system is in state $\mathbf{n}$ can be expressed as

$$\langle \mathbf{n}, \mathbf{n}\rangle \in a_n \cap a_- = a_n \cap a_{n,-},\ a_{n,-} = a_n \cap a_-,$$

$\sum_{i=1}^{M} n_i = n$. No natural transition event appears in this expression; or the nature has only one choice in this case. In Figure 4, we have

$$\langle (0,0,1),(0,0,1)\rangle = a_1 \cap a_{1,-}.$$

Now, let us decompose the event space $\mathcal{E}$. Note that the event b defined in (17), consisting of all internal customer transitions and departures from the network, is also an observable event. Thus, $\mathcal{E}$ can be decomposed into a set of mutually exclusive observable events:

$$\mathcal{E} = \left\{\cup_{n=0}^{N} a_n\right\} \cup b,$$

with $a_i \cap a_j = \emptyset$, $i \neq j$, $a_i \cap b = \emptyset$. Figure 4 shows that all the transitions from states with $n = 1$ belong to either a_1 or b.

Next, when b is observed, there is only one choice for control: do nothing. Thus, b can also be viewed as a special controllable event representing only one action corresponding to "do nothing". We have the mutually exclusive decomposition of $\mathcal{E}$ with the controllable events:

$$\mathcal{E} = a_+ \cup a_- \cup b,$$

with $a_+ \cap a_- = \emptyset$, $a_+ \cap b = a_- \cap b = \emptyset$. Lastly, when a_- happens, the nature also has only one choice. Thus, a_- is a special natural transition event. When b happens, the nature has a few choices; however, we will not elaborate these natural transition events under b since they are not related to the main topic of performance optimization. We have

$$\mathcal{E} = \left\{\cup_{i=1}^{M} a_{+i} \cup a_-\right\} \cup b,$$

with $a_{+i} \cap a_{+j} = \emptyset$, $i \neq j$, $a_{+i} \cap a_- = \emptyset$, and $a_{+i} \cap b = a_- \cap b = \emptyset$. Thus, every transition belongs to one of the exclusive observable (or, controllable, or natural transition) events.

In summary, the event-based approach applies to systems in which the event space can be decomposed into mutually exclusive subsets of observable events, mutually exclusive subsets of controllable events, and mutually exclusive subsets of natural transition events. Every state transition belongs to one event in each type. In addition, there is a logical order in timing among the three types of events: at any time instant, an observable event happens first, followed by a controllable event whose probability is determined by the action, then followed by a natural transition event. As a special case, which happens often, for some observable events only one action (usually "do nothing") is available;

(e.g., when a_N or b is observed, we have only one action: a_- for a_N, and "do nothing" for b) and after some control actions, there is a unique natural transition event (e.g., after a_- happens, the nature does nothing).

4.3. *Performance sensitivity formulas*

Now we derive the performance sensitivity formulas using the construction method illustrated in Section 2.3. To this end, we consider two admission policies $b(n)$ and $b'(n)$ for $n = 0, 1, \cdots, N$. From the event structure depicted above, the change in policy affects the performance only when events a_n, $n = 0, 1, \cdots, N$, occur. Let $\pi(n)$ denote the steady-state probability of event a_n, i.e., the probability that a customer arrives and finds n customers in the system. From the construction approach, the performance difference will depend on the steady-state probabilities of events a_n for policy $b'(n)$, denoted as $\pi'(n)$, $n = 0, 1, \cdots, N$ (instead of the steady-state probability $\pi'(\mathbf{n})$ as shown in (13)). Let $\pi(\mathbf{n} \mid n)$ be the conditional steady-state probability that the system is in state $\mathbf{n}$ when event a_n happens. Then

$$\pi(\mathbf{n}, n) = \pi(n)\pi(\mathbf{n}|n)$$

is the probability of a_n with state $\mathbf{n}$, $n_1 + \cdots + n_M = n$.

Following the construction method discussed in Section 2.3, we start with a perturbed sample path with admission policy $b'(n)$, $n = 0, 1, \cdots, N$. (cf. Path A–C in Figure 2). At every time instant on the perturbed path at which events a_n occur, $n = 0, 1, \cdots, N$, we determine whether the transition would be different if policy $b(n)$ (instead of $b'(n)$) were followed. If so, we add an auxiliary path following policy $b(n)$ (cf. Paths D–B, E–F, and G–H).

