Discrete Event Dynamic Systems: Theory and Applications, 13, 9–39, 2003

From Perturbation Analysis to Markov Decision Processes and Reinforcement Learning

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

Abstract. The goals of perturbation analysis (PA), Markov decision processes (MDPs), and reinforcement learning (RL) are common: to make decisions to improve the system performance based on the information obtained by analyzing the current system behavior. In this paper, we study the relations among these closely related fields. We show that MDP solutions can be derived naturally from performance sensitivity analysis provided by PA. Performance potential plays an important role in both PA and MDPs; it also offers a clear intuitive interpretation for many results. Reinforcement learning, TD(λ), neuro-dynamic programming, etc., are efficient ways of estimating the performance potentials and related quantities based on sample paths. The sensitivity point of view of PA, MDP, and RL brings in some new insight to the area of learning and optimization. In particular, gradient-based optimization can be applied to parameterized systems with large state spaces, and gradient-based policy iteration can be applied to some nonstandard MDPs such as systems with correlated actions, etc. Potential-based on-line approaches and their advantages are also discussed.

Keywords: Potentials, Poisson equations, gradient-based policy iteration, perturbation realization, Q-learning, TD(λ)

1. Introduction

Perturbation analysis (PA) (Cao, 1994; Cao and Chen, 1997; Cassandras and Lafortune, 1999; Fu and Hu, 1997; Glasserman, 1991; Gong and Ho, 1987; Ho and Cao, 1991) provides performance sensitivities for a discrete event dynamic system (DEDS) by analyzing the dynamical behavior of a single sample path of the DEDS. Two types of sensitivities are considered: sensitivities in continuous spaces (performance gradients), and sensitivities in discrete spaces (performance differences). Performance optimization can be achieved by combining PA with stochastic approximation methods. Markov decision process (MDP) (Bertsekas, 1995; Feinberg and Shwartz, 2002; Meyn, 1997; Meyn and Tweedie, 1993; Puterman, 1994) is a general model for performance optimization of DEDSs. Policy iteration, the basic approach in MDPs, can be implemented based on sample paths. The goal of reinforcement learning (RL) (Bertsekas, 1995; Bertsekas and Tsitsiklis, 1996; Singh, 1994; Smart and Kaelbling, 2000; Sutton, 1988; Sutton and Barto, 1998) is to learn how to make decisions to improve a system's performance by observing its behavior. In this paper, we study the relations among these closely related fields. Our study reveals that such relations can be concisely illustrated by Figure 1. We show that MDP solutions can be derived naturally from performance sensitivity analysis in discrete spaces provided by PA; performance potentials play a

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

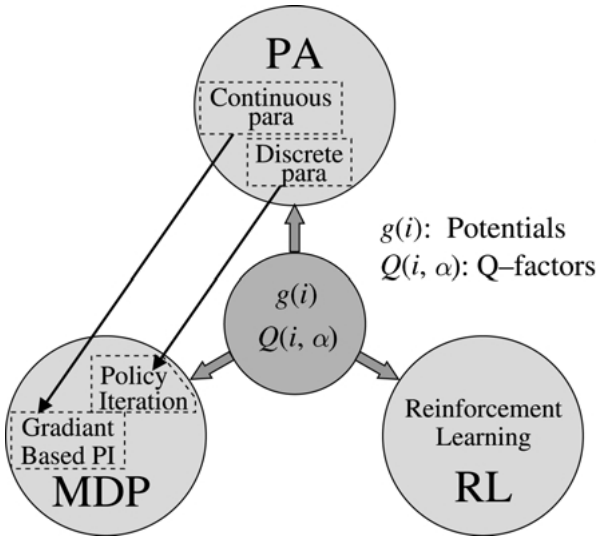

Figure 1. The relations among PA, MDP, and RL.

crucial role in MDP, PA, and other optimization approaches; and reinforcement learning, TD(λ), neuro-dynamic programming, etc, are sample-path-based efficient ways of estimating the performance potentials and other related quantities (e.g., Q-factors). This sensitivity-based view of PA, MDP, and RL brings in some new insights to the area of learning and optimization. In particular, gradient-based optimization can be applied to parameterized systems with large state spaces, and gradient-based policy iteration can be applied to some nonstandard MDPs such as systems with correlated actions, etc. Potential-based on-line approaches and their advantages are also discussed.

We will discuss the details illustrated in Figure 1 in subsequent sections. In Section 2, we start with a brief introduction to the basic principles of PA. We first discuss queueing networks and then extend the results to Markov processes. The special structure of queueing networks makes the sensitivity algorithms very efficient. In Section 2.1, we introduce the main concept of *realization factor* by using closed queueing networks. We show that realization factor measures the average effect of a single perturbation on system performance. We intuitively explain how the effect of a change in a parameter value on the system performance can be decomposed into the effects of many single small perturbations on the performance. From this, we derive the sensitivity formula: the normalized derivative equals the mean of the realization factor. In Section 2.2, we apply the above basic ideas of PA to Markov chains to obtain the performance gradients. We define realization factors for Markov chains, and from which *performance potentials* are introduced and performance derivatives with respect to changes in transition probability matrices are obtained. In Section 2.3, the results are extended to discrete performance sensitivities, i.e., the difference of the performance measures of any two policies. In Section 3, we show that policy iteration in MDP can be easily derived from the discrete sensitivity formula. Gradient-based policy iteration is discussed in Section 3.2.

Performance potentials play a crucial role in both PA and MDP. In Section 4, we extend the results to MDPs with discounted performance criteria. We define α-potentials and α-Poisson equations ($0 < \alpha \leq 1$ is a discount factor) and show that potential theory provides a unified framework to performance sensitivity analysis and policy iteration with both long-run average-cost and discounted performance criteria. The average-cost problem corresponds to the case $\alpha = 1$. In Section 5, we discuss the problem of on-line optimization. In Section 5.1, some sample path-based estimation algorithms for potentials are derived. In Section 5.2, the advantage of potential-based on-line optimization is discussed and illustrated by an example. In Section 5.3, sample path based algorithms for performance gradients are presented; these algorithms can be applied to parameterized systems with large state spaces. In Section 6, we give a brief review of Q-learning, TD(λ), and neuro-dynamic programming, which provide efficient ways in estimating the potentials and Q-factors, the latter are used in place of potentials when the system structure is completely unknown. In Section 7, we give a brief summary as a conclusion. Figure 2 is a diagram that shows the relations among the sections and subsections in this paper.

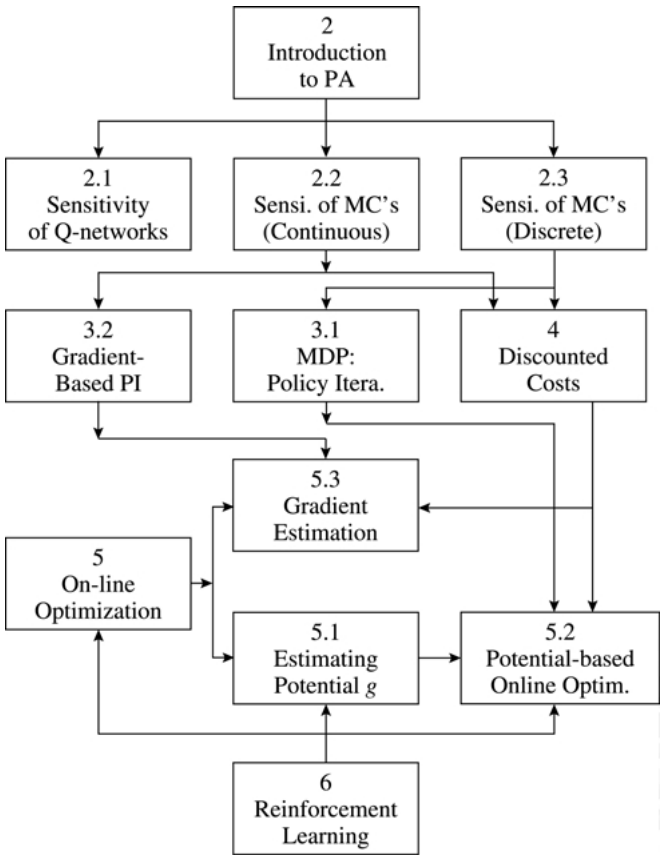

Figure 2. Dependency among the sections.

2. Perturbation Analysis via Perturbation Realization

The basic principle for PA can be explained by using the concept of *perturbation realization*. We first introduce the main ideas with a closed queueing network, and then extend the results to Markov chains.

2.1. Performance Sensitivity of Queueing Systems

Consider an M-server closed Jackson (Gordon-Newell) network in which service times are exponentially distributed with means $\bar{s}_i$, $i = 1, 2, \ldots, M$, respectively, where M is the number of servers in the network. Let $q_{i,j}$, $i,j = 1, 2, \ldots, M$, be the routing probability from Server i to Server j in the closed network. The system state is $\mathbf{n} = (n_1, \ldots, n_M)$, with n_l being the number of customers at Server l. Let $\mathbf{N}(t)$, $t \in [0, \infty)$, denote a sample path ($\mathbf{N}(t)$ is the system state at time t). $\mathbf{N}(t)$ is left continuous. For irreducible networks (in which a customer can reach any server in the network while circulating in the network), the system performance is defined as the long-run average

$$\eta^{(f)} = \lim_{L\to\infty} \frac{1}{L} \int_0^{T_L} f(\mathbf{N}(t))dt = \lim_{L\to\infty} \frac{F_L}{L}, \qquad F_L = \int_0^{T_L} f(\mathbf{N}(t))dt \tag{1}$$

where f is a performance mapping from the state space to $\mathscr{R} = (-\infty, \infty)$, and T_L is the L-th transition time of the system. For example, if $f(\mathbf{n}) = I(\mathbf{n}) \equiv 1$, then $F_L = T_L$ and $\eta^{(I)} = \lim_{L\to\infty} \frac{T_L}{L} = \frac{1}{\eta}$, where $\eta = \lim_{L\to\infty} \frac{L}{T_L}$ is the system throughput, i.e., number of transitions per unit of time. If $f(\mathbf{n}) = n_i$, then F_L is the area underneath the sample path of Server i, and

$$\eta^{(f)} = \lim_{L\to\infty} \frac{F_L}{L} = \left(\lim_{L\to\infty} \frac{T_L}{L}\right)\left(\lim_{L\to\infty} \frac{F_L}{T_L}\right) = \eta^{(I)}\bar{n}_i$$

where $\bar{n}_i$ is the average number of customers in Server i.

The solid lines in Figure 3(A) illustrate a sample path $\mathbf{N}(t)$ of a two-server two-customer cyclic queueing network with transition instants T_1 to T_5.

