DISCRETE EVENT DYNAMICAL SYSTEMS

In recent decades, many modern, large-scale, human-made systems (e.g., flexible manufacturing systems, computer and communication networks, air and highway traffic networks, and the military C^3I/logistic systems) have been emerging. These systems are called *discrete event systems* because of the discrete nature of the events. Research indicates that these human-made systems possess many properties that are similar to those of natural physical systems. In particular, the evolution of these human-made systems demonstrates some dynamic features; exploring these dynamic properties may lead to new perspectives concerning the behavior of discrete event systems. The increasing need for analyzing, controlling, and optimizing discrete event systems has initiated a new research area, the dynamics of discrete event systems. To emphasize their dynamic nature, these systems are often referred to as discrete event dynamic systems (DEDS) in the literature (1).

This article reviews the fundamental theories and applications of DEDS. Since the dynamic behavior is closely related to time, we shall not discuss untimed models such as the automata-based model (2); these models are mainly used to study the logical behavior of a discrete event system.

In 1988, the report of the panel of the IEEE Control Systems Society noted, “Discrete event dynamic systems exist in many technological applications, but there are no models of discrete event systems that are mathematically as concise or computationally as feasible as are differential equations for continuous variable dynamical systems. There is no agreement as to which is the best model, particularly for the purpose of control” (3). This statement is still true today. However, after the hard work of many researchers in the recent years, there are some relatively mature theories and many successful application examples.

Most frequently used models for analyzing DEDS are queueing systems and Petri nets. Queueing models are usually used to analyze the performance (in most cases, steady state performance) of DEDS, and Petri nets provide a graphical illustration of the evolution of the system behavior and are particularly useful in analyzing behaviors comprising concurrency, synchronization, and resource sharing (4). Other models for DEDS include the more general but less structural models such as Markov processes and generalized semi-Markov processes (GSMP), and the max-plus algebra that is particularly suitable for modeling DEDS with deterministic event lifetimes that exhibit a periodic behavior.

One main theory that employs a dynamic point of view to study system behavior is the perturbation analysis (PA). The objective of PA is to obtain performance sensitivities with respect to system parameters by analyzing a single sample path of a discrete event system (5–9). The sample path, which describes how the DEDS evolves, can be obtained by observing the operation of a real system or by simulation. The technique is in the same spirit of the linearization of nonlinear continuous variable dynamic systems (6).

The sample path based approach of PA motivates the research of on-line performance optimization of DEDS. Recent study shows that PA of discrete parameters (parameters that jump among discrete values) is closely related to the Markov decision problem (MDP) in optimization. The PA-based on-line optimization technique has been successfully applied to a number of practical engineering problems.

The following section briefly reviews some basic DEDS models. The next section introduces PA in some details. The final section 4 discusses the application of PA in on-line optimization and points out its relations with the Markov decision problem.

DEDS MODELS

Queueing Systems

The simplest queueing system is the $M/M/1$ queue, where customers arrive at a single server according to a Poisson process with rate λ and the service time for each customer is exponentially distributed with mean $1/\mu$, $\mu < \lambda$. The steady state probability of n, the number of customers in the queue, is

$$p(n) = \rho^n(1-\rho) \qquad \rho = \frac{\lambda}{\mu}, n = 0, 1, \ldots$$

From this, the average number of customers in the queue is

$$\bar{n} = \frac{\rho}{1-\rho}$$

and the average time that a customer stays in the queue is

$$T = \frac{1}{\mu - \lambda}$$

The more general model is a queueing network that consists of a number of service stations. Customers in a network may belong to different classes, meaning that they may have different routing mechanisms and different service time distributions. A queueing network may belong to one of three types: open, closed, or mixed. In an open network, customers arrive at the network from outside and eventually leave the network; in a closed network, customers circulate among stations and no customer arrives or leaves the network; A mixed network is open for some classes of customers and closed for others.

An open Jackson network consists of M single-server stations and N single-class customers. Each server has an buffer with an infinite capacity and the service discipline is first-come-first-served. Customers arrive at server i according to a Poisson process with (external) rate $\lambda_{0,i}$, $i = 1, 2, \cdots, M$. After receiving the service at server i, a customer enters server j with probability $q_{i,j}$ and leaves the network with probability $q_{i,0}$. We have $\sum_{j=0}^{M} q_{i,j} = 1$, $i = 1, 2, \cdots, M$. The service time of server i is exponentially distributed with mean $\bar{s}_i = 1/\mu_i$, $i = 1, 2, \cdots, M$.

The system state is $\mathbf{n} = (n_1, n_2, \cdots, n_M)$, where n_i is the number of customers in server i. Let λ_i be the arrival rate of the customers to server i. Then

$$\lambda_i = \lambda_{0,i} + \sum_{j=1}^{M} \lambda_j q_{j,i} \qquad i = 1, 2, \ldots, M$$

It is known that the steady state distribution is

$$p(\mathbf{n}) = p(n_1, n_2, \ldots, n_M) = \sum_{k=1}^{M} p(n_k)$$

with

$$p(n_k) = (1-\rho_k)\rho_k^{n_k} \qquad \rho_k = \frac{\lambda_k}{\mu_k} \qquad k = 1, 2, \ldots, M$$

A load-independent closed Jackson (Gordon-Newell) network is similar to the open Jackson network described above, except that there are N customers circulating among servers according to the routing probabilities $q_{i,j}$, $\sum_{k=1}^{M} q_{i,k} = 1$, $i = 1, 2, \cdots, M$. We have $\sum_{k=1}^{M} n_k = N$. We consider a more general case: the service requirement of each customer is exponential with a mean =1; the service rates, however, depend on the number of customers in the server. Let μ_{i,n_i} be the service rate of server i when there are n_i customers in the server, $0 \leq \mu_{i,n_i} < \infty$, $n_i = 1, 2, \cdots, N$, $i = 1, 2, \cdots, M$. We call this a *load-dependent network*. In a load-independent network, $\mu_{i,n_i} \equiv \mu_i$ for all n_i, $i = 1, 2, \cdots, M$.

The state of such a network is $\mathbf{n} = (n_1, n_2, \cdots, n_M)$. We use $\mathbf{n}_{i,j} = (n_1, \cdots, n_i - 1, \cdots, n_j + 1, \cdots, n_M)$, $n_i > 0$, to denote a neighboring state of $\mathbf{n}$. Let

$$\epsilon(n_k) = \begin{cases} 1 & \text{if } n_k > 0 \\ 0 & \text{if } n_k = 0 \end{cases}$$

and let

$$\mu(\mathbf{n}) = \sum_{k=1}^{M} \epsilon(n_k)\mu_{k,n_k}$$

Then the flow balance equation for the steady state probability $p(\mathbf{n})$ is

$$\mu(\mathbf{n})p(\mathbf{n}) = \sum_{i=1}^{M}\sum_{j=1}^{M} \epsilon(n_j)\mu_{i,n_i+1} q_{i,j} p(\mathbf{n}_{j,i})$$

Let $y_i > 0$, $i = 1, 2, \cdots, M$, be the visit ratio to server i, that is, a solution (within a multiplicative constant) to the equation

$$y_i = \sum_{j=1}^{M} q_{j,i} y_j \qquad j = 1, 2, \ldots, M$$

Let $A_i(0) = 1$, $i = 1, 2 \cdots, M$, and

$$A_i(k) = \prod_{j=1}^{k} \mu_{i,j} \qquad i = 1, 2, \ldots, M$$

and for every $n = 1, 2, \cdots, N$ and $M = 1, 2, \cdots, M$, let

$$G_m(n) = \sum_{n_1+\cdots+n_m=n} \prod_{i=1}^{m} \frac{y^{n_i}}{A_i(n_i)}$$

Then we have

$$p(\mathbf{n}) = \frac{1}{G_M(N)} \prod_{i=1}^{M} \frac{y_i^{n_i}}{A_i(n_i)}$$

This equation is often referred to as a *product-form* solution.