On a perturbed sample path for policy $b'(n)$ with $L >> 1$ transitions, there are $L\pi'(n)$ transitions at which event a_n occurs. Among them, $L\pi'(n)\,\pi'(\mathbf{n} \mid n)$ transitions are from state $\mathbf{n}$, with $n_1 + \cdots + n_M = n$. At these points, the probabilities that the system transits from state $\mathbf{n}$ to $\mathbf{n}_{+i} := (n_1, \cdots, n_i + 1, \cdots, n_M)$ are $b(n)\,q_{0i}$ and $b'(n)\,q_{0i}$, $i = 1, \cdots, M$, respectively, for the two policies; and the probabilities that the system transit from state $\mathbf{n}$ to $\mathbf{n}$ itself are $1 - b(n)$, and $1 - b'(n)$, respectively, for the two policies; where $b(n) = \mathcal{P}(a_{n,+}|a_n)$ and $b'(n) = \mathcal{P}'(a_{n,+}|a_n)$ are determined by the control actions, and q_{0i}, $i = 1, \cdots, M$, are the natural transition probabilities.

First, let i be a fixed integer. On the perturbed sample path (under policy $b'(n)$), after an arrival customer finds the system state being $\mathbf{n}$, the system transits to state $\mathbf{n}_{+i}$ with probability $b'(n)\,q_{0i}$. However, it would transit to any state $\mathbf{n}_{+j}$, $j = 1, \cdots, M$, with probability $b(n)\,q_{0j}$, and to state $\mathbf{n}$ with probability $1 - b(n)$, if policy $b(n)$ were used. Therefore, after an arrival customer finds the system state being $\mathbf{n}$,

$$\textit{the probability of a jump from } \mathbf{n}_{+j} \textit{ to } \mathbf{n}_{+i} \textit{ is } b'(n)q_{0i}b(n)q_{0j}, \quad j = 1, \cdots, M, \tag{21}$$

and

$$\textit{the probability of a jump from } \mathbf{n} \textit{ to } \mathbf{n}_{+i} \textit{ is } b'(n)q_{0i}[1 - b(n)]. \tag{22}$$

Similarly, On the perturbed sample path (under policy $b'(n)$), after an arrival customer finds the system state being $\mathbf{n}$, the system transits to state $\mathbf{n}$ with probability $1 - b'(n)$. However, it would transit to any state $\mathbf{n}_{+j}$, $j = 1,\cdots, M$, with probability $b(n)q_{0j}$, and to state $\mathbf{n}$ with probability $1 - b(n)$, if policy $b(n)$ were used. Therefore, we have

$$\textit{the probability of a jump from } \mathbf{n}_{+j} \textit{ to } \mathbf{n} \textit{ is } \big[1 - b'(n)\big]b(n)q_{0j}, j = 1,\cdots,M, \quad (23)$$

and

$$\textit{the probability of a jump from } \mathbf{n} \textit{ to } \mathbf{n} \textit{ is } \big[1 - b'(n)\big]\big[1 - b(n)\big]. \quad (24)$$

Note that a jump from state $\mathbf{n}$ to the same state is a fictitious jump; its effect on performance is $d(\mathbf{n}, \mathbf{n}) = 0$. (24) and (21) with $i = j$ are fictitious jumps.

Each jump from state $\mathbf{n}$ to $\mathbf{n}'$ contributes to the performance difference ΔF_L an amount of $d(\mathbf{n}, \mathbf{n}') = g(\mathbf{n}') - g(\mathbf{n})$. Finally, we add up such effects due to all the jumps in (21) to (24) together and obtain (cf. (15)):

$$\begin{aligned} & E(\Delta F_L) = E(F'_L) - E(F_L) \\ \approx\ & \sum_{n=0}^{N} \sum_{all\ \mathbf{n}\in\mathcal{S}_n} L\pi'(n)\pi'(\mathbf{n}|n) \\ & \left\{ \sum_{i=1}^{M}\sum_{j=1}^{M} b'(n)q_{0i}b(n)q_{0j}d(\mathbf{n}_{+j}, \mathbf{n}_{+i}) + \sum_{i=1}^{M} b'(n)q_{0i}\big[1 - b(n)\big]d(\mathbf{n}, \mathbf{n}_{+i}) \right. \\ & \left. + \sum_{j=1}^{M} \big[1 - b'(n)\big]b(n)q_{0j}d(\mathbf{n}_{+j}, \mathbf{n}) + \big[1 - b'(n)\big]\big[1 - b(n)\big]d(\mathbf{n}, \mathbf{n}) \right\}. \end{aligned} \quad (25)$$