We first study the average effect of a small perturbation on F_L. Suppose that at some time t the system is in state $\mathbf{n}$ and Server i obtains a small *perturbation* Δ, which means that the service completion time of the customer being served in Server i at time t is delayed by a small amount Δ (as illustrated by the Δ at the service completion time T_2 of Server 1 in Figure 3(A), $\mathbf{N}(T_2) = (2, 0)$. For convenience, we also say that Server 1 has a perturbation Δ in $[T_1, T_2]$). This perturbation will affect the service completion times of other customers in the system according to some simple rules (called *perturbation propagation rules* in literature (Ho and Cao, 1991)). For example, the service completion time of the next customer in the same server, T_3, will be delayed by the same amount Δ. We say that the perturbation Δ is *propagated* to the next customer in the same server. In addition, at T_2, the service starting time of the customer of Server 2 will be delayed by Δ since it was idle

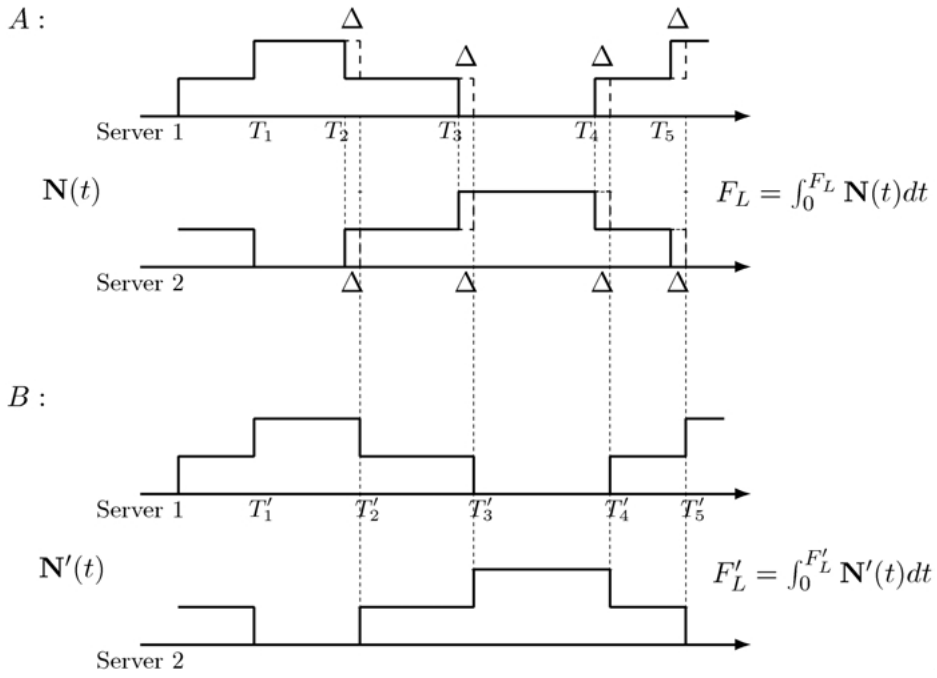

Figure 3. A sample path and its perturbed counterpart.

before the customer arrives from Server 1. We say that the perturbation is *propagated* from Server 1 to Server 2 via an idle period of Server 2. The delay Δ at T_2 at Server 2 will affect the service completion times of later customers at Server 2, etc. All the perturbations due to propagation are illustrated by the dashed lines and denoted by Δ in Figure 3(A). Thus, the sample path of the perturbed system will differ from $\mathbf{N}(t)$, and we call it a *perturbed sample path* and denote it as $\mathbf{N}'(t)$, as shown in Figure 3(B). T'_k, $k = 1, 2, \ldots$, are the transition instants of the perturbed path.

We define the *realization factor* of a perturbation Δ of Server i in state $\mathbf{n}$ for the long-run average-cost performance $\eta^{(f)}$ as

$$\begin{aligned} c^{(f)}(\mathbf{n}, i) &= \lim_{L\to\infty} E\left[\frac{\Delta F_L}{\Delta}\right] = \lim_{L\to\infty} E\left[\frac{F'_L - F_L}{\Delta}\right] \\ &= \lim_{L\to\infty} E\left[\frac{1}{\Delta}\left(\int_0^{T'_L} f(\mathbf{N}'(t))dt - \int_0^{T_L} f(\mathbf{N}(t))dt\right)\right] \end{aligned} \tag{2}$$

It is clear that the realization factor $c^{(f)}(\mathbf{n}, i)$ measures the average effect of a perturbation at $(\mathbf{n}, i)$ on F_L in (1) as $L\to\infty$.

Now we introduce the notion of *perturbation realization*. We know that a perturbation Δ (e.g., the one at T_2 at Server 1 in Figure 3(A)) will be propagated along a sample path from one server to the other through idle periods. A perturbation can also be lost via idle

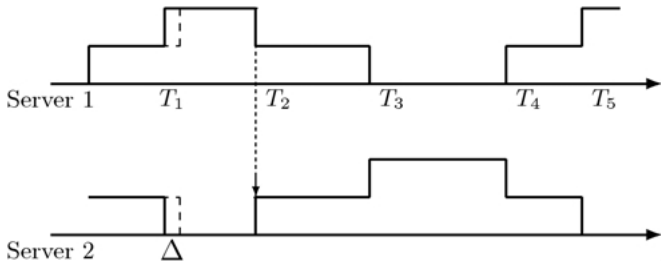

Figure 4. A perturbation of Server 2 at T_1 is lost after the idle period.

periods. In Figure 4, Server 2 obtains a perturbation at T_1, this perturbation is lost after the idle period of Server 2, $[T_1, T_2]$, because the starting time of the customer of the next busy period does not change. Thus, after T_2, the perturbed sample path is exactly the same as the original one.

As discussed above, a perturbation will be propagated or lost through idle periods. It can be easily proved that in an irreducible closed network, if a server obtains a perturbation Δ, then eventually (with probability one) every server in the network will get the same perturbation Δ or every server will loss the perturbation. If every server gets the perturbation, we say that the perturbation is *realized* by the network. After a perturbation is realized, the perturbed sample path is the same as the original one except it is shifted to the right by the amount Δ. That is, there is an L^*, such that $T'_L = T_L + \Delta$ for all $L \geq L^*$. If a perturbation is lost, then the perturbed sample path is exactly the same as the original one; that is, there is an L^*, such that $T'_L = T_L$ for all $L \geq L^*$. In both cases, there is an L^* (depending on the sample path) such that

$$\int_{T_{L^*}}^{T_L} f(\mathbf{N}(t))dt - \int_{T'_{L^*}}^{T'_L} f(\mathbf{N}'(t))dt = 0$$

for all $L \geq L^*$. (In both Figures 3 and 4, $L^* = 2$.) Therefore, (2) becomes

$$c^{(f)}(\mathbf{n}, i) = E\left[\frac{1}{\Delta}\left(\int_0^{T'_{L^*}} f(\mathbf{N}'(t))dt - \int_0^{T_{L^*}} f(\mathbf{N}(t))dt\right)\right] \tag{3}$$

Thus, $c^{(f)}(\mathbf{n}, i)$ defined in (2) is finite.

It is proved that $c^{(f)}(\mathbf{n}, i)$ can be uniquely determined by the following set of linear equations (Cao, 1994).

1. If $n_i = 0$, then $c^{(f)}(\mathbf{n}, i) = 0$;

2. $\sum_{i=1}^{M} c^{(f)}(\mathbf{n}, i) = f(\mathbf{n})$; and,

3. Let $\mathbf{n}_{i,j} = (n_1, \ldots, n_i - 1, \ldots, n_j + 1, \ldots, n_M)$ be a neighboring state of $\mathbf{n}$, then

$$\left\{\sum_{i=1}^{M} \varepsilon(n_i)\mu_i\right\} c^{(f)}(\mathbf{n}, k) = \sum_{i=1}^{M}\sum_{j=1}^{M} \varepsilon(n_i)\mu_i q_{i,j} c^{(f)}(\mathbf{n}_{i,j}, k) + \sum_{j=1}^{M} \mu_k q_{k,j}\left\{(1 - \varepsilon(n_j))c^{(f)}(\mathbf{n}_{k,j}, j) + f(\mathbf{n}) - f(\mathbf{n}_{k,j})\right\} \quad n_k > 0, k \in \Gamma \tag{4}$$

The above equations are easy to derive. Property 1 is simply a convention: when a server is idle, no service completion time can be delayed. Property 2 is a direct consequence of the definition of realization factor. $\sum_{i=1}^{M} c^{(f)}(\mathbf{n}, i)$ measures the total effect of M perturbations together, one at each server. Suppose that at t the system state is $\mathbf{n}$ and all servers' service completion times have the same perturbation Δ, then the perturbed path is the same as the original one except that it is shifted to the right by the amount Δ starting from t, i.e., $T'_k = T_k + \Delta$ for all $T_k \geq t$. For example, in Figure 3(A), at $t = T'_2$, both servers have the same perturbation Δ; we have

$$\begin{aligned} F'_L - F_L &= \int_0^{T'_L} f(\mathbf{N}'(t))dt - \int_0^{T_L} f(\mathbf{N}(t))dt \\ &= \left\{\int_0^{T'_2} f(\mathbf{N}'(t))dt - \int_0^{T_2} f(\mathbf{N}(t))dt\right\} + \left\{\int_{T'_2}^{T'_L} f(\mathbf{N}'(t))dt - \int_{T_2}^{T_L} f(\mathbf{N}(t))dt\right\} \\ &= \int_0^{T'_2} f(\mathbf{N}'(t))dt - \int_0^{T_2} f(\mathbf{N}(t))dt \\ &= \left\{\int_0^{T_2} f(\mathbf{N}'(t))dt - \int_0^{T_2} f(\mathbf{N}(t))dt\right\} + \int_{T_2}^{T'_2} f(\mathbf{N}'(t))dt \\ &= \int_{T_2}^{T'_2} f(\mathbf{N}'(t))dt = f(2,0)\Delta \end{aligned}$$

Equation (4) can be derived by the theorem of total probability. We note that $\varepsilon(n_i)\mu_i q_{i,j} / \sum_{i=1}^{M} \varepsilon(n_i)\mu_i$ is the probability that the next transition is from Server i to Server j, $i, j = 1, 2, \ldots, M$. $f(\mathbf{n}) - f(\mathbf{n}_{k,j})$ is the effect due to the delay of the transition from Server k to Server j. Equation (4) implies that the effect of a perturbation before a transition equals the weighted sum, by probability, of the effects of the perturbations after the transition, plus the effect due to the delay of the transition. The term with $(1 - \varepsilon(n_j))$ reflects the effect of the perturbation propagated via an idle period of Server j, $j = 1, \ldots, M$. It has been proved that (4) and the equations in Properties 1 and 2 have a unique solution for irreducible close Jackson networks (Cao, 1994).

