European Journal of Operational Research 218 (2012) 293–304

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Invited Review

Performance optimization of queueing systems with perturbation realization

Li Xia [a], Xi-Ren Cao [b,c,*]

[a] Division of Mathematical and Computer Sciences & Engineering, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[b] Antai College of Business and School of Electronic Engineering, Shanghai Jiaotong University, China
[c] Institute for Advanced Study, The Hong Kong University of Science and Technology, Hong Kong, China

ARTICLE INFO

Article history:
Received 17 August 2010
Accepted 22 July 2011
Available online 10 August 2011

Keywords:
Queueing
Perturbation analysis
Perturbation realization factors
Performance potential
Sensitivity-based optimization

ABSTRACT

After the intensive studies of queueing theory in the past decades, many excellent results in performance analysis have been obtained, and successful examples abound. However, exploring special features of queueing systems directly in performance optimization still seems to be a territory not very well cultivated. Recent progresses of perturbation analysis (PA) and sensitivity-based optimization provide a new perspective of performance optimization of queueing systems. PA utilizes the structural information of queueing systems to efficiently extract the performance sensitivity information from a sample path of system. This paper gives a brief review of PA and performance optimization of queueing systems, focusing on a fundamental concept called perturbation realization factors, which captures the special dynamic feature of a queueing system. With the perturbation realization factors as building blocks, the performance derivative formula and performance difference formula can be obtained. With performance derivatives, gradient-based optimization can be derived, while with performance difference, policy iteration and optimality equations can be derived. These two fundamental formulas provide a foundation for performance optimization of queueing systems from a sensitivity-based point of view. We hope this survey may provide some inspirations on this promising research topic.

1. Introduction

Queueing theory has been studied for more than one hundred years since the earliest paper by Erlang appeared. In the recent decades, the research of queueing theory becomes even more intensive because of the fast advancement of modern computer and communication networks. Queueing systems are now widely used to model many engineering systems, such as communication networks, logistics management, manufacturing systems, and transportation systems (Chen, 2008; Heidergott, 1998; Rappold and Roo, 2009; Woensel et al., 2008). Performance optimization of queueing systems is therefore important in the practice.

Since many queueing systems can be modeled as Markov processes, the performance optimization of queueing systems, may also be referred to as controlled queueing systems, is usually studied using the theory of Markov decision processes (MDP). For example, we can use the optimality equation in MDP to study the structure properties of optimal control policies of queueing systems. For a comprehensive knowledge, audience can refer to the books and survey papers (Kitaev and Rykov, 1995; Sennott, 1999; Stidham and Weber, 1993). However, performance optimization of queueing systems based on MDP cannot be used for the non-Markovian queues, such as the G/G/1 queues. Moreover, this methodology is only applied to analyze simple Markovian queues. For example, the monotonicity of optimal policies has been studied for simple tandem queues and cyclic queues (Stidham and Weber, 1993). But it is hard to study similar properties for a general Jackson network. Because the related optimality equation of MDP is too complicated to analyze when the queueing system is not simple enough.

Performance optimization of queueing systems based on perturbation analysis (PA) is another type of methodology. PA extracts performance sensitivity information from a sample path of system. Compared with MDP, PA may be applied to general queueing systems, which may not be Markov. Moreover, PA can efficiently explore the structure properties of queueing systems and therefore provide more sensitivity information than MDP. This paper surveys the PA-based performance optimization theory for queueing systems. We hope this review may provide some inspirations for, and attract some attentions to, this promising research topic.

PA of queueing systems was proposed about 30 years ago; it utilizes the structural information of a queueing system to derive performance derivatives by analyzing a single sample path of system. The basic idea of PA is: the change of a system parameter generates (*perturbation generation*) a series of perturbations on a sample path (a perturbation may be a delay of service completion time at a server, or a delay of arrival time of a customer, etc.); each perturbation will affect the system dynamics evolution and the

* Corresponding author.
E-mail addresses: li.xia@kaust.edu.sa (L. Xia), xrcao@sjtu.edu.cn, eecao@ust.hk (X.-R. Cao).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.07.039

system performance; the total effect of a perturbation on the system performance is measured by a quantity called the *perturbation realization factor* (*perturbation realization*), which depends on the system's special structure; and the derivative of the performance with respect to a perturbed parameter can be decomposed into the sum of the effects of all the perturbations generated by the change of that parameter. Thus, perturbation realization factors can be viewed as building blocks for constructing the performance derivatives. With the performance derivatives obtained by PA, gradient-based algorithms or stochastic approximation (Robbins–Monro type) algorithms can be developed for performance optimization (Cao, 2007).

In the last decade, the concept of perturbation realization has been extended to Markov systems (Cao, 2000, 2003; Cao and Chen, 1997). For discrete state Markov systems, a policy corresponds to a transition probability matrix, and a perturbation is a change on a sample path from one state to another. A similar construction procedure based on perturbation generation and perturbation realization leads to not only the performance derivative (with respect to the change in transition probability matrix) formula, but also the performance difference formula which decomposes the difference of the performance of two policies into the sum of perturbation realization factors (note that they are different from those in queueing systems). In Markov systems, it is shown that the perturbation realization factor of a perturbation from state i to state j equals the difference of two *performance potentials* at these two states i and j; and therefore, the performance potentials become the building blocks for performance sensitivity formulas. Based on the difference formula, policy iteration and online algorithms are developed, and the optimality (Hamilton–Jacobi–Bellman or H–J–B) equation can be derived (Cao, 2007). Audience can also refer to Li (2010) for a review of the theory of perturbed Markov systems.