For load-independent networks, $\mu_{i,n_i} \equiv \mu_i$, $i = 1, 2, \cdots, M$. The product-form solution becomes

$$G_m(n) = \sum_{n_1+\cdots+n_m=n} \prod_{i=1}^{m} x_i^{n_i}$$

and

$$p(\mathbf{n}) = \frac{1}{G_M(N)} \prod_{i=1}^{M} x_i^{n_i} \tag{1}$$

where $x_i = y_i/\mu_i = y_i\bar{s}_i$, $i = 1, 2, \cdots, M$.

There are a number of numerical methods for calculating $p(\mathbf{n})$ and the steady state performance, among them are the convolution algorithm and the mean value analysis (7); in addition, analytical expressions exist for the normalizing constant $G_M(N)$ (10). For more about queueing theory, see, for example, Refs. 11 and 12.

One typical example of the queueing model is the resource sharing problem. Consider the case where M resources are shared by N users and each resource can be held by only one user at any time. Every time a user grasps resource i, it holds the resource for a random time with $\bar{s}_i$, A user, after the completion of its usage of resource i, requests the hold of resource j with a probability $q_{i,j}$. This problem can be modeled exactly as a closed queueing network with N customers and M servers. This model can be successfully used in analyzing the performance of packet switches, where the users are the head-of-line packets and the resources are the channels, and the performance of data-base systems, where the users are programs and the resources are data records.

Petri Nets

Many DEDS consist of components [e.g., central processing units (CPUs), disks, memories, and peripheral devices in computer systems; and machines, pallets, tools, and control units in manufacturing systems] that are shared by many users and exhibit concurrency. This feature makes Petri nets a suitable model for DEDS.

In a graphical representation, the structure of a Petri net is defined by three sets: a set of *places* $P = \{p_1, p_2, \cdots, p_n\}$, a set of *transitions* $T = \{t_1, t_2, \cdots, t_m\}$, and a set of arcs. A place is represented by a circle and a transition by a bar. An arc is represented by an arrow from a transition to a place or a place to a transition. A place is an input (output) to a transition if an arc exists from the place (transition) to the transition (place).

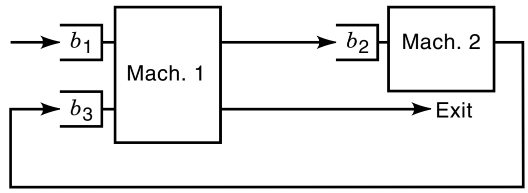

Figure 1. A reentrant line.

The dynamic feature of a Petri net is represented by tokens, which are assigned to the places. Tokens move from place to place during the execution of a Petri net. Tokens are drawn as small dots inside the circle representing the places. The *marking* of a Petri net is a vector $M = (m_1, m_2, \cdots, m_n)$, where m_i is the number of tokens in place p_i, $i = 1, 2, \cdots, n$. A marking corresponds to a state of the system. A Petri net executes by firing transitions. A transition is enabled if its every input place contains at least one token. When a transition is enabled, it may fire immediately or after a firing delay, which can be a random number. The firing delay is used to model the service times. When a transition fires, one token is removed from each input place and one token added to each output place. Thus, the number of tokens in a place and in the system my change during the execution. In addition to the arcs described above, another type of arc, called the *inhibitor arc,* is often used to model the priority among services. An inhibitor arc is drawn from a place to a transition, with a small circle at its end. When an inhibitor arc is used, if there is at least one token in the place, the transition cannot fire.

As an example, we consider the reentrant line (13) shown in Fig. 1. The system consists of two machines and three buffers. Work pieces arrive at buffer b_1 with rate λ and get service from machine 1 with rate μ_1; after the service in b_1, the piece moves to buffer b_2 and gets service at machine 2 with rate μ_2; after the service in b_2, the piece reenters machine 1 at buffer b_3 and receives service with rate μ_3. Machine 1 can serve one piece at a time, and pieces in b_3 have a nonpreemptive higher priority than those in b_1.

The Petri net model for the system is shown in Fig. 2. In the figure, places b_i, $i = 1, 2, 3$, represent the buffers, and transitions p_i, $i = 1, 2, 3$, represent the service processes of the pieces in the three buffers. If there is a token in places m_i, $i = 1, 2$, then machine i is available; if there is a token in s_i, $i = 1, 3$, then the work piece in buffer b_i is under service. It is clear that machine 1 is shared by the pieces in both b_1 and b_3, and the inhibitor arc from b_3 to p_4 models the priority. (For more about Petri net theory and applications, see Refs. 4 and 14–16).

The state process of a queueing system or a Petri net can be modeled as a Markov process, or more generally, as a *generalized semi-Markov process.* In this sense, both Markov processes and GSMPs are more general than queueing systems and Petri nets; however, these general models do not enjoy the structural property that queueing systems and Petri nets possess. In fact, the GSMP model is a formal description of the evolution mechanism of a queueing system. Readers are referred to Refs. 5 and 8 for a discussion of GSMP.

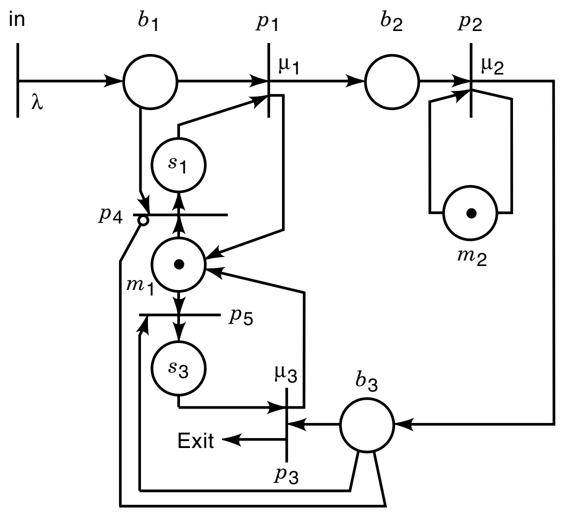

Figure 2. The Petri net model for the reentrant line.