The first and last terms in (25) are zero. Indeed, for the first term we have

$$\begin{aligned} & \sum_{i=1}^{M}\sum_{j=1}^{M} b'(n)q_{0i}b(n)q_{0j}d(\mathbf{n}_{+j}, \mathbf{n}_{+i}) = b'(n)b(n)\sum_{i=1}^{M}\sum_{j=1}^{M} q_{0i}q_{0j}\big[g(\mathbf{n}_{+i}) - g(\mathbf{n}_{+j})\big] \\ & \quad = b'(n)b(n)\left\{ \Big[\sum_{i=1}^{M} q_{0i}g(\mathbf{n}_{+i})\Big] - \Big[\sum_{j=1}^{M} q_{0j}g(\mathbf{n}_{+j})\Big] \right\} = 0. \end{aligned}$$

Therefore, (25) becomes (noting $b(N)=b'(N) = 0$)

$$\begin{aligned} & E(\Delta F_L) \approx \sum_{n=0}^{N-1} \sum_{all\ \mathbf{n}\in\mathcal{S}_n} L\pi'(n)\pi'(\mathbf{n}|n) \\ & \left\{ \sum_{i=1}^{M} b'(n)q_{0i}[1 - b(n)]d(\mathbf{n}, \mathbf{n}_{+i}) + \sum_{j=1}^{M} [1 - b'(n)]b(n)q_{0j}d(\mathbf{n}_{+j}, \mathbf{n}) \right\} \\ & \quad = \sum_{n=0}^{N-1} \sum_{all\ \mathbf{n}\in\mathcal{S}_n} L\pi'(n)\pi'(\mathbf{n}|n)[b'(n) - b(n)]\left\{ \sum_{i=1}^{M} q_{0i}[g(\mathbf{n}_{+i}) - g(\mathbf{n})] \right\}. \end{aligned}$$

Dividing both sides with L and letting $L \to \infty$, we get

$$\eta' - \eta = \sum_{n=0}^{N-1} \big\{\pi'(n)[b'(n) - b(n)]d(n)\big\}, \quad (26)$$

where $d(n)$ is defined as

$$d(n) = \sum_{n \in \mathcal{S}_n} \pi'(\mathbf{n}|n) \left\{ \sum_{i=1}^{M} q_{0i}[g(\mathbf{n}_{+i}) - g(\mathbf{n})] \right\}, \quad n = 0, 1, \cdots, N-1. \tag{27}$$

The construction approach is intuitively clear and directly leads to the desired format. Moreover, both the construction and the form of the difference formulas (26) and (27) depend only on the system parameters q_{0i}, $i = 1, \cdots, M$, and do not depend on the explicit form of the transition probability matrix. Thus, this approach maintains the structure property of the system and avoids the tedious effort in finding and storing the large matrix.

The quantity $d(n)$ in (27) is an aggregation of the performance potentials. The weighting factor is the conditional probability of the perturbed system $\pi'(\mathbf{n}|n)$. However, from the product-form solution of queueing networks (Dijk, 1993), we can prove

$$\pi'(\mathbf{n}|n) = \pi(\mathbf{n}|n). \tag{28}$$

Thus,

$$\begin{aligned} d(n) &= \sum_{\mathbf{n} \in \mathcal{S}_n} \pi(\mathbf{n}|n) \left\{ \sum_{i=1}^{M} q_{0i}[g(\mathbf{n}_{+i}) - g(\mathbf{n})] \right\} \\ &= \tilde{g}_+(n) - \tilde{g}(n). \end{aligned} \tag{29}$$

in which

$$\tilde{g}_+(n) = \sum_{\mathbf{n} \in \mathcal{S}_n} \left\{ \pi(\mathbf{n}|n) \left[\sum_{i=1}^{M} q_{0i} g(\mathbf{n}_{+i}) \right] \right\}, \tag{30}$$

$$\tilde{g}(n) = \sum_{\mathbf{n} \in \mathcal{S}_n} \pi(\mathbf{n}|n) g(\mathbf{n}). \tag{31}$$

Both $\tilde{g}_+$ and $\tilde{g}$ have a clear physical meaning. $\tilde{g}(n)$ is the aggregated potential for the set of states with the same network population n; and $\tilde{g}_+(n)$ is the aggregated potential after a customer is accepted to a network with population n.