The work on performance difference formula and policy iteration shed new insights to performance optimization of queueing systems. Many queueing systems can be modeled as Markov processes, thus the perturbation realization factors in queueing systems must be related to performance potentials in the Markov model. Therefore, we may develop performance difference formulas for queueing systems using perturbation realization factors (in the form of queueing systems) as building blocks. Indeed, we successfully derived the performance difference formula for queueing systems with any two sets of service rates, using the perturbation realization factors; and based on which, we developed policy iteration (online) algorithms for queueing systems. Such algorithms usually converge faster than the gradient-based algorithms. In addition, we derive the H–J–B optimality equation for queueing systems with perturbation realization factors.

In summary, there are two types of optimization approaches for queueing systems with PA theory, the gradient based and the policy iteration based. They are derived from performance derivative or performance difference formulas, respectively. These formulas are constructed by using perturbation realization factors as building blocks, which reflect the special structure of queueing systems and can be estimated based on a single sample path with an online manner.

This paper provides an overview of the results above. In Section 2, we briefly review the basic principles of PA and the traditional PA theory of queueing systems, including perturbation generation, perturbation realization, and the performance derivative formula. In Section 3, we introduce the sensitivity-based optimization of Markov systems, which can be viewed as an extension of the traditional PA theory. In Section 4, we review the recent progress on the performance difference formula and policy iteration for queueing systems, by linking the results in the two sections above together. In Section 5, we give a brief survey on other PA related approaches, including finite PA, smoothed PA, and PA with stochastic fluid model. Finally, to conclude this paper, we give a discussion in Section 6.

2. Traditional perturbation analysis of queueing systems

The traditional PA theory aims to provide an efficient way to estimate the performance derivatives with respect to system parameters in discrete event dynamics systems (DEDS), especially in queueing systems. The earliest formal presentation of PA of queueing systems was proposed by Ho and Cao (1983) and Ho et al. (1983). In the following three decades, many researchers have been devoted to establishing the PA theory and to developing many variants of PA approaches.

After many years' exploration, we come up with the following basic ideas for PA: the derivative of system performance with respect to system parameters (say the service rates, arrival rates, or routing probabilities in queueing systems) can be decomposed into the sum of many small building blocks, called the *perturbation realization factor* (Cao, 1987), each of which measures the effect of a single perturbation on the system performance. Such decomposition is done by utilizing the special structure of queueing systems and it makes PA very efficient. Perturbation realization factors can be estimated based on a single sample path of a queueing system, so is the performance derivative with respect to parameters. In this section, we give an overview of the main results of the traditional PA theory in queueing systems.

We first give a brief explanation for a few simple principles of PA in queueing systems. For example, in a tandem queueing network with 2 servers, if we decrease the service rate of server 1 with an (infinitesimal) amount, the service time of any customer at this server will increase and the service completion time of customers at server 1 will be delayed for a small amount. Such a delay is called a *perturbation*. This is called *perturbation generation*. Perturbations may also be generated by the delay of arrival time of customers, infinitesimal change of routing probabilities, etc.

Next, the delay of the service completion time of the current customer will postpone the service start time of the next customer waiting at server 1. In addition, if a customer completes its service at server 1 and enters the next server, i.e., server 2, which is in the idle state, a perturbation of this customer at server 1 will also delay the service start time of this customer at server 2, i.e., server 2 will also get a perturbation. This is called *perturbation propagation*.

From the description above, a perturbation will affect the system dynamics through propagations. The average effect of a perturbation on the system performance can be measured by a quantity called the *perturbation realization factor*. This is called *perturbation realization*. Obviously, perturbation realization factors depend strongly on the structure of queueing systems.

Finally, the effect of a change in the service rate of server 1 can be decomposed into the sum of the effects of all the perturbations generated due to the rate change. In words, the basic principle of PA theory in queueing systems can be summarized as the following three fundamental phases: perturbation generation, perturbation propagation, and perturbation realization. This process is illustrated by Fig. 1.

Now, let us provide a detailed introduction of PA and derive the PA algorithms, with the closed Jackson network (also called Gordon–Newell network (Gordon and Newell, 1967)) as an example (see Fig. 2). Please note, the same principles may be also applied to more general queueing systems, such as open networks, networks with state-dependent service rates, and non-Markovian queues (see Cao (1994), Cao (2007), and the references therein).

Consider a closed Jackson network with M servers and N customers. The service time of customers at server i obeys an

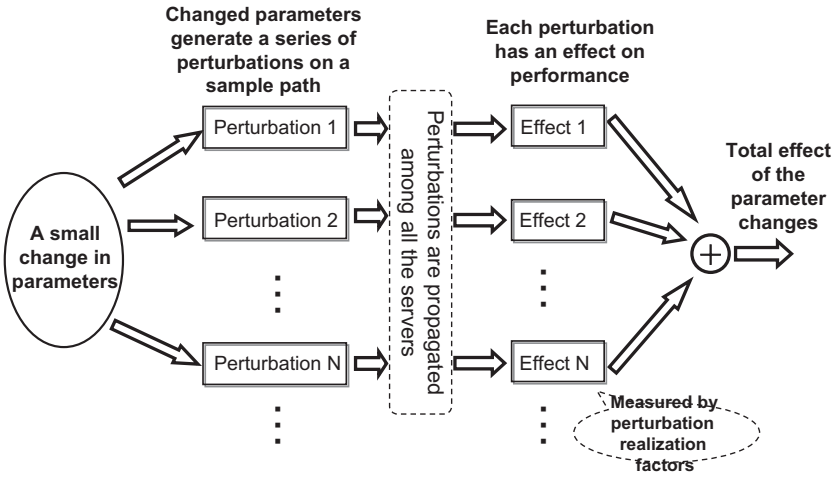

Fig. 1. The basic principles of perturbation analysis.