The Max-Plus Algebra

With the max-plus algebra proposed in Ref. 17, many DEDS can be modeled as linear systems. In the max-plus algebra, we define the operation "multiplication" on two real numbers a and b as $a \otimes b = a + b$ and the "addition" as $a \oplus b = \max\{a, b\}$. It is easy to verify that these two operations indeed define an algebra. We give a simple example to illustrate how the max-plus algebra can be used to analyze the periodic behavior of a DEDS. Consider a single queue with deterministic interarrival time a and service time s. Let a_k and d_k be the arrival and departure times of the kth customer, respectively, $k = 1, 2, \cdots$, and $a_1 > 0$, $d_0 = 0$. Then for $k = 1, 2, \cdots$ we have

$$\begin{aligned} a_{k+1} &= a + a_k \\ d_k &= \max\{a_k + s, d_{k-1} + s\} \end{aligned}$$

In the max-plus algebra, this can be written as a linear equation

$$x_{k+1} = Ax_k \qquad k = 1, 2, \ldots$$

where

$$x_k = \begin{bmatrix} a_k \\ d_{k-1} \end{bmatrix} \qquad A = \begin{bmatrix} a & -H \\ d & d \end{bmatrix}$$

with H being a large positive number. Thus,

$$x_{k+1} = A^{\otimes k} x_k \tag{2}$$

where $A^{\otimes k} = A^{\otimes(k-1)} \otimes A$, $k > 1$, and $A^{\otimes 1} = A$.

An interesting property of a matrix under the max-plus algebra is the periodic property. This can be illustrated by an example. Consider

$$M = \begin{bmatrix} 1 & 5 \\ 3 & 2 \end{bmatrix} = 4 \otimes \begin{bmatrix} -3 & 1 \\ -1 & -2 \end{bmatrix} = 4 \otimes K$$

We can prove that for all $i \geq 1$

$$K^{\otimes(2i)} = \begin{bmatrix} 0 & -1 \\ -3 & 0 \end{bmatrix}$$

$$K^{\otimes(2i+1)} = \begin{bmatrix} -2 & -1 \\ -1 & -2 \end{bmatrix}$$

M is said to be of *order 2 periodic.* Cohen et al. (17) proved that all matrices possess such periodicity. Therefore, Eq. (2) can be used to study the periodic behavior of a DEDS. For more discussions, see Ref. 18.

PERTURBATION ANALYSIS

Perturbation analysis of DEDS is a multidisciplinary research area developed since early 1980s, with the initial work of Ho et al. (19). PA provides the sensitivities of performance measures with respect to system parameters by analyzing a single sample path of a DEDS. This area is promising because of its practical usefulness. First, compared with the standard procedure, which uses the difference of performance measures with two slightly different values for every parameter, this technique saves a great amount of computation in simulation, because PA algorithms can provide the derivatives with respect to all the parameters by using only a single simulation run. In addition, PA estimates are more accurate than those obtained from finite differences because the latter may encounter numerical problems caused by dividing two small numbers. Second, PA can be applied to on-line performance optimization of real-world systems by observing a sample path of an existing system; for these systems, changing the values of their parameters may not be feasible.

Cao (20) observed that the simple PA algorithms based on a single sample path, called *infinitesimal perturbation analysis* (IPA), in fact yield *sample derivatives* of the performance; although these sample derivatives are unbiased or strongly consistent for many systems, this is not the case for many others. This insight has set up two fundamental research directions: to establish IPA theory, including the proof of convergence of IPA algorithms, and to develop new algorithms for systems where IPA does not work well. After the hard work of many researchers in more than one decade, the theory for IPA is relatively mature, and many results have been obtained for problems where IPA does not provide accurate estimates.

Infinitesimal Perturbation Analysis

Let θ be a parameter of a stochastic discrete event system; the underlying probability space is denoted as $(\Omega, \mathscr{F}, \mathscr{P})$. Let $\xi = \xi(\omega)$, $\omega \in \Omega$, be a random vector that determines all the randomness of the system. For example, for a closed queueing network, ξ may include all the uniformly distributed random variables on $[0, 1)$ that determine the customer's service times and their transition destinations (say, in a simulation). Thus, a sample path of a DEDS depends on θ and ξ; such a sample path is denoted as (θ, ξ).

Let $T_0 = 0, T_1, \cdots, T_l, \cdots$ be the sequence of the state transition instants. We consider a sample path of the system in a finite period $[0, T_L)$. The performance measured on this sample path is denoted as $\eta_L(\theta, \xi)$. Let

$$\bar{\eta}_L(\theta) = E[\eta_L(\theta, \xi)] \tag{3}$$

and

$$\eta(\theta) = \lim_{L\to\infty} \eta_L(\theta, \xi) \qquad w.p.1. \tag{4}$$

where E denotes the expectation with respect to the probability measure $\mathscr{P}$. We assume that both the mean and limit in Eq. (3) and Eq. (4) exist. Thus $\eta_L(\theta, \xi)$ is an unbiased estimate of $\overline{\eta}_L(\theta)$ and an strongly consistent estimate of $\eta(\theta)$, respectively.

The goal of perturbation analysis is to obtain the performance derivative with respect to θ by analyzing a single sample path (θ, ξ). That is, we want to derive a quantity based on a sample path (θ, ξ) and use it as an estimate of $\partial\overline{\eta}_L(\theta)/\partial\theta$ or $\partial\eta(\theta)/\partial\theta$.

Given a single sample path, the realization of the random vector ξ is fixed. Therefore, we fix ξ and consider $\eta_L(\theta, \xi)$ as a function of θ. This function is called a *sample performance function.* Now, we consider the following question: given a sample path (θ, ξ), can we determine the sample path $(\theta + \Delta\theta, \xi)$ with the same ξ and $\Delta\theta/\theta \ll 1$? If we can, then we can get the performance for the perturbed system, $\eta_L(\theta + \Delta\theta, \xi)$, and furthermore, the derivative of the sample performance function:

$$\frac{\partial}{\partial\theta}\eta_L(\theta, \xi) = \lim_{\theta\to 0} \frac{\eta_L(\theta + \Delta\theta, \xi) - \eta_L(\theta), \xi)}{\Delta\theta} \tag{5}$$

This is called a *sample derivative.*

It seems reasonable to choose the sample derivative $\partial/\partial\theta$ $\eta_L(\theta, \xi)$ as an estimate for $\partial\overline{\eta}_L(\theta)/\partial\theta$ or $\partial\eta(\theta)/\partial\theta$. This estimate is called the infinitesimal perturbation analysis (IPA) estimate. We require that the estimate be unbiased or strongly consistent; that is, either

$$E\left\{\frac{\partial}{\partial\theta}[\eta_L(\theta, \xi)]\right\} = \frac{\partial}{\partial\theta}E[\eta_L(\theta, \xi)] \tag{6}$$

or

$$\lim_{L\to\infty}\left\{\frac{\partial}{\partial\theta}[\eta_L(\theta, \xi)]\right\} = \frac{\partial}{\partial\theta}\left\{\lim_{L\to\infty}[\eta_L(\theta, \xi)]\right\} \tag{7}$$

In Eq. (5), the same random variable ξ is used for both $\eta_L(\theta + \Delta\theta, \xi)$ and $\eta_L(\theta, \xi)$; this corresponds to the simulation technique "common random variable" in estimating the difference between two random functions. This technique usually leads to small variances. Equations (6) and (7) are referred to as the "interchangeability" in the literature (20). From the above discussion, the two basic issues for IPA are

1. To develop a simple algorithm that determines the sample derivative of Eq. (5) by analyzing a single sample path of a discrete event system; and
2. To prove that the sample derivative is unbiased and/or strongly consistent, that is, the interchangeability of Eq. (6) and/or Eq. (7) holds.