We have quantified the effect of a single perturbation on the long-run average-cost performance. However, a small change in a system parameter may induce many perturbations. Suppose that the mean service time of a server, say Server i, changes from $\bar{s}_i$

to $\bar{s}_i + \Delta\bar{s}_i$. Because the service times are exponentially distributed, every customer's service time, denoted as s, will change to $s\bar{s}_i + \Delta\bar{s}_i/\bar{s}_i = s + s\Delta\bar{s}_i/\bar{s}_i$. That is, every customer's service time at Server i will gain a perturbation of a size proportional to it with a factor $\Delta\bar{s}_i/\bar{s}_i$. More precisely, if the first customer's service time at Server i is $s_{i,1}$, the second customer's service time at Server i is $s_{i,2}$, etc, then the first customer's service completion time will gain a perturbation $\Delta_{i,1} = s_{i,1}\Delta\bar{s}_i/\bar{s}_i$, and the second customer's service completion time will gain an additional perturbation $\Delta_{i,2} = s_{i,2}\Delta\bar{s}_i/\bar{s}_i$, etc. Each of these perturbations will be propagated along the sample path in the same fashion as illustrated in Figure 3(A). To calculate the effect of a small change in the mean service time $\bar{s}_i$, the effects of all these single perturbations have to be taken into account.

Let $p(\mathbf{n})$ be the steady-state probability of state $\mathbf{n}$. Consider a time period $[0, T_L]$ with $L >> 1$. The length of the total time that the system is in state $\mathbf{n}$ in $[0, T_L]$ is $T_L p(\mathbf{n})$. The total perturbation induced in this period at Server i due to the change $\Delta\bar{s}_i$ in the mean service time is $T_L p(\mathbf{n})\Delta\bar{s}_i/\bar{s}_i$. Since each perturbation on the average has an effect of $c^{(f)}(\mathbf{n}, i)$ on F_L, the overall effect on F_L of all the perturbation induced when the system state is $\mathbf{n}$ is $(T_L p(\mathbf{n})\Delta\bar{s}_i/\bar{s}_i)c^{(f)}(\mathbf{n}, i)$. Finally, the total effect of the mean service time change, $\Delta\bar{s}_i$, on F_L is

$$\Delta F_L = \sum_{all\ \mathbf{n}} T_L p(\mathbf{n}) \frac{\Delta\bar{s}_i}{\bar{s}_i} c^{(f)}(\mathbf{n}, i)$$

From this, we have

$$\frac{\bar{s}_i}{T_L/L}\frac{\Delta F_L/L}{\Delta\bar{s}_i} = \sum_{all\ \mathbf{n}} p(\mathbf{n})c^{(f)}(\mathbf{n}, i)$$

Letting $L \to \infty$ and $\Delta\bar{s}_i \to 0$, we obtain the steady-state performance sensitivity as follows:

$$\frac{\bar{s}_i}{\eta^{(I)}}\frac{\partial\eta^{(f)}}{\partial\bar{s}_i} = \sum_{all\ \mathbf{n}} p(\mathbf{n})c^{(f)}(\mathbf{n}, i) \tag{5}$$

where

$$\eta^{(I)} = \lim_{L\to\infty}\frac{T_L}{L} = \lim_{L\to\infty}\frac{1}{L}\int_0^{T_L} I(\mathbf{n})dt$$

in which $I(\mathbf{n}) = 1$ for all $\mathbf{n}$. Thus, the *normalized* performance sensitivity (the left-hand side of (5)) equals the steady-state expectation of the realization factor. Very efficient algorithms can be developed to estimate the normalized sensitivity directly based on a single sample path (Ho and Cao, 1983, 1991). (In computer simulation, only a few lines have to be added to the program in order to get the performance sensitivity of a queueing network in addition to performance itself (Ho and Cao, 1991).) Typically, 5–10% additional CPU times are needed for the derivatives with respect to all mean service times.

This approach provides multi-dimensional gradients with one simulation.) The above discussion offers an intuitive explanation of the sensitivity formula; a rigorous proof involves exchanging the order of $\lim_{L\to\infty}$ and $\Delta\bar{s}_i \to 0$, see Cao (1994).

In summary, the basic principle of PA is as follows. A small change in a system parameter (such as the mean service time of a server) induces a series of changes on a sample path; each change is called a *perturbation* of the sample path. The average effect of each perturbation on the system performance can be precisely measured by a quantity called perturbation *realization factor*. The total effect of the small change in the parameter on the system performance can then be calculated by adding together the average effects of all the perturbations induced by the parameter change. The sensitivity of the performance with respect to the parameter can then be determined. For more details, see Cao (1994) and Ho and Cao (1991).

2.2. *Performance Sensitivity of Markov Chains: the Continuous Case*

The above idea was extended to the case of Markov chains in Cao (1998) and Cao and Chen (1997). Consider an irreducible and aperiodic Markov chain $\mathbf{X} = \{X_n : n \geq 0\}$ on a finite state space $\mathcal{S} = \{1, 2, \ldots, M\}$ with transition probability matrix $P = [p(i,j)] \in [0,1]^{M\times M}$. Let $\pi = (\pi_1, \ldots, \pi_M)$ be the (row) vector representing its steady-state probabilities, and $f = (f_1, f_2, \ldots, f_M)^T$ be the (column) performance vector, where ''T'' represents transpose. We have $Pe = e$, where $e = (1, 1, \ldots, 1)^T$ is an M-dimensional vector whose all components equal 1, and $\pi = \pi P$, which is the standard probability flow balance equation at steady state. The performance measure is the long-run average defined as (cf. (1))

$$\eta = E_\pi(f) = \sum_{i=1}^{M} \pi_i f_i = \pi f = \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} f(X_l) = \lim_{L\to\infty} \frac{F_L}{L} \tag{6}$$

where

$$F_L = \sum_{l=0}^{L-1} f(X_l)$$

Let P' be another irreducible transition probability matrix on the same state space. Suppose P changes to $P(\delta) = P + \delta Q = \delta P' + (1-\delta)P$, with $\delta > 0$, $Q = P' - P = [q(i,j)]$. Since $Pe = P'e = e$, we have $Qe = 0$ and $P(\delta)e = e$. The performance measure will change to $\eta(\delta) = \eta + \Delta\eta(\delta)$. The derivative of η in the direction of Q is defined as $d\eta/d\delta = \lim_{\delta\to 0} \Delta\eta/\delta$.

In this system, a perturbation means that the system is perturbed from one state i to another state j. For example, consider the case where $p(k,i) = 0.5$, $p(k,j) = 0.5$, and $p(k,l) = 0$ for all $l \neq i, j$. In computer simulation, to determine the next state that the Markov chain will jump into from state k, we generate a $[0,1]$ uniformly distributed random variable ξ. If $0 \leq \xi < 0.5$, then the Markov chain jumps into state i; otherwise, it

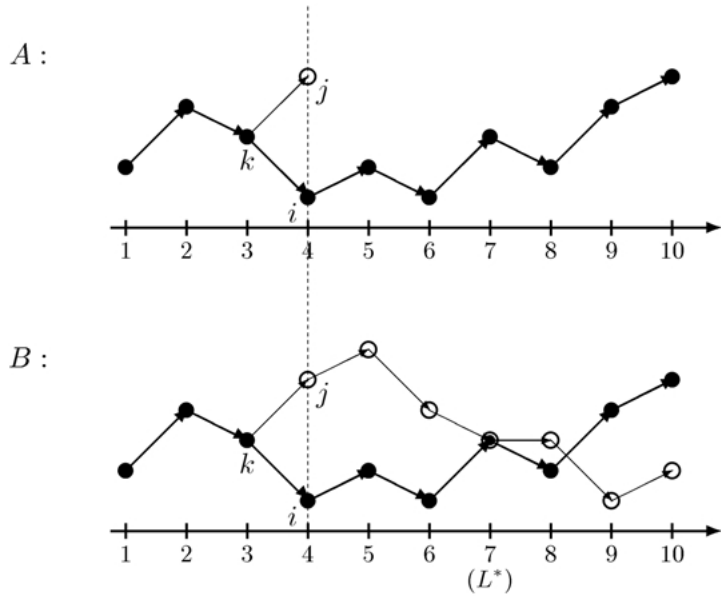

Figure 5. A perturbation and its effect on performance.

jumps into state j. Now suppose that the transition probabilities are perturbed to $p'(k,i) = 0.5 - \delta$, $p'(k,j) = 0.5 + \delta$, and $p'(k,l) = 0$ (i.e., $q(k,i) = -1$, $q(k,j) = 1$, and $q(k,l) = 0$.) Assume that $0.5 - \delta \leq \xi < 0.5$, then the original Markov chain jumps into state i, but the perturbed one jumps into j. This is illustrated in Figure 5(A), which shows a sample path of a Markov chain. At $t = 4$, the sample path is perturbed from state i to j. Figure 5(B) illustrates the effect of this perturbation. At $t = L^*$, the perturbed path (shown as the thin lines) merges with the original one. By the strong Markov property, after L^* both paths have the same statistic behavior. Therefore, the effect of the perturbation takes place in the period from $t = 4$ to L^*. Because δ is very small, we can assume that jumps occur very rarely; in particular, we can assume that between $t = 4$ and $L^* = 7$ there are no other jumps (more precisely, the probability that there is one jump at $t = 4$ and another jump between $t = 4$ to $L^* = 7$ is in the order of δ^2). In other words, we can assume that the thin line in Figure 5B evolves according to the same transition probability P from $t = 4$ to $L^* = 7$.

Thus, we study two independent Markov chains $\mathbf{X} = \{X_n; n \geq 0\}$ and $\mathbf{X}' = \{X'_n; n \geq 0\}$ with $X_0 = i$ and $X'_0 = j$; both of them have the same transition matrix P. The *realization factor* is defined as Cao and Chen (1997) (cf. 2):

$$\begin{aligned} d(i,j) &= \lim_{L\to\infty} E[F'_L - F_L | X'_0 = j,\ X_0 = i,] \\ &= \lim_{L\to\infty} E\left[\sum_{l=0}^{L-1} (f(X'_l) - f(X_l)) \middle| X'_0 = j,\ X_0 = i\right], \qquad i,j = 1,\ldots,M \end{aligned} \tag{7}$$

Thus, $d(i,j)$ represents the average effect of a jump from i to j on F_L in (6).

If P is irreducible, then with probability one the two sample paths of $\mathbf{X}$ and $\mathbf{X}'$ will merge together. That is, there is a random number L^* such that $X'_{L^*} = X_{L^*}$. Therefore, by the strong Markov property, (7) becomes

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

Note that (7) and (8) are similar to (2) and (3), respectively. Essentially, they use the difference of the perturbed path and the original one to measure the effect of a single perturbation. The matrix $D \in \mathscr{R}^{M\times M}$, with $d(i,j)$ as its (i,j)th element, is called a realization matrix. From (8), we have

$$\begin{aligned} d(i,j) &= f(j) - f(i) + \sum_{i'=1}^{M}\sum_{j'=1}^{M} E\left(\sum_{l=1}^{L^*-1} [f(X'_n) - f(X_n)] | X'_1 = j', X_1 = i' \right) \\ &\quad \times P\{X'_1 = j', X_1 = i' | X'_0 = j, X_0 = i\} \\ &= f(j) - f(i) + \sum_{i'=1}^{M}\sum_{j'=1}^{M} p(i,i')p(j,j')d(i',j') \end{aligned} \tag{9}$$

Writing this in a matrix form, we have the Lyapunov equation (Cao and Chen, 1997)

$$D - PDP^T = F \tag{10}$$

where $F = ef^T - fe^T$.