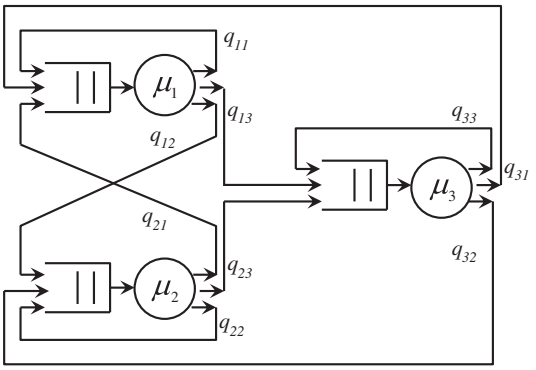

Fig. 2. A closed Jackson network with 3 servers.

exponential distribution with mean $1/\mu_i$, where μ_i is the service rate of server i, $i = 1, 2, \ldots, M$. When a customer completes its service at server i, it will go to another server j with a routing probability q_{ij}, $i,j = 1,2,\ldots,M$, $\sum_{j=1}^{M} q_{ij} = 1$ for all i. When a server is not idle, the newly arriving customers will enter the server's buffer and wait for service according the FCFS (first come first serve) discipline. The number of customers (including the customer being served) at server i is denoted as n_i and the system state is denoted as $\boldsymbol{n} = (n_1, n_2, \ldots, n_M)$. The state space is then defined as $\mathcal{S} = \{all\ \boldsymbol{n} : \sum_{i=1}^{M} n_i = N\}$. Let $\boldsymbol{n}(t) = (n_1(t), n_2(t), \ldots, n_M(t))$ be the system state at time t where $n_i(t)$ is the number of customers at server i at time t, $i = 1, 2, \ldots, M$, $t \geqslant 0$. Define T_l as the lth state transition time of the process $\boldsymbol{n}(t)$ and T_l can also be understood as the time when the number of served customers by all servers reaches l. Fig. 3 illustrates an example of sample path where the stair-style lines mean the trajectory of the number of customers at a server. The dotted lines in Fig. 3 indicate the state transitions and the arrows indicate the directions of customer transitions from one server to another.

Define the cost (or reward) function as $f(\boldsymbol{n})$, where $f(\boldsymbol{n}) \in \mathcal{R}$ and $\boldsymbol{n} \in \mathcal{S}$. The system *time-average performance* η_T is defined as

$$\eta_T = \lim_{l\to\infty} \frac{1}{T_l} \int_0^{T_l} f(\boldsymbol{n}(t))dt = \lim_{l\to\infty} \frac{F_l}{T_l}, \quad w.p.1, \tag{1}$$

where $F_l := \int_0^{T_l} f(\boldsymbol{n}(t))dt$ denotes the accumulated costs until T_l. In addition, *customer-average performance* η_C is defined as

$$\eta_C = \lim_{l\to\infty} \frac{1}{l} \int_0^{T_l} f(\boldsymbol{n}(t))dt = \lim_{l\to\infty} \frac{F_l}{l}, \quad w.p.1. \tag{2}$$

When the network is strongly connected (any customer may visit every server in the network), $\boldsymbol{n}(t)$ is ergodic. Thus, the limits in (1) and (2) exist with probability one (w.p.1). It is clear that η_T is

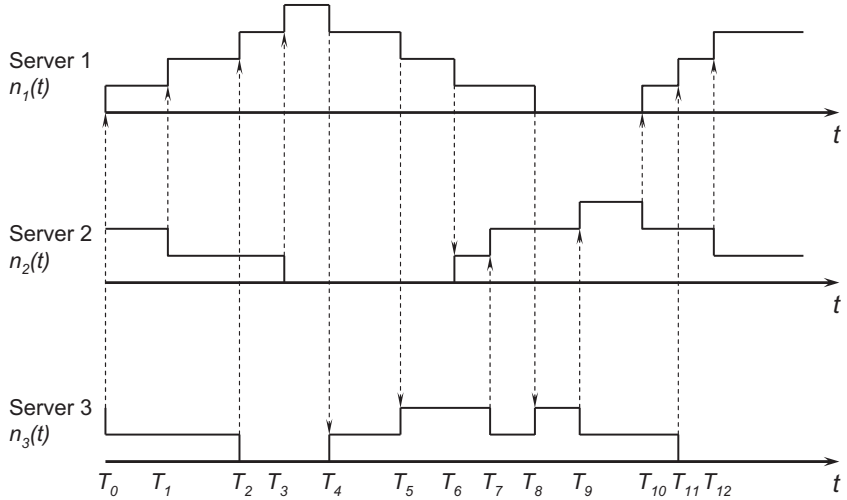

Fig. 3. Example of a sample path of a closed Jackson network with 3 servers and 4 customers.

the system performance averaged for each unit of time, while η_C is the system performance averaged for each customer. These two performance metrics are both important for queueing systems. For example, average queue length is a time-average performance, while average waiting time is a customer-average performance. Obviously, they have the following relation

$$\eta_C = \lim_{l\to\infty}\frac{T_l}{l}\lim_{l\to\infty}\frac{1}{T_l}\int_0^{T_l} f(\boldsymbol{n}(t))dt = \frac{\eta_T}{\eta_{th}} = \eta_T\eta_I, \tag{3}$$

where $\eta_{th} := \lim_{l\to\infty}\frac{l}{T_l}$ is the average throughput of the whole network, η_I is a special case of η_C when $f(\boldsymbol{n}) = l(\boldsymbol{n}) \equiv 1$ for all $\boldsymbol{n}$ and $\eta_I = 1/\eta_{th}$. Since the change of a parameter may affect both η_T and η_{th}, the optimization results for η_T and η_C are generally different (Xia et al., 2009).