For performance derivatives, we assume that the policy changes from $b(n)$ to $b(n) + \delta_n$ for a fixed n. From (26), we have

$$\frac{\partial \eta}{\partial b(n)} = \pi(n) d(n), \quad n = 0, 1, \cdots, N, \tag{32}$$

where $d(n)$ is the aggregated potential in (29). This is the derivative with respect to the admission probability at one population. In general, suppose the policy $b(n)$ depends on a parameter θ and is denoted as $b_\theta(n)$. Then, from (26), we have

$$\frac{d\eta(\theta)}{d\theta} = \sum_{n=0}^{N-1} \pi(n) \frac{db_\theta(n)}{d\theta} d(n), \tag{33}$$

in which both $d(n)$ and $\pi(n)$, $n = 0, 1, \cdots, N$, depend only on the original policy $b(n)$.

4.4. *Sample-path-based estimation*

Both $\tilde{g}_+$ and $\tilde{g}$ in (29) can be estimated on a sample path of the original system. In this section, we first give an intuitive explanation about how an event-based aggregated potential can be estimated on a sample path with the same amount of computation and the same accuracy as estimating the potential of a single state. To this end, we work with the general Markov model. Let us first review how a potential at state i, $g(i)$, is estimated. The simplest way is shown in Figure 6A. After each visit at state i, we sum up the reward function for $N >> 1$ consecutive transitions to obtain $g_1, g_2, \ldots$ For example, in the figure $g_1 = \sum_{l=1}^{1+N} f(X_l)$, $g_2 = \sum_{l=7}^{7+N} f(X_l)$ etc. From (5), we have,

$$g(i) \approx \frac{1}{K} \sum_{k=1}^{K} g_k, \tag{34}$$

where K is the number of visits to state i in the sample path $\{X_0, \cdots, X_L\}$. Furthermore, if we make a slight change in the above expression, we can obtain an estimate of the weighted potential. In fact, for a large integer L, we have

$$\frac{1}{L} \sum_{k=1}^{K} g_k = \frac{K}{L} \frac{1}{K} \sum_{k=1}^{K} g_k \approx \pi(i) g(i),$$

where $\frac{K}{L} \approx \pi(i)$ is the steady-state probability of i.

With the same principle, we can estimate the aggregated potential. This is shown in Figure 6B. Instead of collecting the sums of the reward functions, g_k's, starting from a particular state i, we collect these g_k's starting from any particular event a. Let us consider

$$\frac{1}{K} \sum_{k=1}^{K} g_k, \tag{35}$$

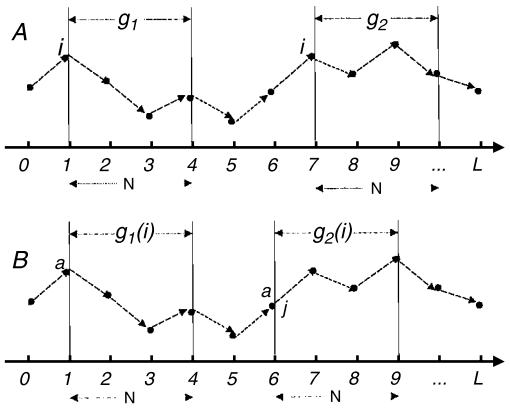

Figure 6. The sample-path based estimation of aggregated potentials.