Equation (7) suggests the following definition for $g(i)$:

$$g(i) \approx E\left[\sum_{l=0}^{L-1} f(X_l) | X_0 = i \right] = E[F_L | X_0 = i] \tag{11}$$

Since $d(i,j)$ measures the difference of the performance starting from states j and i, so $g(i)$ measures the averge contribute of every visit of state i to F_L. Furthermore, only the difference between different $g(i)$s are important for performance sensitivities. Now we consider a sample path consisting of L transitions. Among these transitions, on the average there are $L\pi_i$ transitions at which the system are at state i. After being at state i, the system jumps to state j on the average $L\pi_i p(i,j)$ times. If the transition probability matrix P changes to $P(\delta) = P + \delta Q$, then the change in the number of visits to state j after being at state i is $L\pi_i q(i,j)\delta = L\pi_i[p'(i,j) - p(i,j)]\delta$. This contributes a change of $\{L\pi_i[p'(i,j) - p(i,j)]\delta\}g(i)$ to F_L. Thus, the total change in F_L due to the change of P to $P(\delta)$ is

$$\Delta F_L = \sum_{i,j=1}^{M} L\pi_i[p'(i,j) - p(i,j)]\delta g(i) = \pi[P' - P]g\delta L = \pi Qg\delta L$$

where $g = (g(1), \ldots, g(M))^T$. Finally, using $d\eta/d\delta = \lim_{\delta\to 0} \frac{1}{\delta}\Delta F_L/L$, we have

$$\frac{d\eta}{d\delta} = \pi Qg \tag{12}$$

This is similar to (5).

Equation (11) is only an approximation. When $L\to\infty$, this expression is unbounded. However, because $Qe = 0$, we have $Q(g + ce) = Qg$ for any real constant c. Thus, the sensitivity equation (12) holds when g is replaced by $g + ce$. (This confirms the fact that g is only important up to an additive constant.) Therefore, we may add a constant $-L\eta$ to (11) and obtain

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

Letting $L\to\infty$, we obtain the formal definition of the *performance potential* at state i:

$$g(i) = \lim_{L\to\infty} E\left\{\sum_{l=0}^{L-1}[f(X_l) - \eta]|X_0 = i\right\} \tag{13}$$

which can be proved to be finite for ergodic chains. Equation (12) holds with $g = (g(1), \ldots, g(M))^T$ and $g(i)$ defined in (13). Again, the above discussion only provides an intuitive explanation; for a rigorous proof, see Cao and Chen (1997).

We have $d(i,j) = g(j) - g(i)$ and $D = eg^T - ge^T$. From this, we have

$$\frac{d\eta}{d\delta} = \pi QD^T\pi^T \tag{14}$$

Similar to (9), we have

$$\begin{aligned} g(i) &= f(i) - \eta + \sum_{j=1}^{M}\left\{p(i,j)\lim_{L\to\infty} E\left\{\sum_{l=1}^{L-1}[f(X_l) - \eta]|X_1 = j\right\}\right\} \\ &= f(i) - \eta + \sum_{j=1}^{M} p(i,j)g(j) \end{aligned}$$

In a Matrix form, this is the following *Poisson equation*

$$(I - P)g + \eta e = f \tag{15}$$

Equations (10) and (15) are similar to (4). These equations quantitatively determine the effect of a single perturbation. The solution to (15) is only up to a constant; i.e., if g satisfies (15), then for any constant c, $g + ce$ also does. Therefore, there is one particular solution to (15) (still denoted as g) such that $\pi g = \eta$. For this solution, (15) becomes

$$(I - P + e\pi)g = f \tag{16}$$

For ergodic Markoc chains, $(I - P + e\pi)$ is invertable, thus $g = (I - P + e\pi)^{-1} f$, where $(I - P + e\pi)^{-1}$ is called a *fundamental matrix*.

2.3. *Performance Sensitivity of Markov Chains: the Discrete Case*

In sensitivity analysis with discrete parameters, we wish to obtain the difference of the performance measures of two Markov chains with transition probability matrices P and P', respectively, both are ergodic on the same state space. We use prime ‘‘$'$’’ to denote the values associated with P'.

First, multiplying both sides of the Poisson equation (16) with π on the left, we get

$$\pi g = \pi f = \eta \tag{17}$$

Next, multiplying both sides of the Poisson equation with π' on the left yields

$$\pi' Q g = \pi'(P' - P)g = \pi'(I - P)g = \pi' f - \pi g = \pi' f - \eta$$

That is,

$$\eta' - \eta = \pi' Q g \tag{18}$$

Equation (18) can be viewed as the discrete version of the performance sensitivity. Note that the discrete sensitivity (18) can be obtained from its continuous counterpart (12) by replacing π with π'.

For more general cases, we assume that the performance function also changes from f to f'. Let $h = f' - f$. It is easy to check that

$$\eta' - \eta = \pi'(Qg + h) \tag{19}$$

For continuous sensitivity, we set $f(\delta) = f + \delta h$. Together with $P(\delta) = P + \delta Q$, we have

$$\frac{d\eta}{d\delta} = \lim_{\delta \to 0} \frac{1}{\delta} \frac{\Delta F_L}{L} = \pi(Qg + h) \tag{20}$$

There is a fundamental difference between the continuous sensitivity (20) (or (12)) and its discrete counterpart (19) (or (18)). Given a transition matrix P, one can obtain π and g,

either by solving the flow balance equation and the Poisson equation, or by sample path estimation (cf. Section 5). By (20), we can obtain the performance derivative along any direction $Q = P' - P$ for any given P' without resorting to any quantities for the Markov chain with P'. This feature serves as the basis for the gradient-base policy iteration for MDPs proposed later in Section 3.2. On the other hand, with the discrete sensitivity equation, to know the exact value of the performance difference from (19), one needs to know π'. However, if π' is known, one can get η' directly by $\pi' f$. In addition, in optimization problems it is impossible to calculate π' for all different P's since the Markov chains to be compared are usually too many. Fortunately, since $\pi' > 0$ (componentwise) for all ergodic chains, we know that if $Qg + h > 0$ (componentwise), then $\eta' > \eta$. That is, we may be able to know which one is better without solving for π'. This is the basis for policy iteration in MDP discussed in the next section.

3. Markov Decision Processes

3.1. Policy Iteration

In this section, we show that policy iteration is a natural consequence of the discrete performance sensitivity formula (19).

For two M-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$, we define $a = b$ if $a(i) = b(i)$ for all $i = 1, 2, \ldots, M$; $a \leq b$ if $a(i) < b(i)$ or $a(i) = b(i)$ for all $i = 1, 2, \ldots, M$; $a < b$ if $a(i) < b(i)$ for all $i = 1, 2, \ldots, M$; and $a \preceq b$ if $a(i) < b(i)$ for at least one i, and $a(j) = b(j)$ for other components. The relation $\leq$ includes $=$, $\preceq$, and $<$. Similar definitions are used for the relations $>$, $\succeq$, and $\geq$.

Next, we note that $\pi'(i) > 0$ for all $i = 1, 2, \ldots, M$. Thus, from (19), we know that if $Qg + h = (P' - P)g + (f' - f) \succeq 0$ then $\eta' - \eta > 0$. From (19) and the fact $\pi' > 0$, the proof of the following lemma is straightforward.

LEMMA 1 *If* $Pg + f \preceq P'g + f'$, *then* $\eta < \eta'$.

It is interesting to note that in the lemma, we use only the potentials with one Markov chain, i.e., g. Thus, if the condition in Lemma 1 holds, only the potentials with one policy is needed to compare the performance measures under two policies.

In an MDP, at any transition instant $n \geq 0$ of a Markov chain $\mathbf{X} = \{X_n, n \geq 0\}$, an action is chosen from an action space $\mathscr{A}$ and is applied to the Markov chain. We assume that the number of actions is finite, and we only consider stationary policies. The actions that are available for $i \in \mathscr{S}$ form a nonempty subset $A(i) \subseteq \mathscr{A}$. A stationary policy is a mapping $\mathscr{L} : \mathscr{S} \to \mathscr{A}$, i.e., for any state i, $\mathscr{L}$ specifies an action $\mathscr{L}(i) \in A(i)$. Let $\mathscr{E}$ be the policy space. If action α is taken at state i, then the state transition probabilities at state i are denoted as $p^{\alpha}(i,j)$, $j = 1, 2, \ldots, M$, which depends on α. With a policy $\mathscr{L}$, the Markov process evolves according to the transition matrix $P^{\mathscr{L}} = [p^{\mathscr{L}(i)}(i,j)]_{i=1}^{M}|_{j=1}^{M}$. We use the superscript $*^{\mathscr{L}}$ to denote the quantities associated with policy $\mathscr{L}$.

For most stable systems in real world applications, the Markov chains under any policy are recurrent. For example, in any stable communication systems, from any state it is

always possible to reach the null state where there are no packets in the system. Therefore, in this paper we assume that the Markov chain is recurrent under any policy. In addition, for simplicity we assume that it is aperiodic; although all the results in this paper hold for periodic chains if we replace the steady state value by the corresponding time average value (Puterman, 1994). Therefore, the Markov chains considered are ergodic (Çinlar, 1975). This corresponds to the problems classified in Puterman (1994) as the recurrent case.

The steady-state probabilities corresponding to policy $\mathscr{L}$ is denoted as a vector $\pi^{\mathscr{L}} = (\pi^{\mathscr{L}}(1), \ldots, \pi^{\mathscr{L}}(M))$. Suppose that at each stage with state i and control action $\alpha \in A(i)$, a cost $f(i, \alpha) = f(i, \mathscr{L}(i))$ is incurred. The long-run expected value of the average cost corresponding to policy $\mathscr{L}$ is then

$$\eta^{\mathscr{L}} = \lim_{L \to \infty} \frac{1}{L} E\left\{ \sum_{l=0}^{L-1} f[X_l, \mathscr{L}(X_l)] \right\}$$

For ergodic chains, the above limit exists and does not depend on the initial state. Our objective is to minimize this average cost over the policy space $\mathscr{S}$, i.e., to obtain $\min_{\mathscr{L} \in \mathscr{E}} \eta^{\mathscr{L}}$.

Define $f^{\mathscr{L}} = (f[1, \mathscr{L}(1)], \ldots, f[M, \mathscr{L}(M)])^T$. Equations (16) and (17) becomes

$$(I - P^{\mathscr{L}} + e\pi^{\mathscr{L}})g^{\mathscr{L}} = f^{\mathscr{L}} \tag{21}$$

and

$$\pi^{\mathscr{L}} g^{\mathscr{L}} = \pi^{\mathscr{L}} f^{\mathscr{L}}$$

The following optimality theorem follows almost immediately from Lemma 1, which is derived directly from the sensitivity formula (19).