Suppose that the service rate of a server, say server i, changes from μ_i to $\mu_i + \Delta\mu_i$, where $\Delta\mu_i \to 0$. Since the service time of server i is exponentially distributed, based on the inverse transform method, the service time is generated according to $s = -\frac{1}{\mu_i}\ln\zeta$, where ζ is a uniformly distributed random number in $[0,1]$. Because of the change in service rate, the service time s will be changed to $s' = -\frac{1}{\mu_i+\Delta\mu_i}\ln\zeta = -\frac{1-\Delta\mu_i/\mu_i}{\mu_i}\ln\zeta + o(\Delta)$. So, the perturbation of service time is

$$\Delta s = s' - s = -\frac{\Delta\mu_i}{\mu_i}s. \tag{4}$$

In summary, because of the infinitesimal change of service rate $\Delta\mu_i$, every customer's service time at server i will obtain a perturbation whose amount is proportional to its original service time with a multiplier $-\frac{\Delta\mu_i}{\mu_i}$. This is the rule of perturbation generation.

After the perturbation is generated, it will be propagated throughout the network according to some rules. First, the perturbations at a server will accumulate until this server is idle. When a server becomes idle, all its perturbations will be lost. (After the idle period, the service start time is determined by the customer that terminates the idle period, which has the perturbations of another server.) Second, when a customer finishes its service at server i and enters server j, if server j is idle at this time, server j will obtain the same amount of perturbations as server i; otherwise, the perturbations of server i will only affect the arrival time of this customer to server j, which does not have any influence on the following dynamics of server j.

The process described above is illustrated by Fig. 4, where we just consider that the service rate of server 1 is decreased with an infinitesimal amount. This figure is based on the same sample path of Fig. 3 and it demonstrates the process of perturbation generation and propagation in a queueing network. The effect of perturbations on system performance can also be observed in this figure since the distribution of system states is also changed slightly.

From the introduction above, we get a clear picture about the dynamics of perturbations in a queueing network. During the evolution of a sample path of the system (either from simulation or running a real system), we record these perturbations and get a perturbed sample path and finally get the performance changes caused by these perturbations. All the calculations only require a very few additional storage and computation resources during the original system process. Thus, we can get an estimate of performance derivative with respect to the perturbed parameters based on a single sample path.

To quantitatively analyze the effect of perturbations on the system performance, we define the perturbation realization factors. Consider a single perturbation Δ of service time of server i when the system is at state $\boldsymbol{n}$. The effect of this perturbation on the total system performance can be measured by the *perturbation realization factor* $c^{(f)}(\boldsymbol{n},i)$, which is defined as follows.

$$\begin{aligned} c^{(f)}(\boldsymbol{n},i) &= \lim_{l\to\infty}\lim_{\Delta\to 0} E\left\{\frac{\Delta F_l}{\Delta}\right\} = \lim_{l\to\infty}\lim_{\Delta\to 0} E\left\{\frac{F'_l - F_l}{\Delta}\right\} \\ &= \lim_{l\to\infty}\lim_{\Delta\to 0} E\left\{\frac{1}{\Delta}\left[\int_0^{T'_l} f(\boldsymbol{n}'(t))dt - \int_0^{T_l} f(\boldsymbol{n}(t))dt\right]\right\}, \end{aligned} \tag{5}$$

where $\boldsymbol{n}'(t)$ is the state of the perturbed system (with the perturbation Δ at time 0) at time t, T'_l is the lth service completion time on the perturbed sample path. It is clear that $c^{(f)}(\boldsymbol{n},i)$ measures the long term effect of a unit perturbation at $(\boldsymbol{n},i)$ on F_l on average. By (5), the *perturbation realization probability* $c(\boldsymbol{n},i)$ is defined as a special case of $c^{(f)}(\boldsymbol{n},i)$ when $f(\boldsymbol{n}) = l(\boldsymbol{n}) \equiv 1$ for all $\boldsymbol{n} \in \mathcal{S}$. That is, $c(\boldsymbol{n},i)$ is the probability that a single perturbation of service time of server i at state $\boldsymbol{n}$ will be finally propagated to all the servers. In other words, this

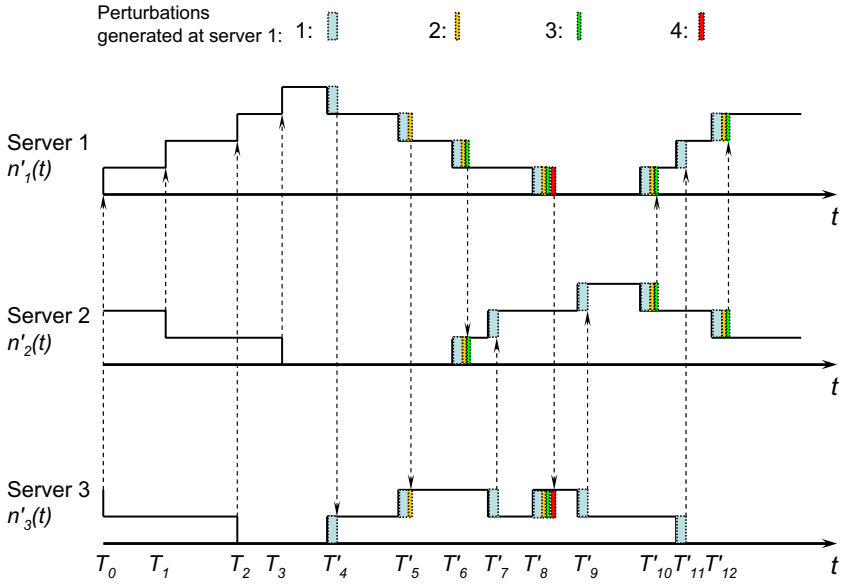

Fig. 4. Process of perturbation generation and propagation in a queueing network.

perturbation will be totally lost in the network with the probability $1 - c(\boldsymbol{n}, i)$.