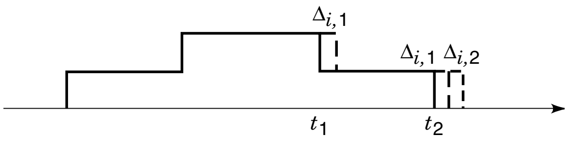

Figure 3. The perturbation in a busy period.

All the papers published in the area of IPA deal with these two basic issues. Roughly speaking, the interchangeability requires that the sample performance function $\eta_L(\theta, \xi)$ be continuous with respect to θ. General conditions can be found in Refs. 20 and 8.

The algorithms for obtaining sample derivatives are called *IPA algorithms.* Given a finite-length sample path (θ, ξ), we first, by applying IPA algorithms, fictitiously construct a sample path for the DEDS with a slightly changed parameter and the same random vector, $(\theta + \Delta\theta, \xi)$, called a *perturbed sample path.* The derivative of the performance with respect to θ can be obtained by comparing these two sample paths, the original one and the perturbed one.

The principles used in IPA to determine the perturbed path are very simple. We take closed queueing networks as an example. The basic idea is that a change in parameter (say, a mean service time) will induce changes of the service completion times, and a change of a customer's service completion time will affect the other customers' service completion times. IPA rules describe how these changes can be determined.

Figure 3 illustrates a busy period of a server, say server i, in a queueing network. Let $F_i(s, \theta)$ be its service time distribution. The service time of its kth customer is

$$s_{i,k} = F_i^{-1}(\xi_{i,k}, \theta) = sup\{s : F(s, \theta) \leq \xi_{i,k}\},$$

where $\xi_{i,k}$, $k = 1, 2, \cdots$, are uniformly distributed random variables on [0, 1). With the same $\xi_{i,k}$, in the perturbed system, the service time changes to $F_i^{-1}(\xi_{i,k}, \theta + \Delta\theta)$. Thus, the service time increases by

$$\begin{aligned}\Delta_{i,k} &= F_i^{-1}(\xi_{i,k}, \theta + \Delta\theta) - F_i^{-1}(\xi_{i,k}, \theta)\\ &= \frac{\partial F_i^{-1}(\xi_{i,k}, \theta)}{\partial\theta}\bigg|_{\xi_{i,k}=F_i(s_{i,k},\theta)} \Delta\theta\end{aligned} \tag{8}$$

Equation (8) is called the *perturbation generation rule,* and $\Delta_{i,k}$ is called the *perturbation generated* in the kth customer's service time. If the service time is exponentially distributed with its mean changed from $\bar{s}_i$ to $\bar{s}_i + \Delta\bar{s}_i$, then Eq. (8) becomes

$$\Delta_{i,k} = \frac{\Delta\bar{s}_i}{\bar{s}_i} s_{i,k} \tag{9}$$

The delay of a servers' service completion time is called the perturbation of the server, or the perturbation of the customer being served. In Fig. 3, the perturbation of the first customer is $\Delta_{i,1} = (s_{i,1}/\bar{s}_i)\,\Delta\bar{s}_i$. Because of this perturbation, the service starting time of the next customer is delayed by the same amount. Furthermore, the service time of the second customer increases by $\Delta_{i,2} = (s_{i,2}/\bar{s}_i)\,\Delta\bar{s}_i$, and thus the perturbation of the second customer is $\Delta_{i,1} + \Delta_{i,2}$ (see Fig. 3). In general, the service completion time of the kth customer in a

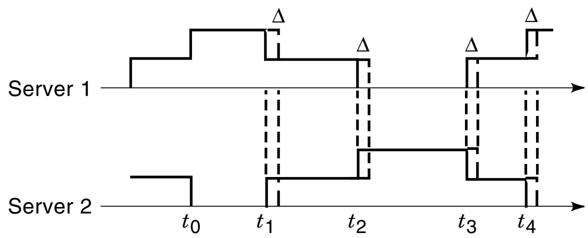

Figure 4. Propagation of a single perturbation.

busy period will be delayed by $\sum_{j=1}^{k} \Delta_{i,j}$, with $\Delta_{i,j}$ being determined by Eq. (8) or Eq. (9). This can be summarized as follows: a perturbation of a customer will be propagated to the next customer in the same busy period; the perturbation of a customer equals the perturbation generated in its service period plus the perturbation propagated from the preceding customer.

If a perturbation at the end of a busy period is smaller than the length of the idle period following the busy period, the perturbation will not affect (i.e., will not be propagated to) the next busy period, because the arrival time of the next busy period depends on another server's service completion time.

A perturbation at one server may affect other servers through idle periods. To see how servers may affect each other, we study the evolution of a single perturbation. In Fig. 4, at t_1, server 1 has a perturbation Δ, and before t_1, server 2 is idle. At t_1, a customer arrives from server 1 to server 2. Because server 1's service completion time is delayed by Δ, server 2's service starting time will also be delayed by Δ; and as a result, its service completion time will also be delayed by the same amount. We say the perturbation Δ is *propagated* from server 1 to server 2 through an idle period (server 2 has the same perturbation as server 1 after t_1). At t_3, this perturbation is propagated back to server 1.

In summary, if a perturbation is smaller than the lengths of idle periods (we say that the original and the perturbed paths are *similar*), then the evolution of this perturbation on the sample path can be determined by the following *IPA perturbation propagation rules:*

1. A perturbation of a customer at a server will be propagated to the next customer at the server until it meets an idle period, and
2. If, after an idle period, a server receives a customer from another server, then after this idle period the former will have the same perturbation as the latter (the perturbation is propagated from the latter to the former).

The perturbation generation rule describes how perturbations are generated because of a change in the value of a parameter; perturbation propagation rules describe how these perturbations evolve along a sample path after being generated. Combining these rules together, we can determine the perturbed path.

To apply the propagation rules, the size of the perturbation at the end of each busy period should not be larger than the length of the idle period that follows, and the size of the perturbation of a customer that terminates an idle period should not be larger than the length of the idle period; otherwise the idle period in the original sample path will disappear in the perturbed one and the simple propagation rules illustrated by Fig. 4 no longer hold. It is easy to see that for any finite-length sample path (θ, ξ), we can always (with probability one) choose a $\Delta\theta$ that is small enough (the size depends on ξ) such that the perturbations of all customers in the finite sample path are smaller than the shortest length of all the idle periods in the sample path. This explains the word "infinitesimal" in IPA. Therefore, we can always use IPA propagation rules to get the perturbed sample path and the sample derivative.

The perturbed sample path is completely determined by the perturbations of the servers. Given a sample path of a single-class closed Jackson network of M servers with $\bar{s}_m$, $m = 1, 2, \cdots, M$, being the mean service times, the perturbations of the perturbed system with $\bar{s}_i$ changed to $\bar{s}_i + \Delta\bar{s}_i$ (with i fixed) can be determined by the following algorithm.

IPA Algorithm for Closed Jackson Networks

0. Create a vector $\mathbf{v} = (v_1, v_2, \cdots, v_M)$; set its initial value $\mathbf{v} = (0, 0, \cdots, 0)$
1. At the kth, $k = 1, 2, \cdots$, service completion time of server i, set $v_i := v_i + s_{i,k}$
2. If on the sample path, a customer from server j terminates an idle period of server l, then set $v_l := v_j$.