THEOREM 1 *A policy $\mathscr{L}$ is optimal if and only if*

$$P^{\mathscr{L}} g^{\mathscr{L}} + f^{\mathscr{L}} \leq P^{\mathscr{L}'} g^{\mathscr{L}} + f^{\mathscr{L}'} \tag{22}$$

for all $\mathscr{L}' \in \mathscr{E}$.

The optimality condition (22) is, of course, equivalent to the other conditions in the literature. To see this, we rewrite (21) in the following form:

$$\eta^{\mathscr{L}} e + g^{\mathscr{L}} = f^{\mathscr{L}} + P^{\mathscr{L}} g^{\mathscr{L}} \tag{23}$$

Then Theorem 1 becomes: A policy $\mathscr{L}$ is optimal if and only if

$$\eta^{\mathscr{L}} e + g^{\mathscr{L}} = \min_{\mathscr{L}' \in \mathscr{E}} \{P^{\mathscr{L}'} g^{\mathscr{L}} + f^{\mathscr{L}'}\} \tag{24}$$

The minimum is taken for every component of the vector. Equation (24) is Bellman's optimality equation, which is the basis for the value iteration approach in literature (Bertsekas, 1995). From (24), $g^{\mathscr{L}}$ is equivalent to the ''differential'' or ''relative cost vector'' in Bertsekas (1995), or the ''bias'' in Puterman (1994). In our approach, g is directly related to the long-run expected performance, and many results, such as the existence and the uniqueness of the solution to Equation (23), the optimality condition (24), and the convergence of the optimal algorithms, become almost obvious. In addition, as shown in the Section 5, $g^{\mathscr{L}}$ can be easily estimated based on a single sample path. This is an important feature which allows optimization algorithms to be implemented online for real world systems.

Policy iteration algorithms for determining the optimal policy can be easily developed by combining Lemma 1 and Theorem 1. Roughly speaking, at the k-th step with policy $\mathscr{L}_k$, we set the policy for the next step (the $(k+1)$th step) as $\mathscr{L}_{k+1} = \arg\{\min_{\mathscr{L}}[P^{\mathscr{L}} g^{\mathscr{L}_k} + f^{\mathscr{L}}]\}$, (the minimum is taken componentiwisely, i.e., to determine an action for each state), with $g^{\mathscr{L}_k}$ being the solution to the Poisson equation for $P^{\mathscr{L}_k}$. Lemma 1 implies that performance usually improves at each iteration. Theorem 1 shows that the minimum is reached when no performance improvement can be achieved. We shall not state the details here because they are standard.

Finally, since $P^{\mathscr{L}_k}$ and $f^{\mathscr{L}_k}$ are fixed at the k-th step, we note that $(P^{\mathscr{L}} - P^{\mathscr{L}_k})g^{\mathscr{L}_k} + (f^{\mathscr{L}} - f^{\mathscr{L}_k})$ takes the minimal value (componentwisely) at $P^{\mathscr{L}_{k+1}}$. Thus, because $\pi^{\mathscr{L}_k} > 0$, from (20) the performance derivative also reaches minimum at $P^{\mathscr{L}_{k+1}}$ (i.e., the largest absolute value, since the derivative is negative). Therefore, we conclude that *policy iteration in fact chooses the steepest direction to go in the policy space.*

3.2. Gradient-Based Policy Iteration

Because in (18) π' is unknown, one has to compare the effect of the actions on the performance state by state. Therefore, the policy iteration described in Section 3.1 can be used only for problems in which actions at difference states can be chosen independently. That is, the approach does not apply to systems in which actions at different states may be related. For example, consider a system in which there are three actions available for both states 1 and 2, denoted as $\alpha_1, \alpha_2, \alpha_3$ and $\beta_1, \beta_2, \beta_3$. Suppose that the current potental is g, current actions are α_3 and β_3, and

$$\sum_{j=1}^{M} [p^{\alpha_1}(1,j) - p^{\alpha_3}(1,j)]g(j) < 0 \tag{25}$$

and

$$\sum_{j=1}^{M} \left[p^{\beta_2}(2,j) - p^{\beta_3}(2,j)\right]g(j) < 0 \tag{26}$$

Then applying the standard policy iteration we can choose α_1 and β_2 for states 1 and 2, respectively, as the new policy and according to Lemma 1 the new policy has a smaller performance. However, if the problem requires that when action i, $i = 1, 2, 3$, is used at state 1, then action β_i (with the same i) must be used at state 2, then the above pair α_1 and β_2 is not a feasible solution. Further, if we assume that

$$\sum_{j=1}^{M} [p^{\alpha_2}(1,j) - p^{\alpha_3}(1,j)]g(j) > 0 \tag{27}$$

and

$$\sum_{j=1}^{M} \left[p^{\beta_1}(2,j) - p^{\beta_3}(2,j)\right]g(j) > 0 \tag{28}$$

Then by (25) and (28), (α_1, β_1) cannot be chosen for the next iteration; and by (26) and (27), (α_2, β_2) cannot either. In this case, Lemma 1 does not help us to compare the performance of (α_1, β_1) (or (α_2, β_2)) with that of the current policy (α_3, β_3). One needs to use (18); in which, however, π' is unknown. Thus, the standard policy iteration procedure may not work for this system.

On the other hand, the performance gradient (12) or (20) does not depend on π', and both π and g can be obtained from the current policy P. Thus, principally one can solve for π and determine the smallest πQg to implement policy iteration. This is the gradient-based policy iteration method, which can be used to solve the optimization problem for systems where the actions at different states are correlated. For the aforementioned example, with the gradient-based policy iteration, we simply choose the action pair (α_i, β_i) that minimizes $\pi(1)\sum_{j=1}^{M} p^{\alpha_i}(1,j)g(j) + \pi(2)\sum_{j=1}^{M} p^{\beta_i}(2,j)g(j)$ for both states 1 and 2 for the next iteration.

A more realistic example is the M/G/1 queue with G being a phase type distribution. The system state is (n, k) with n being the number of customers and k the phase of the customer being served. In a real system, k is not observable; we need to develop the optimal policy based on n, i.e., a policy which assigns the same action to (n, k) with the same n and different k's. More details are reported in a recent paper Cao and Fang (2002). It is interesting to note that the problem is related to the partially observable MDP (Kaelbling et al., 1998), and MDP's with constraints (Altman, 1999).

Unfortunately, even if $d\eta/d\delta < 0$, we may not have $\eta' < \eta$. Furthermore, in MDPs with correlated actions the optimal performance may be obtained only at random policies. Thus, stochastic approximation in the continuous space may be used together with policy iteration. There are many open problems in this direction.

4. Problems with Discounted Performance Criteria

In this section, we extend the above results to MDPs with discounted performance criteria (Cao, 2000). We shall see, with performance potentials, both the average and the

discounted cost problems can be solved with the same framework, and the average cost problem corresponds to the case with the discount factor α equal 1. One interesting feature of this approach is that the proof of the results is the same for both the average and discounted cases, and the former is not treated as a limiting case of the latter.

Let $f(i)$, $i \in \mathscr{S}$, be a performance function and α, $0 < \alpha \leq 1$, be a discount factor. For $0 < \alpha < 1$, the performance cost is defined as a column vector $\eta_\alpha = (\eta_\alpha(1), \eta_\alpha(2), \ldots, \eta_\alpha(M))^T$ with

$$\eta_\alpha(i) = (1-\alpha)E\left\{\sum_{n=0}^{\infty} \alpha^n f(X_n) | X_0 = i\right\} \tag{29}$$

The factor $(1-\alpha)$ in (29) is used to obtain the continuity of η_α at $\alpha = 1$. In fact, we define

$$\eta_1 = \lim_{\alpha \to 1-} \eta_\alpha \tag{30}$$

It is proved in (31) and (35) that the above limit exists.

LEMMA 2 *$\eta_1 = \eta e$ with η being the average-cost performance:*

$$\eta = \pi f = \lim_{N \to \infty} \left\{ E\left[\frac{1}{N}\sum_{n=0}^{N-1} f(X_n)\right]\right\}$$

Proof: In a matrix form, we have

$$\eta_\alpha = (1-\alpha)\sum_{n=0}^{\infty} \alpha^n P^n f = (1-\alpha)(I - \alpha P)^{-1} f, \qquad 0 < \alpha < 1 \tag{31}$$

The second equation in (31) holds because for $0 < \alpha < 1$ all the eigenvalues of αP are within the unit circle (Berman and Plemmons, 1994). Next, it is easy to verify

$$(I - \alpha P)^{-1} = (I - \alpha P + \alpha e\pi)^{-1} + \frac{\alpha}{1-\alpha} e\pi, \qquad \alpha < 1 \tag{32}$$

Thus,

$$\lim_{\alpha \to 1-} (1-\alpha)(I - \alpha P)^{-1} = e\pi \tag{33}$$

The lemma then follows directly from (31) and (33). ■

Similar to (15), the α-*Poisson equation* is defined as

$$(I - \alpha P + \alpha e\pi) g_\alpha = f \tag{34}$$

g_α is called the α-*potential*, which is the same as what is defined in the classical potential theory for $0 < \alpha < 1$ (see (41) and Çinlar, 1975). Moreover, (32) and (31) leads to

$$\eta_\alpha = (1-\alpha)g_\alpha + \alpha\eta e \tag{35}$$

It becomes obvious that (30) does converge.

When $\alpha = 1$, (34) is the standard Poisson equation. From (34), we have

$$\begin{aligned} g_\alpha &= (I - \alpha P + \alpha e\pi)^{-1} f \\ &= \left\{ \sum_{n=0}^{\infty} \alpha^n (P - e\pi)^n \right\} f \\ &= \left\{ I + \left[\sum_{n=1}^{\infty} \alpha^n (P^n - e\pi) \right] \right\} f, \qquad 0 < \alpha \leq 1 \end{aligned}$$

In particular,

$$g_1 = (I - P + e\pi)^{-1} f = \left\{ I + \sum_{n=1}^{\infty} (P^n - e\pi) \right\} f$$

This is the same as the performance potentials defined for the average-cost MDPs. We have

$$\begin{aligned} \lim_{\alpha \to 1-} g_\alpha &= g_1, \\ \pi g_\alpha &= \pi f \end{aligned}$$

Now suppose that P changes to P', then π and η will change to π' and η', respectively. Let $Q = P' - P$. From (31), we get

$$\begin{aligned} \eta'_\alpha - \eta_\alpha &= \alpha(P'\eta'_\alpha - P\eta_\alpha) \\ &= \alpha(P' - P)\eta_\alpha + \alpha P'(\eta'_\alpha - \eta_\alpha) \end{aligned}$$

This leads to

$$\eta'_\alpha - \eta_\alpha = \alpha(I - \alpha P')^{-1} Q\eta_\alpha$$

Substituting (35) into the right-hand side of the above equation and noting that $Qe = 0$, we obtain

$$\eta'_\alpha - \eta_\alpha = \alpha(1-\alpha)(I - \alpha P')^{-1} Q g_\alpha, \qquad 0 < \alpha < 1 \tag{36}$$

Letting $\alpha \to 1-$, we obtain (18) as a special case.