The perturbation realization factors can be uniquely determined by the following set of linear equations (Cao, 1994)

$$\text{If } n_i = 0, \quad \text{then } c^{(f)}(\boldsymbol{n}, i) = 0; \tag{6}$$

$$\sum_{i=1}^{M} c^{(f)}(\boldsymbol{n}, i) = f(\boldsymbol{n}); \tag{7}$$

$$\begin{aligned}\left\{\sum_{k=1}^{M} \mathbf{1}(n_k)\mu_k\right\} c^{(f)}(\boldsymbol{n}, i) &= \sum_{k=1}^{M}\sum_{j=1}^{M} \mathbf{1}(n_k)\mu_k q_{kj} c^{(f)}(\boldsymbol{n}_{kj}, i) \\ &\quad + \sum_{j=1}^{M} \mu_i q_{ij}\left\{\left[1 - \mathbf{1}(n_j)\right] c^{(f)}(\boldsymbol{n}_{ij}, j) + f(\boldsymbol{n}) - f(\boldsymbol{n}_{ij})\right\};\end{aligned} \tag{8}$$

where $\boldsymbol{n}_{ij} = (n_1, \ldots, n_i - 1, \ldots, n_j + 1, \ldots, n_M)$ is a neighboring state of $\boldsymbol{n}$ with $n_i > 0$, $\mathbf{1}(n_i)$ is an indicator function which is defined as, if $n_i > 0$, $\mathbf{1}(n_i) = 1$; otherwise $\mathbf{1}(n_i) = 0$. Eq. (6) is simply a convention: When a server is idle, the perturbation of its service time has no effect on the queueing system. Eq. (7) means that if all the service times of servers have the same delay Δ at state $\boldsymbol{n}$, it equals the effect of the whole sample path being shifted by a time Δ, thus the induced performance perturbation is $f(\boldsymbol{n})\Delta$ and the sum of perturbation realization factors is $f(\boldsymbol{n})$. Eq. (8) reflects how a perturbation at server i is propagated to other servers. The first term of right-hand side of (8) measures the effect that the perturbation is kept at server i. The second term of right-hand side of (8) measures the effect that an idle server j obtain the perturbation propagated from server i. The term $f(\boldsymbol{n}) - f(\boldsymbol{n}_{ij})$ is the effect due to the delay of the customer transition from server i to server j.

The performance derivative formula based on perturbation realization factors can be intuitively explained as follows. Suppose that the service rate μ_i is changed to $\mu_i + \Delta\mu_i$ where $\Delta\mu_i$ is an infinitesimal amount. For every state $\boldsymbol{n} \in \mathcal{S}$, the service time at server i will have a delay $-\frac{\Delta\mu_i}{\mu_i} s$ where s is the original service time. Therefore, for a time period T_l with $l \to \infty$, the total amount of service time delay of server i at state $\boldsymbol{n}$ is $-\frac{\Delta\mu_i}{\mu_i} T_l \pi(\boldsymbol{n})$, where $\pi(\boldsymbol{n})$ is the steady-state probability of state $\boldsymbol{n}$. Since the effect of a unit perturbation of service time at $(\boldsymbol{n}, i)$ is measured by $c^{(f)}(\boldsymbol{n}, i)$, the total effect of $\Delta\mu_i$ on performance F_l is

$$\Delta F_l = -\sum_{\boldsymbol{n}\in\mathcal{S}} \frac{\Delta\mu_i}{\mu_i} T_l \pi(\boldsymbol{n}) c^{(f)}(\boldsymbol{n}, i). \tag{9}$$

Divided by $\Delta\mu_i l$ on both sides and let $l \to \infty$ and $\Delta \to 0$, (9) can be written as the following performance derivative formula

$$\frac{\mu_i}{\eta_l}\frac{d\eta_C}{d\mu_i} = -\sum_{\boldsymbol{n}\in\mathcal{S}} \pi(\boldsymbol{n}) c^{(f)}(\boldsymbol{n}, i). \tag{10}$$

Therefore, with the perturbation realization factors as building blocks, we may obtain the performance derivative formula (10). The effect of the perturbed parameters equals the total effect of all small perturbations generated and the effect of each small perturbation can be represented by perturbation realization factors.

With (10), we can develop an efficient estimation algorithm of performance derivatives in queueing systems, which obeys the aforementioned three rules of perturbation analysis. For reference, we list the derivative estimation algorithm in a closed Jackson network. Note that in the algorithm we estimate the derivative directly without estimating each perturbation realization factor. The possible extensions for other general queueing systems are similar.

Algorithm 1. Estimation algorithm for performance derivatives with respect to service rate of server k in a state-independent closed Jackson network

1. *Initialization:* Set variables $\Delta F = 0$, $\Delta_i = 0$, $i = 1, 2, \ldots, M$;
2. *Perturbation generation:* At the oth service completion time of server k, set $\Delta_k := \Delta_k - \frac{\Delta\mu_k}{\mu_k} s_{k,o}$, $o = 1, 2, \ldots, s_{k,o}$ is the service time of the oth customer at server k;
3. *Perturbation propagation:* If a customer from server i terminates an idle period of server j, set $\Delta_j := \Delta_i$, $i, j = 1, 2, \ldots, M$;
4. *UpdateΔF:* At every service completion time of every server i, $i = 1, 2, \ldots, M$, set $\Delta F := \Delta F + [f(n) - f(n')]\Delta_i$, where n and n' are the system states before and after this service completion time, respectively.
5. Finally, at the lth, $l \gg 1$, service completion time, set $\frac{\Delta F}{\Delta\mu_k l}$ as an estimate of the performance derivative $\frac{d\eta_C}{d\mu_k}$.