Note that for simplicity in the algorithm we add $s_{i,k}$, instead of $(\Delta\bar{s}_i/\bar{s}_i)\, s_{i,k}$, to the perturbation vector. Thus, the perturbation of server m, $m = 1, 2, \cdots, M$, is $(\Delta\bar{s}_i/\bar{s}_i)\, v_m$, with v_m being determined by the algorithm. We shall see that the term $\Delta\bar{s}_i/\bar{s}_i$ is eventually cancelled in Eq. (11).

The sample derivative can be obtained from these perturbations. Let the sample performance measure be

$$\eta_L^{(f)} = \frac{1}{L}\int_0^{T_L} f[N(t)]\, dt \tag{10}$$

where $\mathbf{N}(t)$ is the state process and f is a function defined on the state space. The steady state performance measure is

$$\eta^{(f)} = \lim_{L\to\infty} \eta_L^{(f)} \qquad w.p.1$$

If $f \equiv I = 1$ for all $\mathbf{n}$, then we have $\eta_L^{(I)} = T_L/L$ and $\eta^{(I)} = 1/\eta$, where η is the throughout of the system. The sample derivative of $\eta_L^{(f)}$ can be easily obtained by adding performance calculations to the basic IPA algorithm as follows.

0. Set $\mathbf{v} = (0, 0, \cdots, 0)$, $H = 0$, and $\Delta H = 0$

1,2. Same as steps 1 and 2 in the basic IPA algorithm

3. At server m's service completion time, denoted as T_l, set $H = H + f[\mathbf{N}(T_l-)](T_l - T_{l-1})$ and $\Delta H = \Delta H + \{f[\mathbf{N}(T_l-)] - f[\mathbf{N}(T_l+)]\}v_m$.

In the algorithm, H records the value of the integral in Eq. (10), and

$$\frac{1}{L}\frac{\Delta\bar{s}_i}{\bar{s}_i}(\Delta H)$$

represents the difference of $\eta_L^{(f)}$ between the original and the perturbed paths. At the end of the sample path, we have

$$\eta_L^{(f)}(\bar{s}_i, \xi_i) = \frac{H}{L}$$

and

$$\Delta\eta_L^{(f)}(\bar{s}_i, \xi_i) = \frac{\Delta\bar{s}_i}{\bar{s}_i}\frac{\Delta H}{L}$$

Thus, we can calculate the sample elasticity as follows

$$\frac{\bar{s}_i}{\eta_L^{(f)}(\bar{s}_i, \xi)}\frac{\partial\eta_L^{(f)}(\bar{s}_i, \xi_i)}{\partial\bar{s}_i} = \frac{\Delta H}{H} \tag{11}$$

It has been shown that Eq. (11) is strongly consistent (9,6), that is,

$$\lim_{L\to\infty}\frac{\bar{s}_i}{\eta_L^{(f)}(\bar{s}_i, \xi)}\frac{\partial\eta_L^{(f)}(\bar{s}_i, \xi_i)}{\partial\bar{s}_i} = \frac{\bar{s}_i}{\eta^{(f)}}\frac{\partial\eta^{(f)}}{\partial\bar{s}_i} \qquad w.p.1$$

Similar algorithms and convergence results have been obtained for open networks, networks with general service time distributions, and networks in which the service rates depend on the system states (6). Glasserman (8) studied various IPA algorithms and their unbiasedness and strong consistency in the GSMP framework.

Extensions of IPA

For a sample derivative (i.e., the IPA estimate) to be unbiased and strongly consistent, usually requires that the sample function be continuous. This request, however, is not always satisfied. A typical example illustrating the failure of IPA is the two-server multiclass queueing network discussed in Ref. (21); later, Heidelberger et al. (22) discussed in detail a few extensions of the example.

In the past decade, many methods have been proposed to extend IPA to a wider class of problems. Each of these methods has some success on some problems at the cost of increasing analytical difficulty and computational complexity. We shall review only briefly the basic concepts of these methods.

Smoothed Perturbation Analysis. The idea of smoothed perturbation analysis (SPA) is to "average out" the discontinuity over a set of sample paths before taking the derivative and expectation. To illustrate the idea, we first write the expected value of $\eta_L(\theta, \xi)$ as

$$\bar{\eta}_L(\theta) = E[\eta_L(\theta, \xi)] = E\{E[\eta_L(\theta, \xi)\,|\,\mathscr{Z}]\}$$

where $\mathscr{Z}$ represents some random events in $(\Omega, \mathscr{F}, \mathscr{P})$. Let $\eta_L(\theta, \mathscr{Z}) = E[\eta_L(\theta, \xi)|\mathscr{Z}]$, then

$$\bar{\eta}_L(\theta) = E[\eta_L(\theta, \mathscr{Z})]$$

and Eq. (6) becomes

$$E\left\{\frac{\partial}{\partial\theta}[\eta_L(\theta, \mathscr{Z})]\right\} = \frac{\partial}{\partial\theta}E[\eta_L(\theta, \mathscr{Z})] \tag{12}$$

SPA attempts to find a suitable $\mathscr{Z}$ so that the "average" performance function $\eta_L(\theta, \mathscr{Z}$ is smooth enough and the interchangeability of Eq. (12) holds, even if Eq. (6) does not. If this is the case and the derivative of the conditional mean $\eta_L(\theta, \mathscr{Z})$ can be calculated, then $\partial/\partial\theta\ \eta_L(\theta, \mathscr{Z})$ can be used as an unbiased estimate of $\partial/\partial\theta\ \eta_L(\theta)$.

The method was first proposed in Ref. (23); a recent book Ref. (5) contains a detailed discussion about it. The main issues associated with this method are that it may not be easy to calculate $\partial/\partial\theta\ E[\eta_L(\theta, \mathscr{Z})]$ and that the computation effort required may be large.

Finite Perturbation Analysis. The sample derivative does not contain any information about the jumps in the performance function. This is because as $\Delta\theta$ goes to zero, the event sequence of any perturbed sample path is the same as that of the original path (two paths are similar). Thus, with IPA, we do not study the possibility that because of a parameter change $\Delta\theta$, two events may change their order. In finite perturbation analysis (FPA), a fixed size of $\Delta\theta$ is assumed. For any fixed $\Delta\theta$, the event sequence in the perturbed path may be different from that in the original path. FPA develops some rules that determine the perturbation when the event order changes. The FPA algorithm is more complicated than IPA, and it is usually approximate since only order changes between adjacent events are taken into account (9).

Sample Path Constructability Techniques. Given the nature of IPA, it cannot be applied to sensitivities with respect to changes of a fixed size or changes in discrete parameters. Motivated by the principles of IPA, we ask the following question: Given a sample path of discrete event system under parameter θ, it is possible to construct a sample path of the same system under a different parameter θ'? This problem is formulated as the *sample path constructability* (7). Normally, such constructability requires that the sets of events and states of the sample path to be constructed (with parameter θ') belong to the sets of events and states of the original sample path. For example, one may construct a sample path for an $M/M/1/K - 1$ queue (where $K - 1$ denotes the buffer size) from a sample path of an $M/M/1/K$ queue. Ref. 24 shows that for some systems with additional computation such sample path construction can be done even if some states in the sample path to be constructed do not appear in the original sample path.

Techniques in this class include *augmented system analysis* (7,25), *extended perturbation analysis* (26), and the *standard clock approach* (27).