Similar to Lemma 1, if we assume that f also changes to f', then we have

LEMMA 3 *If* $\alpha P' g_\alpha + f' \preceq \alpha P g_\alpha + f$, *then* $\eta'_\alpha < \eta_\alpha$, $0 < \alpha \leq 1$.

Proof: For $0 < \alpha < 1$, we have

$$(I - \alpha P')^{-1} = I + \alpha P' + \alpha^2 P'^2 + \cdots$$

Since the Markov chain is positive recurrent, every item in $(I - \alpha P')^{-1}$ is positive. The lemma for $0 < \alpha < 1$ follows directly from (36); for $\alpha = 1$, it is the same as Lemma 1. ■

Now, policy iteration algorithms for discounted cost can be derived easily. At the k-th step, it simply chooses the policy that minimizes $\alpha P^{\mathscr{L}} g_\alpha^{\mathscr{L}_k} + f^{\mathscr{L}_k}$, componentwisely, as the policy for the next step.

Next, we study the performance sensitivity. Let $P(\delta) = P + Q\delta$. Thus, $P(1) = P'$, $P(0) = P$. Then from (36), we have

$$\frac{d\eta_\alpha}{d\delta} = \alpha(1-\alpha)(I - \alpha P)^{-1} Q g_\alpha, \qquad 0 < \alpha < 1 \tag{37}$$

From (36) and (37), it is clear that policy iteration for discounted problems also chooses the next policy along the steepest direction. We have

$$\frac{d\eta_1}{d\delta} = e\frac{d\eta}{d\delta} = \lim_{\alpha \to 1} \frac{d\eta_\alpha}{d\delta}$$

5. On-line Optimization

In this section, we show that given a sample path of a system run under a policy, we can estimate the potentials for that policy and the performance derivative in any given direction. Based on this, the research in on-line optimization goes two directions: First, with the potentials estimated, we can implement policy iteration. When the new policy is found, we run the system under the new policy for another sample path and update the policy again until some criterion is met. We shall refer to this procedure as the *potential-based on-line policy iteration*. Stochastic approximation techniques can also be used in this procedure to provide fast algorithms. Research in this direction is reported in Fang and Cao (submitted). Second, performance derivatives can be estimated on a sample path. The derivatives can then be used together with stochastic approximation techniques to search for the optimal performance. Algorithms and proofs of convergence and other results in this direction is reported in Marbach and Tsitsiklis (2001). This is in the same spirit of the

PA-based optimization (e.g., Chong and Ramadge, 1992; Fu, 1990; Ho and Cao, 1983; and Vazquez-Abad et al., 1998).

In Section 5.1, we discuss the sample-path-based estimation of potentials. In Section 5.2, we give an example to illustrate the advantages of the potential-based on-line policy iteration approach. In Section 5.3, we discuss the estimation of performance gradients and its application to performance optimization.

5.1. Estimating Performance Potentials

In this subsection, we present some sample-path-based estimation algorithms for potentials and realization factors (Cao, 1999; Cao and Wan, 1998).

First, from (13), we can choose a fixed L and use $g_L(i) = E[\sum_{l=0}^{L-1} f(X_l)|X_0 = i] - L\eta$ as an estimate of $g(i)$. Define $\varepsilon_i(x) = 1$, if $x = i$, and $\varepsilon_i(x) = 0$, otherwise. Given a long sample path $(X_0, X_1, \ldots, X_K)$, we can prove

$$g_L(i) = \lim_{K\to\infty} \left\{ \frac{\sum_{k=0}^{K-L+1} \left\{ \varepsilon_i(X_k) \left[\sum_{j=0}^{L-1} f(X_{k+j}) \right] \right\}}{\sum_{k=0}^{K-L+1} \varepsilon_i(X_k)} - \frac{L}{K} \sum_{k=0}^{K-1} f(X_k) \right\}, \qquad w.p.1 \qquad (38)$$

For realization factor, algorithms can be developed from (8). However, such algorithms require two sample paths, one starting with $X_0 = i$ and the other with $X'_0 = j$. Another algorithm overcomes this difficulty and is based on a finite sample path. On a Markov chain $\mathbf{X} = \{X_n, n \geq 0\}$ with $X_0 = i$, we define $L_i(j) = \min\{n : n \geq 0, X_n = j\}$; i.e., at $n = L_i(j)$, the Markov chain reaches state j for the first time. We have $E[L_i(j)|X_0 = i] < \infty$ (Çinlar, 1975), and from Cao and Chen (1997)

$$d(j,i) = E\left\{ \sum_{l=0}^{L_i(j)-1} [f(X_l) - \eta] | X_0 = i \right\} \qquad (39)$$

(39) relates $d(i,j)$ to a finite portion of the sample paths of $\mathbf{X}$. To develop an algorithm based on (39), we define $u_0 = 0$, and $u_{k+1} = \min\{n : n > u_k, X_n = i\}$, $k \geq 0$, where i is a fixed state. u_k, $k \geq 0$, are regenerative points. For any $j \neq i$, define $v_k(j) = \min\{n : u_{k+1} > n > u_k, X_n = j\}$ and $\chi_k(j) = 1$, if $\{u_{k+1} > n > u_k, X_n = j\} \neq \emptyset$; and $\chi_k(j) = 0$, otherwise. From (39), we have

$$d(i,j) = \lim_{K\to\infty} \frac{1}{\sum_{k=0}^{K-1} \chi_k(j)} \left\{ \left[\sum_{k=0}^{K-1} \{\chi_k(j)\} \sum_{n=v_k(j)}^{u_{k+1}-1} f(X_n) \right] - \left[\sum_{k=0}^{K-1} \chi_k(j)[u_{k+1} - v_k(j)] \right] \eta \right\} \qquad w.p.1 \qquad (40)$$

where η can be simply estimated by

$$\eta = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} f(X_n), \qquad w.p.1.$$

Other algorithms may be developed (see Section 6). With the potential estimates, optimization can be carried out either using the performance derivatives or policy iteration, as previously explained.

The above discussion applies to discounted MDPs as well. In particular, ignoring the constant term, the α-potential can be written as

$$g_\alpha(i) = E\left\{ \sum_{n=0}^{\infty} \alpha^n f(X_n) | X_0 = i \right\}, \qquad 0 < \alpha < 1 \tag{41}$$

This is the same as the α-potential defined in the standard potential theory Çinlar (1975). When $\alpha = 1$, the above expression goes to infinity. Thus, (13) can be viewed as the extension of the α-potential to the case of $\alpha = 1$.

5.2. Potential Based On-line Optimzation: An Example

We use an example to illustrate the advantages of the potential-based on-line approach.

Example 1: A manufacturing system consists of two machines and N pieces of works, which are circulating between the two machines, as shown in Figure 6. Each work piece has to undertake three consecutive operations at machine 1; thus machine 1 is illustrated by three circles in the figure. The service rates of these three operations are λ_1, λ_2, and λ_3, respectively. Machine 2 has only one operation with service rate λ_4. The system can be modeled as a Markov process with state denoted as (n, i), where n is the number of pieces in machine 1 and $i = 1, 2, 3$ denotes the operation that the piece at machine 1 is undertaking. A work piece, after the completion of service at machine 1, goes to machine 2 with probability $p^\alpha(n)$ and feeds back to machine 1 with probability $1 - p^\alpha(n)$. The subscript ''α'' represents an action with $\alpha \in \mathscr{A}$. For any $\alpha \in \mathscr{A}$ and $n = 1, 2, \ldots, N$,

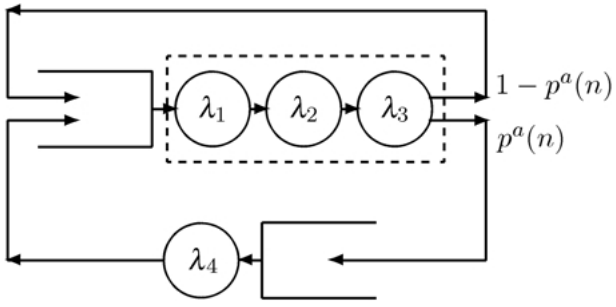

Figure 6. The example.

$p^\alpha(n) \in [0, 1]$. The performance to be optimized is the weighted sum of the two machines' throughput. (The cost function f does not depend on the actions.) Some transition probabilities of the Markov chain embedded at the operation completion times are (for $0 < n < N$):

$$\begin{aligned} p[(n,1),(n+1,1)] &= \frac{\lambda_4}{\lambda_1+\lambda_4} \\ p[(n,1),(n,2)] &= \frac{\lambda_1}{\lambda_1+\lambda_4}, \\ p^\alpha[(n,3),(n+1,3)] &= \frac{\lambda_4}{\lambda_3+\lambda_4}, \\ p^\alpha[(n,3),(n-1,1)] &= \frac{\lambda_3}{\lambda_3+\lambda_4} p^\alpha(n), \\ p^\alpha[(n,3),(n,1)] &= \frac{\lambda_3}{\lambda_3+\lambda_4}[1-p^\alpha(n)] \end{aligned}$$

Other nonzero transition probabilities have a similar form.

The transitions from states $(n, 1)$ and $(n, 2)$ do not depend on actions. For state $(n, 3)$, the inequality for policy iteration (see Lemma 1) is

$$\begin{aligned} &\frac{\lambda_4}{\lambda_3+\lambda_4} g^\alpha(n+1,3) + \frac{\lambda_3}{\lambda_3+\lambda_4} p^\alpha(n) g^\alpha(n-1,1) + \frac{\lambda_3}{\lambda_3+\lambda_4}[1-p^\alpha(n)] g^\alpha(n,1) \\ &\quad \le \frac{\lambda_4}{\lambda_3+\lambda_4} g^\alpha(n+1,3) + \frac{\lambda_3}{\lambda_3+\lambda_4} p^{\alpha'}(n) g^\alpha(n-1,1) + \frac{\lambda_3}{\lambda_3+\lambda_4}[1-p^{\alpha'}(n)] g^\alpha(n,1) \end{aligned}$$

for all $\alpha' \in \mathscr{A}$. This is equivalent to

$$[p^\alpha(n) - p^{\alpha'}(n)] g^\alpha(n-1,1) - [p^\alpha(n) - p^{\alpha'}(n)] g^\alpha(n,1) \le 0 \tag{42}$$

In (42), only the action-related probabilities are involved; the system parameters λ_1, λ_2, λ_3, and λ_4 do not appear.