where K is the number of transitions at which the event a occur. When a occurs, the system may be in different state $i \in \mathcal{S}$. Let K_i be the number of transitions at which the event a occurs and meanwhile the state is i. Then $\sum_{i\in\mathcal{S}} K_i = K$. To distinguish the states, we denote the sums g_k when the state is i as $g_k(i)$. Then (35) becomes

$$\frac{1}{K}\sum_{k=1}^{K} g_k = \sum_{i\in\mathcal{S}}\left\{\frac{1}{K}\sum_{k:\text{with state } i} g_k(i)\right\} = \sum_{i\in\mathcal{S}}\left\{\frac{K_i}{K}\left[\frac{1}{K_i}\sum_{k:\text{with state } i} g_k(i)\right]\right\}.$$

Clearly we have $\frac{1}{K_i}\sum_{k:\text{with state } i} g_k(i) \approx g(i)$, and $\frac{K_i}{K} \approx \pi(i|a)$ being the steady-state conditional probability of i given that event a occurs. Thus,

$$\frac{1}{K}\sum_{k=1}^{K} g_k \approx \sum_{i\in\mathcal{S}} \pi(i|a)g(i). \tag{36}$$

In the same spirit, we have

$$\frac{1}{L}\sum_{k=1}^{K} g_k \approx \pi(a)\sum_{i\in\mathcal{S}} \pi(i|a)g(i) = \sum_{i\in\mathcal{S}} \pi(i,a)g(i), \tag{37}$$

where $\pi(a)$ is the steady-state probability of event a, and $\pi(i, a)$ is the steady-state joint probability of event a and state i.

Many event-based aggregated potentials have either form (36) or (37). For example, the $\tilde{g}(n)$ in (31) is in the form of (36), with a being the event denoting that the system population is n. $\tilde{g}_+(n)$ in (30) takes a similar form, except that it also distinguishes which server the accepted customer joins.

As an example, we now develop the details for the sample-path-based estimation of $\tilde{g}_+(n)$ in the admission control problem. Consider a sample path $\{X_0, X_1,\cdots, X_L\}$, with $L >> 1$. Denote the sequence of the time instants at which event $a_{n,+}$ happens (i.e., an arriving customer finding n customers in the system is accepted) on the sample path as $l_1,\cdots, l_{Ln,+}$. Then at l_k, $k = 1, 2,\cdots, L_{n,+}$, there are $n + 1$ customers in the system. Choose a large integer N. Set

$$g_k = \sum_{l=k}^{k+N} [f(X_l)].$$

Next, we group the set $\mathcal{T}_{n,+} := \{l_k, k = 1, 2, \cdots, L_{n,+}\}$ into sub-groups $\mathcal{T}_{n,+} = \cup_{\mathbf{n}\in\mathcal{S}_n}\mathcal{T}_{\mathbf{n},+}$, such that before the customer arriving at $l \in \mathcal{T}_{n,+}$ is accepted the system state is $\mathbf{n}$ with population n. Let $L_{\mathbf{n},+}$ be the number of instants in $\mathcal{T}_{\mathbf{n},+}$. We have $L_{n,+} = \sum_{\mathbf{n}\in\mathcal{S}_n} L_{\mathbf{n},+}$. We further group the subset $\mathcal{T}_{\mathbf{n},+}$ into $\mathcal{T}_{\mathbf{n},+} = \cup_{i=1}^{M}\mathcal{T}_{\mathbf{n}_{+i}}$; in $\mathcal{T}_{\mathbf{n}_{+i}}$, the accepted customer joins Server i, $i = 1,\cdots, M$. Let $L_{\mathbf{n}_{+i}}$ be the number of instants in $\mathcal{T}_{\mathbf{n}_{+i}}$. We have

$$L_{\mathbf{n},+} = \sum_{i=1}^{M} L_{\mathbf{n}_{+i}},\ L_{n,+} = \sum_{\mathbf{n}\in\mathcal{S}_n}\sum_{i=1}^{M} L_{\mathbf{n}_{+i}}.$$

From the above definitions, we have

$$\begin{aligned}\frac{1}{L_{n,+}}\sum_{k=1}^{L_{n,+}} g_{l_k} &= \frac{1}{L_{n,+}}\sum_{\mathbf{n}\in\mathcal{S}_n}\sum_{l_k\in\mathcal{T}_{\mathbf{n},+}} g_{l_k} \\ &= \frac{1}{L_{n,+}}\sum_{\mathbf{n}\in\mathcal{S}_n}\sum_{i=1}^{M}\sum_{l_k\in\mathcal{T}_{\mathbf{n}+i}} g_{l_k} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_n}\left\{\frac{L_{\mathbf{n},+}}{L_{n,+}}\sum_{i=1}^{M}\frac{L_{\mathbf{n}+i}}{L_{\mathbf{n},+}}\left[\frac{1}{L_{\mathbf{n}+i}}\sum_{l_k\in\mathcal{T}_{\mathbf{n}+i}} g_{l_k}\right]\right\}.\end{aligned} \tag{38}$$