Structural Infinitesimal Perturbation Analysis. Structural infinitesimal perturbation analysis (SIPA) was developed to address the problem of estimating the performance sensitivity with respect to a class of parameters such as the transition probabilities in Markov chains. At each state transition, in addition to the simulation of the original sample path, an extra simulation is performed to obtain a quantity needed to get the performance sensitivity. It has been shown that the extra simulation requires bounded computational effort, and that in some cases the method can be efficient (28). It is interesting to note that this approach can be explained by using the concept of realization discussed in the next subsection.

Rare Perturbation Analysis. Brémaud (29) studies the performance sensitivity with respect to the rate of a point process and proposes the method of rare perturbation analysis (RPA). The basic idea is that the perturbed Poisson process with rate $\lambda + \Delta\lambda$ with $\Delta\lambda > 0$ is the superposition of the original Poisson process with rate λ and an additional Poisson process with rate $\Delta\lambda$. Thus, in a finite interval, the difference between the perturbed path and the original one is rare. The performance derivative is then obtained by studying the effect of these "rare" but big (meaning finite) perturbations on the system performance. The case $\Delta\lambda > 0$ is called the *positive RPA*.

When $\Delta\lambda < 0$, the perturbed Poisson process with rate $\lambda + \Delta\lambda$ can be constructed by thinning the original Poisson process with the thinning probability $\Delta\lambda/\lambda$. That is, some arrival points in the original process will be taken away. The performance derivative is then obtained by studying the effect of the removal of these rare arrival points. This is called the *negative RPA*. Others in this direction include Refs. 30 and 31.

Estimation of Second Order Derivatives. The single path based approach can also be used to estimate the second order derivatives of the performance of a DEDS by calculating the conditional expectations. See Ref. 32 for GI/G/1 queues and Ref. 33 for Jackson networks.

Others. In addition to the above direct extensions of IPA, it also motivated the study of a number of other topics, such as the Maclaurin series expansion of the performance of some queueing systems (35), the rational approximation approach for performance analysis (36), and the analysis of performance discontinuity (37).

Finally, besides the PA method, there is another approach, called the *likelihood ratio* (LR) method (38–40), that can be applied to obtain estimates of performance derivatives. The method is based on the *importance sampling* technique in simulation. Compared with IPA, the LR method may be applied to more systems but the variances of the LR estimates are usually larger than those of IPA.

Perturbation Realization

One important concept regarding the sensitivity of steady state performance of a DEDS is the *perturbation realization*. The main quantity related to this concept is called the *realization factor*. This concept may provide a uniform framework for IPA and non-IPA methods. The main idea is: The realization factor measures the final effect of a single perturbation on the performance measure of a DEDS; the sensitivity of the performance measure with respect to a parameter can be decomposed into a sum of the final effects of all the single perturbations induced by the parameter change.

Perturbation Realization For Closed Jackson Networks. Suppose that at time $t = 0$, the network state is $\mathbf{n}$ and server i obtains a small perturbation Δ, which is the only perturbation generated on the sample path. This perturbation will be propagated through the sample path according to the IPA propagation rules and will affect system performance. The realization factor of a perturbation of server i at $t = 0$ with state $\mathbf{n}$, denoted as $c^{(f)}(\mathbf{n}, i)$, is defined as

$$c^{(f)}(\mathbf{n},i) = \lim_{L\to\infty} E\left\{\frac{1}{\Delta}\left[\int_0^{T'_L} f[\mathbf{N}'(t)]\,dt - \int_0^{T_L} f[\mathbf{N}(t)]\,dt\right]\right\} \tag{13}$$

where T'_L and $\mathbf{N}'(t)$ represents the quantities in the perturbed path.

A perturbation Δ is said to be *realized* if at some time T_l all the servers have the same perturbation Δ; it is said to be *lost* if at some time T_l no server has any perturbation. It was proved that in an irreducible closed network a perturbation will be either realized or lost with probability one. The probability that a perturbation is realized is called the *realization probability*.

Suppose that a perturbation is realized or lost at T_{L^*}. L^* depends on the sample path, that is, ξ. If the perturbation is lost, then $f[\mathbf{N}'(t)] = f[\mathbf{N}(t)]$, for all $t > T_{L^*}$; if it is realized, then $f[\mathbf{N}'(t)] = f[\mathbf{N}(t - \Delta)]$ for all $t > T_{L^*}$. Therefore, from the Markov property, Eq. (13) becomes

$$c^{(f)}(\boldsymbol{n},i) = E\left\{\frac{1}{\Delta}\left[\int_0^{T'_{L^*}} f[\mathbf{N}'(t)]\,dt - \int_0^{T'_{L^*}} f[\mathbf{N}(t)]\,dt\right]\right\} \tag{14}$$

where L^* is a random number, which is finite with probability one.

Realization factors can be uniquely determined by a set of linear equations (6). The steady state performance sensitivity can be obtained by

$$\frac{\bar{s}_i}{\eta^{(I)}}\frac{\partial\eta^{(f)}}{\partial\bar{s}_i} = \sum_{all\,\boldsymbol{n}} p(\boldsymbol{n})c^{(f)}(\boldsymbol{n},i) \tag{15}$$

where $I(\boldsymbol{n}) = 1$ for all $\boldsymbol{n}$ and $p(\boldsymbol{n})$ is the steady state probability of $\boldsymbol{n}$.

A close examination reveals that the IPA algorithm provides a simple way for estimating the quantity $\sum_{all\,\boldsymbol{n}} p(\boldsymbol{n})c^{(f)}(\boldsymbol{n}, i)$ on a single sample path. The theory has been extended to more general networks, including open networks, state-dependent networks, and networks with generally distributed service times (6).

Perturbation Realization for Markov Processes. Consider an irreducible and aperiodic Markov chain $X = \{X_n;\ n \geq 0\}$ on a finite state space $\mathscr{E} = \{1, 2, \cdots, M\}$ with transition probability matrix $P = [p_{ij}]_{i=1}^{M}|_{j=1}^{M}$. Let $\pi = (\pi_1, \pi_2, \cdots, \pi_M)$ be the vector representing its steady state probabilities, and $f = [f(1), f(2), \cdots, f(M)]^T$ be the performance vector, where T represents transpose and f is a column vector. The performance measure is defined as its expected value with respect to π:

$$\eta = E_\pi(f) = \sum_{i=1}^{M} \pi_i f(i) = \pi f. \tag{16}$$

Assume that P changes to $P' = P + \delta Q$, with $\delta > 0$ being a small real number and $Qe = 0$, $e = (1, 1, \cdots, 1)^T$. The performance measure will change to $\eta' = \eta + \Delta\eta$. We want to estimate the derivative of η in the direction of Q, defined as

$\partial\eta/\partial Q = \lim_{\delta\to 0} \Delta\eta/\delta$. It is well known that IPA does not work for this problem.

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 $q_{ki} = -\delta$, $q_{kj} = \delta$, and $q_{kl} = 0$ for all $l \neq i,j$. Suppose that in the original sample path the system is in state k and jumps to state i, then in the perturbed path, it may jump to state j instead. Thus, we study two independent Markov chains $X = \{X_n;\ n \geq 0\}$ and $\{X'_n;\ n \geq 0\}$ with $X_0 = i$ and $X'_0 = j$; both of them have the same transition matrix P. The *realization factor* is defined as (34):

$$d_{ij} = E\left\{\sum_{n=0}^{\infty}[f(X'_n) - f(X_n)]|X_0 = i, X'_0 = j\right\} \quad i,j = 1,2,\ldots,M \qquad (17)$$

Thus, d_{ij} represents the long term effect of a change from i to j on the system performance. Equation (17) is similar to Eq. (13).