In the above example, the service rates govern the evolution of the system, which runs autonomously. The control action can affect only some of the system transitions. The transition probabilities corresponding to the uncontrolled transitions are the same under all policies; they cancel each other in $(P' - P)g$ and hence do not appear in the final form. Since we can estimate g^αs on a sample path, we can implement policy iteration without knowing these transition probabilities by observing a sample path.

Many practical systems have the same feature as the above example. Indeed, for many such systems, control can be exercised only at a very limited region (e.g., admission control can be applied only at the access points of a high-speed communication network); the rest part of the systems simply evolves by its own nature. The dashed box in Figure 6 can also be viewed as a machine whose service time has an Erlangian distribution, in such cases the transitions between the three stages are not controllable. This type of service

distribution and the more general forms, such as the coxian distribution and the phase-type distributions, are very common in practical systems.

In summary, Example 1 and the above discussion illustrate that the potential-based on-line approach has the following advantages.

1. Given a sample path, policy iteration can be implemented without knowing the whole transition matrix; only those items related to control actions (e.g., $p^{\alpha}(n)$ in (42)) have to be known. Matrix inversion is not required.

2. The approach saves memory spaces required for implementing MDPs. Generally speaking, only the M potentials, not the $M \times M$ transition matrix, have to be stored. This can be further reduced when there are some states which cannot be reached by controlled states. As shown in (42), in Example 1 only $g^{\alpha}(n, 1)$, $n = 0, 1, \ldots, N$, have to be estimated and stored; $g^{\alpha}(n, 2)$ and $g^{\alpha}(n, 3)$, $n = 0, 1, \ldots, N$, do not even need to be estimated.

3. Different from the standard computational approach where all the potentials are obtained altogether by a matrix inversion, in the sample-path-based approach potentials can be estimated one by one. This feature brings in a number of possible applications.

    a. The computational efforts and memory spaces of each iteration may be further reduced at the cost of the convergence rate. The idea is, if the state space is too large, at each iteration we can estimate the potentials of only a subset of the state space and update the actions for the states in this subset. For instance, in Example 1 we may set $0 = n_0 < n_1 < n_2 < \cdots < n_{k-1} < n_k = M$. Then in the i-th iteration, $i = 1, 2, \ldots, k$, we may estimate $g^{\alpha}(n, 1)$ only for $n_{i-1} - 1 \leq n \leq n_i$, $i = 1, 2, \ldots, k$. Of course, it may need more iterations to reach the optimal policy; however, at each iteration the computation and the memory requirements may be reduced to fit the capacity of the available computing equipments. This feature may be important for on-line optimization using special designed hardwares which may have limited capacities. In high speed communication networks, the effect of a slow convergence rate in terms of iterations may be compensated by the fast speed in system evolution.

    b. For many practical systems, we may have some a priori knowledge about which states are more important than others. Then we can estimate only the potentials for the states that are needed for updating the actions on these important states. This may reduce the computation and memory at the cost of the best performance achieved.

5.3. Algorithms for Performance Gradients

One drawback of the above approach is that the number of states may be too large. However, we may develop on-line algorithms to estimate $d\eta/d\delta$ for any given $Q = P' - P$

on a single sample path of a Markov chain with transition probability P. One of such algorithms is developed in Cao and Wan (1998). Since the second term in (38) is a constant, we can remove it and obtain

$$g_L(i) = \lim_{K\to\infty} \frac{\sum_{k=0}^{K-L+1}\left\{\varepsilon_i(X_k)[\sum_{j=0}^{L-1} f(X_{k+j})]\right\}}{\sum_{k=0}^{K-L+1}\varepsilon_i(X_k)}, \qquad w.p.1$$

Based on this, it was proved in Cao and Wan (1998) that

$$\begin{aligned}
\frac{\partial\eta}{\partial\delta} &\approx \pi Q g_L \\
&= \lim_{K\to\infty}\frac{1}{K-L+1}\sum_i\sum_j\left\{\sum_{k=0}^{K-L}\varepsilon^i(X_k)\varepsilon^j(X_{k+1})\frac{q(i,j)}{p(i,j)}\left[\sum_{l=0}^{L-1}f(X_{k+l+1})\right]\right\} \\
&= \lim_{K\to\infty}\frac{1}{K-L+1}\left\{\sum_{k=0}^{K-L}\sum_i\sum_j\left\{\varepsilon^i(X_k)\varepsilon^j(X_{k+1})\frac{q(i,j)}{p(i,j)}\right\}\left[\sum_{l=0}^{L-1}f(X_{k+l+1})\right]\right\} \\
&= \lim_{K\to\infty}\frac{1}{K-L+1}\left\{\sum_{k=0}^{K-L}\left\{\frac{q(X_k,X_{k+1})}{p(X_k,X_{k+1})}\right\}\left[\sum_{l=0}^{L-1}f(X_{k+l+1})\right]\right\}, \qquad w.p.1 \qquad (43)
\end{aligned}$$

Algorithms can be developed based on (43) (see Cao and Wan, 1998). Equation (43) has a significant meaning: we can estimate the performance sensitivity along any direction on a single sample path without estimating every component of g. Thus, it may be applied to systems with large state spaces.

In general, we assume that the system transition probability matrix contains a parameter θ; i.e., $P = P(\theta)$. The system performance depends on θ: $\eta = \eta(\theta)$. When θ changes to $\theta + \Delta\theta$, $P(\theta)$ changes to $P(\theta) + dP/d\theta\Delta\theta$. Therefore, we can set $Q = dP/d\theta$. Along the direction of $Q\delta$ applying (43) yields the performance derivative $d\eta/d\theta$.

Equation (43) was extended to the discounted cost problems and partially observable MDP in Baxter and Bartlett (2001) and Baxter et al. (2001). Marbach and Tsitsiklis (2001) applied stochastic approximation methods to the gradient estimation and developed recursive algorithms that converge to a local optimal point with the performance derivative being zero.

6. Reinforcement Learning

As shown in the last section, potentials can be estimated on a single sample path. More efficient algorithms can be developed with stochastic approximation methods. For example, from (13), we have

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

Given a sample path $\{X_0, \ldots, X_n, X_{n+1}, \ldots,\}$, at the n-th transition, we have

$$g(X_n) = E\left\{\sum_{l=0}^{\infty}[f(X_{n+l}) - \eta]|X_n\right\} = [f(X_n) - \eta] + E[g(X_{n+1})|X_n] \tag{44}$$

Now suppose we observe a transition from state X_n to X_{n+1}, then $[f(X_n) - \eta] + g(X_{n+1})$ may be a more accurate estimate than $g(X_n)$ because it reflects the information at this transition. Define the *temporal difference* (TD) as

$$d_n = \left[f(X_n) - \eta + g(X_{n+1}) - g(X_n)\right]$$

which may ''reflect'' the stochastic error observed at transition n. Based on (44) and with the stochastic approximation approach, some recursive on-line algorithms (called TD(λ) algorithms) for estimating $g(i)$ can be developed. Tsitsiklis and Van Roy (1999) presents TD(λ) algorithms with linearly parameterized function approximation. For TD(λ) algorithms for discounted or total cost problems, see Bertsekas and Tsitsiklis (1996), Sutton (1988) and Sutton and Barto (1998).

When the system structure, i.e., the transition probabilities, $p(i,j)s$, are completely unknown, we cannot apply policy iteration directly by using the potentials. (As shown in Example 1, we need to know at least the transition probabilities related to the actions, i.e., $p^{\alpha}(n)$ to implement policy iteration with estimated potentials.) In this case, we can estimate the Q-factor defined as (Bertsekas and Tsitsiklis, 1996 and Sutton and Barto, 1998)

$$Q(i,\alpha) = \left\{\sum_{j=1}^{M} p^{\alpha}(i,j)g(j)\right\} + f^{\alpha}(i) - \eta \tag{45}$$

for every state-action pair (i,α). From Theorem 1, we can choose $\alpha = \arg\{\min_{\alpha' \in \mathscr{A}}[Q(i,\alpha')]\}$ as the action at state i in the policy iteration. However, there is one problem for this approach: it is impossible to estimate $Q(i,\alpha')$ on the current sample path with $\alpha \neq \alpha'$ being the action taken at state i, because the pair (i,α') does not appear on the sample path. One way to obtain the estimation of $Q(i,\alpha')$, $\alpha' \neq \alpha$, is to apply the importance sampling technique. This requires to know the ratios of $p^{\alpha'}(i,j)$ and $p^{\alpha}(i,j)$. In Example 1, this ratio can be obtained if $p^{\alpha'}(n)$ and $p^{\alpha}(n)$ are known. This shows that policy iteration based on Q-factors is almost the same as that based on potentials.

The value iteration of Q-factor, however, leads to the optimal Q-factor and can be implemented when the system structure is completely unknown. This is briefly explained as follows. From the definition (45), we have

$$\eta + Q(i,\alpha) = \left\{ \sum_{j=1}^{M} p^{\alpha}(i,j) g(j) \right\} + f^{\alpha}(i) \tag{46}$$

Taking minimum on both sides, we get

$$\eta + \min_{\alpha \in \mathscr{A}} Q(i,\alpha) = \min_{\alpha \in \mathscr{A}} \left\{ \sum_{j=1}^{M} p^{\alpha}(i,j) g(j) + f^{\alpha}(i) \right\}$$

Comparing with the Bellman equation

$$\eta + g(i) = \min_{\alpha \in \mathscr{A}} \left\{ \sum_{j=1}^{M} p^{\alpha}(i,j) g(j) + f^{\alpha}(i) \right\}$$

we conclude that at the optimal policy $g(i) = \min_{\alpha \in \mathscr{A}} Q(i,\alpha)$. Substituting this into (46), we have at the optimal policy

$$\eta + Q(i,\alpha) = \left\{ \sum_{j=1}^{M} p^{\alpha}(i,j) \left[\min_{\beta \in \mathscr{A}} Q(j,\beta) \right] \right\} + f^{\alpha}(i) \tag{47}$$

This is the Bellman equation for Q-factors. Based on (47), the Robbins-Monroe stochastic approximation method leads to

$$Q(i,\alpha) := (1-\gamma) Q(i,\alpha) + \gamma [\min_{\beta \in \mathscr{A}} Q(j,\beta)] + f^{\alpha}(i) - \eta], \qquad 0 < \gamma < 1 \tag{48}$$

Applying this to a sample path leads to the Q-learning algorithm (Abounadi et al., 1998; Singh, 1994): when a system jumps from state i to j, the Q-factor is updated according to (48). It can be proved that (48) converges to the optimal $Q^*(i,\alpha)$ and $\alpha^* = \arg\{\min_{\alpha' \in \mathscr{A}} [Q^*(i,\alpha')]\}$ is the optimal action at state i. For Q-learning algorithms with discounted or total cost problems, see Bertsekas and Tsitsiklis (1996), Singh (1994) and Sutton and Barto (1998).