Note that because of $\Delta_k := \Delta_k - \frac{\Delta\mu_k}{\mu_k} s_{k,o}$ in Step 2, Δ_k for all k is proportional to $\frac{\Delta\mu_k}{\mu_k}$. Therefore, to simplify the algorithm, we may replace Step 2 and Step 5 by (this may also remove the numerical error due to a small $\Delta\mu_k$).

2. *Perturbation generation:* At the oth service completion time of server k, set $\Delta_k := \Delta_k + s_{k,o}$, $o = 1, 2, \ldots, s_{k,o}$ is the service time of the oth customer at server k;
5. Finally, at the lth, $l \gg 1$, service completion time, set $-\frac{\Delta F}{\mu_k l}$ as an estimate of the performance derivative $\frac{d\eta_C}{d\mu_k}$.

This simplification will also be used in Algorithm 2.

It is obvious that this algorithm can be implemented easily and only a few storage and computation budgets are required compared with the original simulation procedure. Moreover, this algorithm can be implemented online, i.e., there is no need to store the history of sample paths.

Please note, the discussion above is based on a closed Jackson network with state-independent service rates. Actually, PA has also been extended to other queueing systems, such as GI/G/m queues, load-dependent or state-dependent service rates Jackson networks. Moreover, the perturbation may be generated by not only service time, but also arrival time, or even routing probabilities, etc. However, those perturbations can be translated into service time delays with some proper techniques.

For the case of state-dependent service rates, the service rate is denoted as $\mu_{i,\boldsymbol{n}}$, $i = 1, 2, \ldots, M, \boldsymbol{n} \in \mathcal{S}$. The basic principle of PA is the same except that the perturbation propagation rule has some changes. At each customer transition time, since the change of system state will affect all the servers' service time, the perturbation of a server will be partly propagated to all the other servers even though these servers are not idle. The details can be referred to by Cao (1994). Similarly, the performance derivative formula is given as below.

$$\frac{\mu_{i,\boldsymbol{n}}}{\eta_l}\frac{d\eta_C}{d\mu_{i,\boldsymbol{n}}} = -\pi(\boldsymbol{n}) c^{(f)}(\boldsymbol{n}, i). \tag{11}$$

With (10) and (11), it is clear that the performance derivative can be extracted from the information of the current system, it has nothing to do with the perturbed system. For theoretical analysis, we can use (6)–(8) to numerically calculate the value of perturbation realization factors, then we can obtain the numerical value of performance derivatives with (10) or (11). For online estimation, we can estimate perturbation realization factors based on a single sample path under the basic principles of perturbation propagation, and then get the estimate of derivatives. Details can be referred to by Xia et al. (2009). Here we briefly list the

estimation algorithm for perturbation realization factors in a state-dependent closed Jackson network for example.

Algorithm 2. Estimation algorithm for perturbation realization factors of server k in a state-dependent closed Jackson network

1. *Initialization:* Set variables $\Delta(n,i) = 0$, $T(n) = 0$, $\Delta F(n) = 0$, for all $\boldsymbol{n} \in \mathcal{S}$, $i = 1, 2, \ldots, M$;
2. *Perturbation generation:* At the service completion time of every server, denote the system state before the customer transition as n_1. Set $\Delta(n_1,k) := \Delta(n_1,k) + t(n_1)$ and $T(n_1) := T(n_1) + t(n_1)$, where $t(n_1)$ is the period length that the system sojourns at the current state n_1;
3. *Perturbation propagation:* For all $\boldsymbol{n} \in \mathcal{S}$, at each service completion time, set $\Delta(n,j) := [1 - \kappa_j(n_1,n_2)]\Delta(n,i) + \kappa_j(n_1,n_2)\Delta(n,j)$ for all $j = 1, 2, \ldots, M$, where i is the index of server which has this service completion, n_1 and n_2 are the system states before and after this service completion time respectively, and $\kappa_j(\boldsymbol{n}_1, \boldsymbol{n}_2) = \mu_{j,\boldsymbol{n}_1}/\mu_{j,\boldsymbol{n}_2}$;
4. *UpdateΔF:* At the same time of step 3, set $\Delta F(n) := \Delta F(n) + [f(n_1) - f(n_2)]\Delta(n,i)$, for all $\boldsymbol{n} \in \mathcal{S}$.

This process continues until the system has served $l \gg 1$ customers. In step 4, there is a point needs to be noted: At the end of simulation, when a server i finishes the service of the lth customer, we simply set $\Delta F(\boldsymbol{n}) := \Delta F(\boldsymbol{n}) + f(\boldsymbol{n}_1)\Delta(\boldsymbol{n},i)$, for all $\boldsymbol{n} \in \mathcal{S}$. This is because at time T_l, the sample path ends and $\boldsymbol{n}_2$ does not exist. Finally, we obtain $\frac{\Delta F(\boldsymbol{n})}{T(\boldsymbol{n})}$ as an estimate of the perturbation realization factor $c^{(f)}(\boldsymbol{n},k)$, for all $\boldsymbol{n} \in \mathcal{S}$.

In summary, the performance derivative formulas (10) and (11) give a clear explanation from the viewpoint of the perturbation generation and realization in queueing systems. This is the foundation of the traditional PA theory and can be extended to other systems such as Markov systems. With the performance derivatives provided by PA, gradient-based algorithms can be developed to optimize the performance of queueing systems. We can also use the perturbation realization factor to build up the performance difference formula, which is the basis for policy iteration based optimization, see the next two sections.