If P is irreducibile, then with probability one the two sample paths of X and X' will merge together. That is, there is a random number L^* such that $X'_{L^*} = X_{L^*}$ for the first time. Therefore, from the Markov property, Eq. (17) becomes

$$d_{ij} = E\left\{\sum_{n=0}^{L^*-1}[f(X'_n) - f(X_n)]|X_0 = i, X'_0 = j\right\} \quad i,j = 1,2,\ldots,M \qquad (18)$$

which is similar to Eq. (14).

The matrix $D = [d_{ij}]$ is called a *realization matrix,* which satisfies the Lyapunov equation

$$D - PDP^T = F$$

where $F = ef^T - fe^T$, and $e = (1, 1, \cdots, 1)^T$ is a column vector all of whose components are ones. The performance derivative is

$$\frac{\partial\eta}{\partial Q} = \pi Q D^T \pi^T \qquad (19)$$

Since D is skew-symmetric, that is, $D^T = -D$, we can write $D = eg^T - ge^T$, where $g = [g(1), g(2), \cdots, g(M)]^T$ is called a *potential vector.* We have

$$\frac{\partial\eta}{\partial Q} = \pi Q g. \qquad (20)$$

$g(i)$ can be estimated on a single sample path by $g_n(i) = E\{\sum_{l=0}^{n}[f(X_l)]|X_0 = i\}$. There are a few other methods for estimating g and D by using a single sample path.

The potential g satisfies the Poisson equation

$$(I - P + e\pi)g = f \qquad (21)$$

Thus, perturbation realization in a Markov process relates closely to Markov potential theory and Poisson equations.

APPLICATIONS: ON-LINE OPTIMIZATION

A direct application of perturbation analysis is in the area of stochastic optimization. Because of the complexity of most discrete event systems, analytical approach does not usually exist for parametric optimization problems. One has to resort to simulation or experimental approaches, where the derivative estimates obtained by perturbation analysis can play an important role.

There are two major algorithms used in stochastic optimization: Kiefer–Wolfowitz (KW) and Robbins–Monro (RM). Both are essentially the hill-climbing type of algorithm. The KW algorithm employs the performance difference as an estimate of the gradient. With PA, we can obtain, based on a single sample path of a DEDS, the estimates of the gradients. Thus, the RM algorithm, which is known to be faster than the KW algorithm, can be used.

Suppose that we want to minimize a performance measure $\eta(\theta)$, where $\theta = (\theta_1, \theta_2, \cdots, \theta_M)$ is a vector of parameters. In the RM algorithm using PA derivative, the $(n + 1)$th value of the parameter θ, θ^{n+1}, is determined by (see, e.g., Ref. 41)

$$\theta^{n+1} = \theta^n - \alpha_n \overline{\frac{\partial\eta}{\partial\theta}}(\theta^n) \qquad (22)$$

where

$$\overline{\frac{\partial\eta}{\partial\theta}}(\theta^n) = \left[\overline{\frac{\partial\eta}{\partial\theta_1}}(\theta^n), \overline{\frac{\partial\eta}{\partial\theta_2}}(\theta^n), \cdots, \overline{\frac{\partial\eta}{\partial\theta_M}}(\theta^n)\right]$$

is the estimate of the gradient of the performance function $\eta(\theta)$ at θ^n with each component being the PA estimate, and α_n, $n = 1, 2, \cdots$, are the step sizes. It usually requires that $\sum_{n=1}^{\infty} \alpha_n = \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$.

Many results have been obtained in this direction. For example, Ref. 41 studied the optimization of $J(\theta) = T(\theta) + C(\theta)$ for a single server queues, where $T(\theta)$ is the mean system time, $C(\theta)$ a cost function, and θ the mean service time. It was proved that under some mild conditions, the Robbins–Monro type of algoritm (22) converges even if we update θ using the IPA gradient estimate at any random times (e.g., at every customer arrival time). Other works include Refs. 42, 43, 44, and 45.

The optimization procedures using perturbation analysis have been applied to a number of real-world problems. Successful examples include the bandwidth allocation problem in communications, (46,47), and optimization of manufacturing systems (48–52).

For performance optimization over discrete parameters, for example, in problems of choosing the best transition matrix, we may use the approach of realization matrix and potentials discussed in the last section. It is interesting to note that in this context, PA is equivalent to the Markov decision process (MDP) approach. To see this, let π' be vector of the steady state probability for the Markov chain with transition matrix P'. From the Poisson equation [Eq. (21)], it is easy to prove

$$\eta' - \eta = \pi' Q g \qquad (23)$$

The right-hand side of Eq. (23) is the same as that of Eq. (20) except π is replaced by π'. In policy iteration of an MDP problem, we choose the P' corresponding to the largest $Qg = (P' - P)g$ (component-wise) as the next policy. This corresponds to choosing the largest $\partial\eta/\partial Q$ in PA, because all the components of π and π' are positive. Therefore, the policy iteration procedure in MDP in fact chooses the steepest direction of the performance measure obtained by PA as the policy in the next iteration. Thus, in this setting, PA is simply a single sample-path-based implementation of MDP. Further research is needed in this direction. Another on-going research related to DEDS optimization is the *ordinal optimization* technique (53), whose main idea is that by softening the goal of optimization and by comparing different schemes ordinally instead of obtaining the exact performance values, we can dramatically reduce the demand in the accuracy of the estimates.