By definitions, we have

$$\lim_{N\to\infty}\lim_{L_{\mathbf{n}+i}\to\infty}\frac{1}{L_{\mathbf{n}+i}}\sum_{l_k\in\mathcal{T}_{\mathbf{n}+i}} g_{l_k} = g(\mathbf{n}+i),$$

and

$$\lim_{L_{\mathbf{n},+}\to\infty}\frac{L_{\mathbf{n}+i}}{L_{\mathbf{n},+}} = q_{0i},\ \lim_{L_{n,+}\to\infty}\frac{L_{\mathbf{n},+}}{L_{n,+}} = \pi(\mathbf{n}|n).$$

With these equations and taking $\lim_{L_{n,+}\to\infty}$ on both sides of (38), we obtain

$$\lim_{L_{n,+}\to\infty}\frac{1}{L_{n,+}}\sum_{k=1}^{L_{n,+}} g_{l_k} = \tilde{g}_+(n), \tag{39}$$

in (30). Sample-path-based algorithms can be developed with (39).

We have demonstrated that the event-based aggregated potentials can be estimated on a sample path, as shown in (36) and (37). It is important to note that (36) and (37) have the same form as the estimation for the potential of a single state (34); thus estimating an aggregated potential requires the same computation and has the same accuracy as estimating the potential of a state. Because the number of aggregated potentials is usually much less than that of the states, the event-based optimization saves considerably computations.

4.5. Performance optimization

Both the performance difference and derivative formulas (26) and (33) take a similar form as these for the standard MDPs (13) and (14). Therefore, gradient-based and policy iteration type of optimization approaches may be developed based on these two formulas.

Policy iteration can be derived from performance difference formula (26). Following the same reasoning as the standard MDPs, because $\pi'(n) > 0$ for any policy $b'(n)$, we have $\eta' > \eta$ if $[b'(n) - b(n)]\, d(n) \geq 0$ for all n with $[b'(n) - b(n)]\, d(n) > 0$ for at least one n, $0 \leq n < N$. From this, as for the standard MDPs, we can always find a better policy, if such better policies exist, by analyzing the current system, or by estimating $d(n)$ on a sample path. In addition, from the form of (26), we can conclude that the optimal policy $b^*(n)$

must be at the corner of the feasible policy space: if $d^*(n) > 0$ (or <0) then $b^*(n)$ is the largest (or the smallest) among all possible $b(n)'s$. From the physical meaning of $\tilde{g}$ and $\tilde{g}_+$, $d(n) = \tilde{g}_+(n) - \tilde{g}(n) > 0$ implies that the aggregated potential after accepting a customer is larger than the original aggregated potential at population n; in this case, accepting the arriving customer makes the performance better.

However, the policy iteration with aggregated potentials depends heavily on the condition (28). In general, it says that the conditional probability of a system state given that an observable event occurs is the same for different policies. This condition is satisfied for the admission control problem and some other queueing-type systems. It is, however, restrictive in general. This may explain why policy iteration algorithms do not generally exist for non-standard MDP problems and aggregation is not widely applicable for policy iteration. This also explains that the gradient-based approach may have more advantages since in the form of performance derivative (32) and (33), the aggregated potentials $d(n)$'s depend always only on the original policy. There are other conditions under which policy iteration algorithms can be developed with the event-based performance difference formulas. These are the topics in the next research paper (Cao, 2004c).

As explained, the admission control problem cannot be solved by the standard MDP method, because the same action $b(n)$ has to be chosen for many different states $\mathbf{n}$ with the same population n. The event-based approach with the performance difference formula provides a neat solution in a way similar to the standard policy iteration; the potentials are aggregated; the aggregated potentials can be obtained analytically or can be estimated on sample paths; the number of potentials to be estimated is reduced significantly.