Lastly, there are two important issues worth mentioning about the gradient estimates with PA. Set $\eta_{l,C} = \frac{F_l}{l}$ in (2), then $\eta_{l,C}$ is a finite length sample-path based estimate. (2) becomes $\eta_C = \lim_{l\to\infty}\eta_{l,C}$, which means that $\eta_{l,C}$ is a strongly consistent estimate of η_C. In Algorithm 1, we may denote $-\frac{\Delta F}{\mu_k l} = \frac{d\eta_{l,C}}{d\mu_k}$, and we can prove that

$$\lim_{l\to\infty}\frac{d\eta_{l,C}}{d\mu_k} = \lim_{l\to\infty}\left\{-\frac{\Delta F}{\mu_k l}\right\} = \frac{d\eta_C}{d\mu_k},$$

i.e., $\frac{d\eta_{l,C}}{d\mu_k}$, or $\{-\frac{\Delta F}{\mu_k l}\}$, is indeed a strongly consistent estimate of the derivative $\frac{d\eta_C}{d\mu_k}$.

Another related issue is that for any finite l, do we have

$$E\left\{\frac{d\eta_{l,C}}{d\mu_k}\right\} = E\left\{-\frac{\Delta F}{\mu_k l}\right\} = \frac{dE\{\eta_{l,C}\}}{d\mu_k}?$$

That is, is $\frac{d\eta_{l,C}}{d\mu_k}$, or $\{-\frac{\Delta F}{\mu_k l}\}$, an unbiased estimate? Compared with the strongly consistency, much more researches have been done in proving the unbiasedness of the PA estimates for various cases since it is formulated in Cao (1985). From the physical meanings, the unbiasedness of PA gradient estimates usually requires that the infinitesimal perturbation of parameters will not change the occurrence order of the series of events (state transitions) on the original sample path, or the so called "commuting condition" (however, it is neither necessary nor sufficient). More details can be found in Cao (1985, 2007), Heidelberger et al. (1989), and Glasserman (1991). Next, we should note that the estimate with realization factors is usually strongly consistent, because each realization factor already contains the information of event order change. As seen in the next section, this statement also holds for perturbations with finite sizes.

3. Sensitivity-based optimization of Markov systems

During the last decade, PA theory has been further developed for Markov systems (Cao, 2003; Cao and Chen, 1997). Since many queueing systems (for example, M/M/1 queues and Jackson networks) can be formulated as Markov processes, we may apply the results in PA of Markov systems to queueing systems together with the special feature of queueing systems to optimize their performance. Furthermore, this approach aims to optimize the time-average performance (cf. (1)) of queueing systems, which is different from the customer-average performance (cf. (2)) in the traditional PA theory of queueing systems.

Similar to the PA of queueing systems, PA of Markov systems decomposes the derivative performance of a Markov system with respect to the changes in its transition probability matrix into the weighted sum of perturbation realization factors (cf. (10)). Because a perturbation realization factor in Markov systems equals the difference of performance potentials (refer to its definition (13)), performance potentials become the building blocks of performance sensitivities (cf. (19)). There are two types of performance sensitivities, performance derivatives and performance difference. Performance derivative formula provides the basis for the gradient-based optimization, and performance difference formula provides the basis for the policy iteration based optimization. Online learning algorithms can be further developed to implement the sensitivity-based optimization approach. We will call both gradient-based and policy iteration based approaches the sensitivity-based optimization for Markov systems.

We use a discrete time ergodic Markov chain as an example to overview the main results of sensitivity-based optimization theory of Markov systems (it has also been extended to continuous time Markov systems and multi-chain Markov systems (Zhang and Cao, 2009)). Consider an ergodic Markov chain $\boldsymbol{X} = \{X_0, X_1, X_2, \ldots\}$ where X_t is the system state at time t, $t = 0, 1, 2, \ldots$ The finite state space is denoted as $\mathcal{S} = \{1, 2, \ldots, S\}$ and $X_t \in \mathcal{S}$. The transition probability matrix is denoted as $P = [p(v|u)]_{u,v=1}^{S}$ with $Pe = e$, where e is an S-dimension column vector whose elements are all 1. The steady-state probability is denoted as a row vector $\pi = (\pi(1), \pi(2), \ldots, \pi(S))$ and it has $\pi e = 1$. Obviously, from the flow balance of steady-state probability we have $\pi P = \pi$. The system cost (reward) function is denoted as a column vector $f = (f(1), f(2), \ldots, f(S))^T$, where the superscript T indicates the transpose operation. The time-average performance is

$$\eta_T = E\{f(X_t)\} = \sum_{u=1}^{S}\pi(u)f(u) = \pi f = \lim_{l\to\infty}\frac{1}{l}\sum_{t=0}^{l-1}f(X_t), \tag{12}$$

where E denotes the expectation over steady-state distribution. Since a policy determines both a transition probability matrix P and a cost function f, for simplicity, we use (P,f) to represent a policy in Markov systems. The optimization problem is to choose (P,f) to make the corresponding performance η_T optimal.

One of the most fundamental concepts in the sensitivity-based approach of Markov systems is the performance potential, which measures the contribution of a state to the total system performance η_T. The *performance potential* g is defined as a column vector whose element $g(u), u \in \mathcal{S}$, is

$$g(u) = \lim_{T\to\infty} E\left\{\sum_{t=0}^{T}[f(X_t) - \eta_T]|X_0 = u\right\}. \tag{13}$$

It is also called bias or relative value function in the MDP theory (Puterman, 1994).