BIBLIOGRAPHY

1. Y. C. Ho (ed.), *Dynamics of Discrete Event Systems, Proc. IEEE,* **77**: 1–232, 1989.
2. P. J. Ramadge and W. M. Wonham, Supervisory control of a class of discrete event processes, *SIAM J. Control Optim.,* **25**: 206–230, 1987.
3. W. H. Fleming, (chair), *Future directions in control theory—A mathematical perspective,* Report of the Panel on Future Directions in Control Theory, 1988.
4. R. David and H. Alla, Petri nets for modeling of dynamic systems—A survey, *Automatica,* **30**: 175–202, 1994.
5. M. C. Fu and J. Q. Hu, *Conditional Monte Carlo: Gradient Estimation and Optimization Applications,* Norwell, MA: Kluwer Academic Publishers, 1997.
6. X. R. Cao, *Realization Probabilities: the Dynamic of Queueing Systems,* New York: Springer-Verlag, 1994.
7. C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis,* Aksen Associates, Inc., 1993.
8. P. Glasserman, *Gradient Estimation Via Perturbation Analysis,* Norwell, MA: Kluwer Academic Publishers, 1991.
9. Y. C. Ho and X. R. Cao, *Perturbation Analysis of Discrete-Event Dynamic Systems,* Norwell, MA: Kluwer Academic Publishers, 1991.
10. P. G. Harrison, On normalizing constants in queueing networks, *Oper. Res.* **33**: 464–468, 1985.
11. L. Kleinrock, *Queueing Systems,* vols. I, II, New York: Wiley, 1975.
12. J. Walrand, *An Introduction to Queueing Networks,* Englewood Cliffs, NJ: Prentice Hall, 1988.
13. S. H. Lu and P. R. Kumar, Distributed scheduling based on due dates and buffer priorities, *IEEE Trans. Autom. Control,* **AC-36**: 1406–1416, 1991.
14. J. L. Peterson, *Petri Net Theory and the Modeling of Systems,* Englewood Cliffs, NJ: Prentice Hall, 1981.
15. C. Reutenauer, *The Mathematics of Petri Nets,* London: Prentice-Hall, 1990.
16. M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems,* Norwell, MA: Kluwer Academic Publisher, 1993.
17. G. Cohen et al., A linear system theoretical view of discrete event processes and its use for performance evaluation in manufacturing, *IEEE Trans. Autom. Control,* **AC-30**: 210–220, 1985.
18. F. Baccelli et al., *Synchronization and Linearity,* New York: Wiley, 1992.
19. Y. C. Ho, A. Eyler, and T. T. Chien, A gradient technique for general buffer storage design in a serial production line, *Int. J. Production Res.,* **17**: 557–580, 1979.
20. X. R. Cao, Convergence of parameter sensitivity estimates in a stochastic experiment, *IEEE Trans. Autom. Control,* **AC-30**: 834–843, 1985.
21. X. R. Cao, First-order perturbation analysis of a single multiclass finite source queue, *Performance Evaluation,* **7**: 31–41, 1987.
22. P. Heidelberger et al., Convergence properties of infinitesimal perturbation analysis estimates, *Management Science,* **34**: 1281–1302, 1988.
23. W. B. Gong and Y. C. Ho, Smoothed perturbation analysis of discrete event dynamic systems, *IEEE Trans. Autom. Control,* **32**: 858–866, 1987.
24. Y. Park and E. K. P. Chong, Distributed inversion in timed discrete event systems, *Discrete Event Dynamic Systems: Theory and Applications,* **5**: 219–241, 1995.
25. A. A. Gaivoronski, L. Y. Shi, and R. S. Sreenivas, Augmented infinitesimal perturbation analysis: An alternate explanation, *Discrete Event Dynamic Systems: Theory and Applications,* **2**: 121–138, 1992.
26. Y. C. Ho and S. Li, Extensions of perturbation analysis of discrete event dynamic systems, *IEEE Trans. Autom. Control,* **33**: 427–438, 1988.
27. P. Vakili, Using a standard clock technique for efficient simulation, *Oper. Res. Lett.,* **11**: 445–452, 1991.
28. L. Y. Dai and Y. C. Ho, Structural infinitesimal perturbation analysis (SIPA) for derivative estimation of discrete event dynamic systems, *IEEE Trans. Autom. Control,* **40**: 1154–1166, 1995.
29. P. Brémaud, Maximal coupling and rare perturbation sensitivity analysis, *Queueing Systems: Theory and Applications,* **10**: 249–270, 1992.
30. F. Baccelli, M. Klein, and S. Zuyev, Perturbation analysis of functionals of random measures, *Adv. in Appl. Prob.,* **27**: 306–325, 1995.
31. F. J. Vazquez-Abad and K. Davis, Efficient implementation of the phantom RPA method, with an application to a priority queueing system, *Adv. in Appl. Prob.,* submitted, 1995.
32. M. A. Zazanis and R. Suri, Perturbation analysis of the GI/G/1 queue, *Queueing Systems: Theory and Applications,* **18**: 199–248, 1994.
33. B. Gang, C. Cassandras, and M. A. Zazanis, First and second derivative estimators for closed Jackson-like queueing networks using perturbation analysis techniques, *Discrete Event Dynamic Systems: Theory and Applications,* **7**: 29–68, 1997.
34. X. R. Cao and H. F. Chen, Perturbation realization, potentials, and sensitivity analysis of Markov processes, *IEEE Trans. Autom. Control,* **42**: 1382–1393, 1997.
35. W. B. Gong and S. Nananukul, Rational interpolation for rare event probabilities, *Stochastic Networks: Stability and Rare Event,* New York: Springer, 1996.
36. W. B. Gong and J. Q. Hu, On the MacLauring Series of GI/G/1 Queue, *J. Appl. Prob.,* **29**: 176–184, 1991.
37. X. R. Cao, W. G. Gong, and Y. Wardi, Ill-conditioned performance functions of queueing systems, *IEEE Trans. Autom. Control,* **40**: 1074–1079, 1995.
38. P. W. Glynn, Likelihood ratio gradient estimation: An overview, *Proc. Winter Simulation Conf.,* 366–375, 1987.
39. M. I. Reiman and A. Weiss, Sensitivity analysis via likelihood ratio, *Oper. Res.,* **37**: 830, 844, 1989.
40. R. Y. Rubinstein, Sensitivity analysis and performance extrapolation for computer simulation models, *Oper. Res.,* **37**: 72–81, 1989.
41. E. K. P. Chong and P. J. Ramadge, Optimization of queues using infinitesimal perturbation analysis-based algorithms with general update times, *SIAM J. Control Optim.,* **31**: 698–732, 1993.
42. Y. C. Ho and X. R. Cao, Perturbation analysis and optimization of queueing networks, *J. Optim. Theory Appl.,* **40**: 559–582, 1983.

43. C. G. Cassandras and S. G. Strickland, On-line sensitivity analysis of Markov chains, *IEEE Trans. Autom. Control,* **34**: 76–86, 1989.

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

45. Q. Y. Tang and H. F. Chen, Convergence of perturbation analysis based optimization algorithm with fixed number of customers period, *Discrete Event Dynamic Systems: Theory and Applications,* **4**: 359–375, 1994.

46. C. A. Brooks and P. Varaiya, Using perturbation analysis to solve the capacity and flow assignment problem for general and ATM networks, *IEEE Globcom,* 1994.

47. N. Xiao, F. F. Wu, and S. M. Lun, Dynamic bandwidth allocation using infinitesimal perturbation analysis, *IEEE Infocom,* 383–389, 1994.

48. M. Caramanis and G. Liberopoulos, Perturbation analysis for the design of flexible manufacturing system flow controllers, *Oper. Res.,* **40**: 1107–1125, 1992.

49. A. Haurie, P. L'Ecuyer, and C. van Delft, Convergence of stochastic approximation coupled with perturbation analysis in a class of manufacturing flow control models, *Discrete Event Dynamic Systems: Theory and Applications,* **4**: 87–111, 1994.

50. H. M. Yan and X. Y. Zhou, Finding optimal number of Kanbans in a manufacturing system via perturbation analysis, *Lecture Notes in Control and Information Sciences,* 199, Springer-Verlag, 572–578, 1994.

51. H. M. Yan, G. Yin, and S. X. C. Lou, Using stochastic optimization to determine threshold values for the control of unreliable manufacturing systems, *J. Optim. Theory Appl.,* **83**: 511–539, 1994.

52. N. Miyoshi and T. Hasegawa, On-line derivative estimation for the multiclass single-server priority queue using perturbation analysis, *IEEE Trans. Autom. Control,* **41**: 300–305, 1996.

53. Y. C. Ho, *Soft Optimization for Hard Problems,* Computerized lecture via private communication, 1996.

XI-REN CAO
The Hong Kong University of
Science and Technology