Compared with the approach based on potentials, Q-learning can be applied to systems where the structure is completely unknown. However, from (48) it is clear that Q-learning requires the sample path visit every state-action pair. In addition, the number of Q-factors increases to $M \times K$ (where K is the number of possible actions at each state).

In addition to the above approaches, *Neuro-Dynamic programming* is proposed to overcome the difficulty of the so-called ''curse of dimensionality''. Roughly speaking, in neuro-dynamic programming, we try to approximate the potential function $g(i)$ by $g(i,r)$, with a continuous parameter r. This generally involves two steps:

1. develop an approximation architecture, e.g., a neuro-network, to represent $g(i,r)$,

2. find a training algorithm for updating the parameter vector **r**, based on the information observed on a sample path.

After training, the parameter r reaches a proper value. The neuro-network will output an approximate value of $g(i)$ for an input integer i. For details and successful examples, see Bertsekas and Tsitsiklis (1996). Another approach based on approximation is the ''actor-critic'' algorithms, see Barto et al. (1983), Konda and Borkar (1999) and Konda and Tsitsiklis (2001).

The discussion about the relations among PA, MDP, and RL brings up a new direction: the development of efficient learning algorithms for performance derivatives using the stochastic approximation methods. The derivatives can be used in optimization directly or in the gradient-based policy iteration.

7. Discussions and Conclusions

The basic principle for PA is that the effect of a change in a system parameter on its performance can be decomposed into the effects of many single small perturbations on the performance. The effect of a single perturbation can be measured by realization factor. This principle has been successfully applied to obtain the performance derivatives for queueing networks as well as Markov processes. The results were extended to sensitivities with respect to discrete parameters, and the difference of the performance of the Markov chains under two different policies is found. With the discrete performance sensitivity formula, the standard policy iteration optimization algorithm can be very easily derived. Compared with the continuous sensitivity formula, it is clear that at each step the policy iteration algorithm simply chooses the ''steepest'' direction (with the largest absolute value of the directional derivative) to go for its next policy.

The concept of performance potentials plays a significant role in both sensitivity analysis of Markov processes and MDPs. Just as the potential energy in physics, only the relative values (i.e., the differences) of the performance potentials at different states are meaningful, i.e., the potentials are determined only up to an additive constant. Realization factor is defined as the difference between the potentials at two states i and j, $g(i) - g(j)$, which measures the effect of a jump from state j to state i on the long-run (average- or discounted-cost) performance. This offers an intuitive explanation for performance sensitivity and MDPs (in view of discrete sensitivity). The performance potential at state i can be approximated by the mean sum of the performance at the first L states on a sample path starting from state i, and hence it can be estimated online. A number of other single sample path-based estimation algorithms for potentials have been derived.

With the potentials estimated, performance derivatives (i.e., PA) can be obtained and policy iteration (i.e., MDPs) can be implemented based on a single sample path. The potential-based approach is practically important because it can be applied online to real engineering systems; in many cases, it does not require the system parameters to be completely known. There are two ways to achieve the optimal performance with this approach: by perturbation analysis using performance derivatives, or by policy iteration, both can be implemented online. Stochastic approximation methods can be used in these

two cases to improve the convergence speeds and to reduce stochastic errors. The advantages of the potential-based online approach is discussed through an example. The performance gradient can be obtained on a sample path without estimating each component of the potential. Thus, the gradient-based approach can be applied to parameterized systems with large state spaces. Furthermore, Reinforcement learning, TD(λ), neuro-dynamic programming, etc, are efficient ways of estimating the performance potentials and related quantities based on sample paths. In particular, Q-learning can be used when the system structure is unknown.

Inspired by the potential-based sensitivity view of MDPs and the single sample path-based approach, a number of new research topics emerge. First, we observe that policy iteration can be implemented by using the performance derivatives. This leads to a new approach, the gradient-based policy iteration, to MDPs with correlated actions at different states. This gradient-based policy iteration can be applied to problems such as MDPs with hidden state components and the distributed control of MDPs. Many problems remain to be solved in this direction. The other topics include the time aggregation of MDPs (Cao et al., 2002), which was first used in Zang and Ho (1991) for perturbation analysis. When the number of controllable states is small, this approach may save computations, storage spaces, and/or the number of transitions.

Finally, with the knowledge reviewed and discussed in this paper, the relations among different areas and methodlogies illustrated in Figure 1 can be modified and explained in more details as in Figure 7.

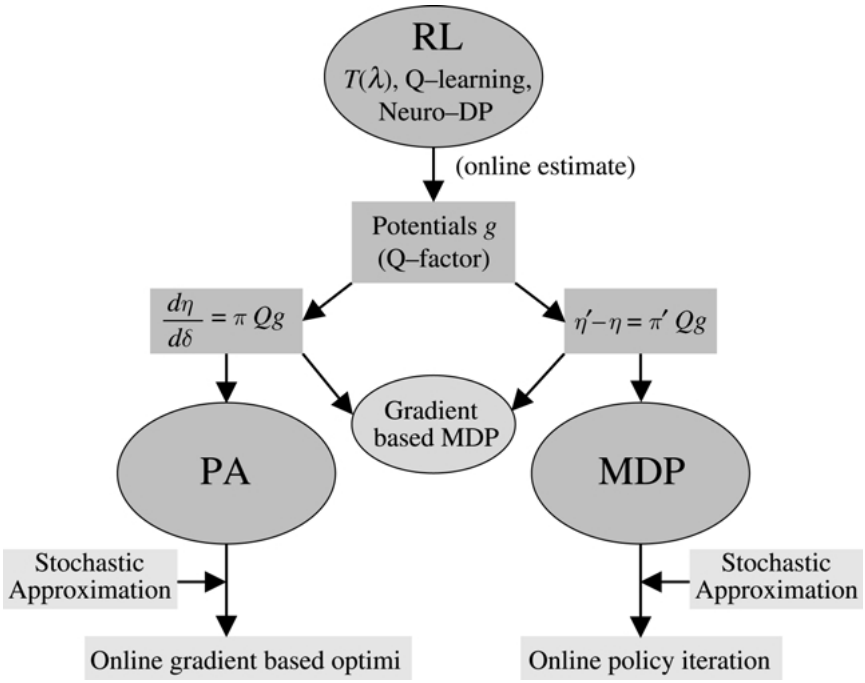

Figure 7. Potentials, performance sensitivities, PA, MDP, and RL.

References

Abounadi, J., Bertsekas, D., and Borkar, V. Learning algorithms for Markov decision processes with average cost, Report LIDS-P-2434, Lab. for Info. and Decision Systems, October 1998; to appear in *SIAM J. on Control and Optimization*.

Altman, E. 1999. *Constrained Markov Decision Processes*. Chapman Hall/CRC.

Barto, A., Sutton, R., and Anderson, C. 1983. Neuron-like elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics* 13: 835–846.

Baxter, J., and Bartlett, P. L. 2001. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research* 15: 319–350.

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

Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control*. Vols. I, II, Athena Scientific, Belmont, Massachusetts.

Berman, A., and Plemmons, R. J. 1994. Nonnegative matrices in the mathematical sciences. *SIAM*, Philadelphia.

Bertsekas, D. P., and Tsitsiklis, T. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts.

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

Cao, X. R. 1998. The relation among potentials, perturbation analysis, Markov decision processes, and other topics. *Journal of Discrete Event Dynamic Systems* 8: 71–87.

Cao, X. R. 1999. Single sample path-based optimization of markov chains. *Journal of Optimization: Theory and Application* 100: 527–548.

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

Cao, X. R., and Chen, H. F. 1997. Potentials, perturbation realization, and sensitivity analysis of Markov processes. *IEEE Transactions on AC* 42: 1382–1393.

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.

Cao, X. R., and Fang, H. T. Gradient-based policy iteration: an example. To appear in *2002 IEEE Conference on Decision and Control*.

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

Cassandras, C. G., and Lafortune, S. 1999. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers.

Chong, E. K. P., and Ramadge, P. J. 1992. Convergence of recursive optimization algorithms using infinitesimal perturbation analysis estimates. *Journal of Discrete Event Dynamic Systems* 1: 339–372.

Çinlar, E. 1975. *Introduction to Stochastic Processes*. Prentice Hall, Englewood cliffs, NJ.

Fang, H. T., and Cao, X. R. Single sample path-based recursive algorithms for Markov decision processes. *IEEE Trans. on Automatic Control*, submitted.

Feinberg, E. A., and Adam Shwartz (ed.) 2002. *Handbook of Markov Decision Processes*. Kluwer, 2002.

Fu, M. C. 1990. Convergence of a stochastic approximation algorithm for the GI/G/1 queue using infinitesimal perturbation analysis. *Journal of Optimization Theory and Applications* 65: 149–160.

Fu, M. C. and Hu, J. Q. 1997. *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*. Kluwer Academic Publishers, Boston.

Glasserman, P. 1991. *Gradient Estimation Via Perturbation Analysis*. Kluwer Academic Publishers, Boston.

Glynn, P. W., and Meyn, S. P. 1996. A Lyapunov bound for solutions of Poisson's equation. *Ann. Probab.* 24: 916–931.

Gong, W. B., and Ho, Y. C. 1987. Smoothed perturbation analysis of discrete event systems. *IEEE Transactions on Control Systems Technology* 32: 858–866.

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

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

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101.

Kemeny, J. G., and Snell, J. L. 1960. *Finite Markov Chains*. Van Nostrand, New York.

Konda, V. R., and Borkar, V. S. 1990. Actor-critic like learning algorithms for Markov decision processes. *SIAM Journal on Control and Optimization* 38: 94–123.

Konda, V. R., and Tsitsiklis, J. N. 2001. Actor-critic Algorithms. Submitted to *SIAM Journal on Control and Optimisation*, February.

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

Meyn, S. P. 1997. The policy improvement algorithm for Markov decision processes with general state space. *IEEE Transactions on Automatic Control* 42: 1663–1680.

Meyn, S. P., and Tweedie, R. L. 1993. *Markov Chains and Stochastic Stability.* Springer-Verlag, London.

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

Singh, S. P. 1994. Reinforcement learning algorithms for average-payoff Markovain decision processes. *Proceedings of the Twelfth National Conference on Artificial Intelligence* 202–207.

Smart, W. D., and Kaelbling, L. P. 2000. Practical reinforcement learning in continuous spaces. *Proceedings of the Seventeenth International Conference on Machine Learning*.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3: 835–846.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Tsitsiklis, J. N., and Van Roy, B. 1999. Average cost temporal-difference learning. *Automatica* 35: 1799–1808.

Vazquez-Abad, F. J., Cassandras, C. G., and Julka, V. 1998. Centralized and decentralized asynchronous optimization of stochastic discret event systems. *IEEE Transactions on Automatic Control* 43: 631–655.

Zhang, B., and Ho, Y. C. 1991. Performance gradient estimation for very large finite Markov chains. *IEEE Transactions on Automatic Control* 36: 1218–1227.