The aggregated potentials $d(n)$ in the performance gradient formula (29) depends only on the policy $b(n)$. As shown in Section 4.4, $d(n)$ can be estimated on a sample-path-based under policy $b(n)$. This is in consistent with the observation that we can obtain local information including the gradient by observing and analyzing a sample path of the original policy. The advantage of the approach presented above is that with aggregation, the number of potentials to be estimated is reduced to N, which is much less than the number of states. With the N estimates for $d(n)$, $n = 0,1,\cdots, N-1$, we can obtain all the N partial derivatives in (32). Thus, the event-based approach scales to the system size N, while the number of potentials $g(\mathbf{n})$, $n \in \mathcal{S}$, is $\sum_{n=0}^{N} \frac{(N+M-1)!}{N!(M-1)!}$, which grows exponentially in N. In addition, in this approach, the potential aggregation is carried out by directly using the structure property of a system, and it does not require to know the transition probability matrix.

5. Discussions and conclusions

In this paper, we first presented a sensitivity point of view for learning and performance optimization of Markov systems. By observing and analyzing a system's behavior under a policy, one can in general obtain only the local information around that policy in the policy space, including the performance gradients. Gradient-based optimization, in which system parameters change by a small amount in each step, is based on the performance

derivative formula. Policy iteration, in which the policy changes between discrete points in the policy space, depends heavily on the particular form of the performance difference formulas. Thus, the two performance sensitivity formulas, the derivative and the difference, form the basis for learning and optimization. We also reviewed the construction method; with this method, we can derive performance sensitivity formulas for many different problems flexibly, by first principles, with performance potentials as building blocks.

With the sensitivity point of view of optimization and the flexible construction of sensitivity formulas, we proposed an event-based optimization approach, which is illustrated by an example (a rigorous mathematical framework is provided in (Cao, 2004c). This approach utilizes the special feature of a system and illustrates how the potentials can be aggregated based on the special feature. The aggregated potentials can be used to build performance sensitivity formulas which lead to gradient-based optimization and, with some conditions, event-based policy iteration. An aggregated potential can be estimated on a sample path with the same computation and same accuracy as estimating a potential of a state. This approach reduces the policy space and hence saves computation. As the admission control problem indicates, the approach may scale to the system size, while the number of system states grows exponentially. The approach applies to practical problems that do not fit well the standard MDP formulation.

The event-based framework can be applied to many problems. For example, in the two-level hierarchical control problem, we may denote the high-level (with a slow time scale) state as x and the lower-level (with a fast time scale) as y. The overall system state is (x, y). Any transition out from a high-level state x can be viewed as an observable event.

The rest can be formulated according to the specifics of the problem. In singular perturbed systems, the system stays in the same mode for a long period before it moves to other modes. We can define the observable events to represent the transitions among different modes. Finally, in partially observable MDPs (POMDPs), the state x is not observable, but a random variable y with distribution $f_x(y)$ can be observed. We may use y, or the "belief state," or the "internal state" (Meuleau et al., 1999; Theocharous and Kaelbling, 2004), to define observable events and then apply the event-based approach proposed in this paper.

Performance difference and derivative formulas can be developed for these problems with the event-based approach. With these formulas, potentials are aggregated in gradient-based optimization algorithms, and we can find the conditions under which the aggregated potentials can be used to implement policy iteration. All those topics require further research.

Acknowledgment

The author would like to express his sincere thanks to Prof. Y. C. Ho of Harvard University for his carefully reading of a few earlier drafts of this paper; his insightful comments and suggestions have helped greatly in improving the presentation of the paper. Of course, all remaining errors are solely the responsibility of the author.

Note

1. For readers whose primary interest is in Sections 4.3 and 4.4, the first two Sections 4.1 and 4.2 can be skimmed on a first reading since not all the notations which are introduced for completeness will be needed for understanding of Sections 4.3 and 4.4.

References

Barto, A., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning, special issue on reinforcement learning. *Discret. Event Dyn. Syst. Theory Appl*. 13: 41–77.

Baxter, J., and Bartlett, P. L. 2001. Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res*. 15: 319–350.

Baxter, J., Bartlett, P. L., and Weaver, L. 2001. Experiments with infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res*. 15: 351–381.

Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control, Volume I and II*. Belmont, MA: Athena Scientific.

Cao, X. R. 1994. *Realization Probabilities: The Dynamics of Queueing Systems*. New York: Springer-Verlag.

Cao, X. R. 1998. The relation among potentials, perturbation analysis, Markov decision processes, and other topics. *J. Discret. Event Dyn. Syst*. 8: 71–87.

Cao, X. R. 1999. Single sample path based optimization of Markov chains. *J. Optim. Theory Appl*. 100(3): 527–548.

Cao, X. R. 2000. A unified approach to Markov decision problems and performance sensitivity analysis. *Automatica* 36: 771–774.

Cao, X. R. 2004a. The potential structure of sample paths and performance sensitivities of Markov systems. *IEEE Trans. Automat. Contr*. 49: 2129–2142.

Cao, X. R. 2004b. A basic formula for on-line policy gradient algorithms. *IEEE Trans. Automat. Contr*. to appear.

Cao, X. R. 2004c. Event-based optimization of Markov systems. Manuscript to be submitted.

Cao, X. R., and Chen, H. F. 1997. Perturbation realization, potentials and sensitivity analysis of Markov processes. *IEEE Trans. Automat. Contr*. 42: 1382–1393.

Cao, X. R., and Guo, X. 2004. A unified approach to Markov decision problems and performance sensitivity analysis with discounted and average criteria: Multichain cases. *Automatica* 40: 1749–1759.

Cao, X. R., and Wan, Y. W. 1998. Algorithms for sensitivity analysis of Markov systems through potentials and perturbation realization. *IEEE Trans. Control Syst. Technol*. 6: 482–494.

Cao, X. R., Yuan, X. M., and Qiu, L. 1996. A single sample path-based performance sensitivity formula for Markov chains. *IEEE Trans. Automat. Contr*. 41: 1814–1817.

Cao, X. R., Ren, Z. Y., Bhatnagar, S., Fu, M., and Marcus, S. 2002. A time aggregation approach to Markov decision processes. *Automatica* 38: 929–943.

Chong, E. K. P., and Ramadge, P. J. 1994. Stochastic optimization of regenerative systems using infinitesimal perturbation analysis. *IEEE Trans. Automat. Contr*. 39: 1400–1410.

Cooper, W. L., Henderson, S. G., and Lewis, M. E. 2003. Convergence of simulation-based policy iteration. *Probab. Eng. Inf. Sci*. 17: 213–234.

Dijk, N. V. 1993. *Queueing Networks and Product Forms: A Systems Approach*. Chichester: John Willey and Sons.

Fang, H. T., and Cao, X. R. 2004. Potential-based on-line policy iteration algorithms for Markov decision processes. *IEEE Trans. Automat. Contr*. 49: 493–505.

Ho, Y. C., and Cao, X. R. 1983. Perturbation analysis and optimization of queueing networks. *J. Optim. Theory Appl*. 40(4): 559–582.

Ho, Y. C., and Cao, X. R. 1991. *Perturbation Analysis of Discrete-Event Dynamic Systems*. Boston: Kluwer Academic Publisher.

Ho, Y. C., Zhao, Q. C., and Pepyne, D. L. 2003. The no free lunch theorem, complexity and computer security. *IEEE Trans. Automat. Contr*. 48: 783–793.

Marbach, P., and Tsitsiklis, T. N. 2001. Simulation-based optimization of Markov reward processes. *IEEE Trans. Automat. Contr.* 46: 191–209.

Meuleau, N., Peshkin, L., Kim, K.- E., and Kaelbling, P. L. 1999. Learning finite-state controllers for partially observable environments. *Proceedings of the Fifteenth International Conference on Uncertainty in Artificial Intelligence*.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley.

Suri, R., and Leung, Y. T. 1989. Single run optimization of discrete event simulations—An empirical study using the M/M/1 queue. *IIE Trans.* 21: 35–49.

Theocharous, G., and Kaelbling, P. L. 2004. Approximate planning in POMDPS with macro-actions. *Advances in Neural Information Processing Systems 16 (NIPS-03)*. Cambridge, MA: MIT Press. 775–782.

Watkins, C., and Dayan, P. 1992. Q-learning. *Mach. Learn*. 8: 279–292.