By decomposing the right-hand side of (13) into the terms at $t = 0$ and those with $t \geqslant 1$, we have

$$g(u) = f(u) - \eta_T + \sum_{v=1}^{S} p(v|u) \lim_{T\to\infty} E\left\{\sum_{t=1}^{T}[f(X_t) - \eta_T]|X_1 = v\right\}$$
$$= f(u) - \eta_T + \sum_{v=1}^{S} p(v|u) g(v). \quad (14)$$

In a matrix form, this is the *Poisson equation*

$$(I - P)g = f - \eta_T e. \quad (15)$$

The perturbation realization factor in PA theory of Markov systems is defined as the difference between the performance potentials of two states. Therefore, all the perturbation realization factors form a matrix $D = [d(u, v)]_{u,v=1}^{S}$ and its element $d(u, v)$ is defined as below.

$$d(u, v) = g(v) - g(u)$$
$$= \lim_{T\to\infty} E\left\{\sum_{t=0}^{T}[f(X'_t) - f(X_t)]|X_0 = u, X'_0 = v\right\}, \quad (16)$$

where X_t and X'_t are two sample paths with the initial state u and v, respectively. From the definition, $d(u, v)$ measures the effect on the long term system performance if the initial state is perturbed from u to v.

We can derive the performance sensitivity formulas based on the performance potentials. Suppose that the transition probability matrix changes from P to P' and the corresponding cost function changes from f to f'. The steady-state probability of the perturbed system with P' is denoted as π' and the corresponding system performance is $\eta'_T = \pi' f'$. Multiplying both sides of (15) with π', we have

$$\pi'(I - P)g = \pi' f - \eta_T = \pi'(f - f') + \pi' f - \eta_T$$
$$= \pi'(f - f') + \eta'_T - \eta_T. \quad (17)$$

Denoting $\Delta P = P' - P$ and $h = f' - f$, we further obtain the following *performance difference formula*

$$\eta'_T - \eta_T = \pi'(P' - P)g + \pi' h = \pi'(\Delta P g + h), \quad (18)$$

where the equality $\pi' P' = \pi'$ is used. We can use (18) to guide the optimization of Markov systems. Note that P, P', f, and f' are known parameters. If we obtain the value of performance potential g of the current system, we can try to choose (if possible) a proper (P', f') to make the value of all the elements in the bracket of the right-hand side of (18) non-positive with at least one negative. Since the elements of π' are always positive for ergodic Markov chains, we have $\eta'_T < \eta_T$ and the system performance is improved for the minimization problem. This is exactly the basic idea of policy iteration in MDP: from an initial policy, we analyze it to obtain its potentials and find a better policy with the performance difference formula, then we analyze this better policy and get an even better policy, this process stops when an optimal policy is obtained. The detailed algorithm of policy iteration can be found in Cao (2007) and Puterman (1994). The optimality (H–J–B) equation follows directly from the performance difference formula (18) (Cao, 2007).

From (18), we can see that performance potential g is a fundamental quantity in policy iteration. Besides the numerical calculation of Poisson equation to obtain the value of g (Cao, 2007), we can also estimate it based on a sample path of system. We can analyze the system behavior under a current policy (P, f) to estimate g and further to determine a better policy (P', f') based on (18). This is also called the online learning and optimization for Markov systems and the detailed estimation algorithms can be found in Cao (2007). Moreover, it is known that policy iteration requires the action at each state should be chosen independently (Puterman, 1994). If the independent requirement is not satisfied, policy iteration may be not applicable. In such situation, policy gradient-based optimization may be a feasible way and it is introduced as follows.

Consider a random policy which adopts P with probability $1 - \delta$ and adopts P' with probability δ, where $0 < \delta < 1$. Obviously, the transition probability matrix under this random policy is $P^\delta = P(1 - \delta) + P'\delta = P + \delta\Delta P$. When $\delta \ll 1$, the cost function under this random policy is denoted as $f^\delta = f + \delta h$, where $h = f' - f$. The time average performance of this system with (P^δ, f^δ) is denoted as η_T^δ. Replacing P' and f' with P^δ and f^δ in (18) and making $\delta \to 0$, we get the derivative of η_T with respect to δ as follows.

$$\frac{d\eta_T}{d\delta} = \lim_{\delta\to 0} \frac{\eta_T^\delta - \eta_T}{\delta} = \pi(\Delta P g + h). \quad (19)$$

This is called the *performance derivative formula* in PA of Markov systems. With the performance derivative, policy-gradient based approach can be developed for the continuous parameters optimization problem of Markov systems (Cao, 2007; Marbach and Tsitsiklis, 2001).

Historically, this derivative formula is originally constructed based on sample paths with an approach similar to the derivation of (10) (Cao et al., 1996), following the same principle as illustrated in Fig. 1. The performance difference formula (18) can also be intuitively constructed by decomposition based on perturbations (Cao and Chen, 1997).

Performance difference formula (18) and performance derivative formula (19) are the two basic formulas in the sensitivity-based optimization of Markov systems. With (18), we may derive policy iteration based algorithms; and with (19), we may derive policy-gradient optimization algorithms (see Fig. 5). It is obvious that the performance difference can provide more information than the performance derivative. In general, policy iteration is more efficient than the gradient-based optimization, since policy iteration jumps in the total search space to find a better (possibly much better) solution, while gradient-based optimization can only move along the gradient-descent direction with a small step-size (small improvement); and it may be trapped into a local optimum. Therefore, policy iteration is usually preferred than the policy-gradient optimization, except for some situations where policy iteration is not applicable (e.g., policy iteration requires the actions at different states be independent (Cao and Chen, 1997; Puterman, 1994)). On the other hand, gradient-based approach is usually applicable to more systems.

Since many queueing systems can be modeled by Markov processes, we may apply the sensitivity-based optimization of Markov systems to queueing systems. For example, in a closed Jackson network, if we want to control the service rates of servers to minimize the queue length of server i, we can set the cost function as $f(\boldsymbol{n}) = n_i$. Then η_T is exactly the average queue length of server i and we can

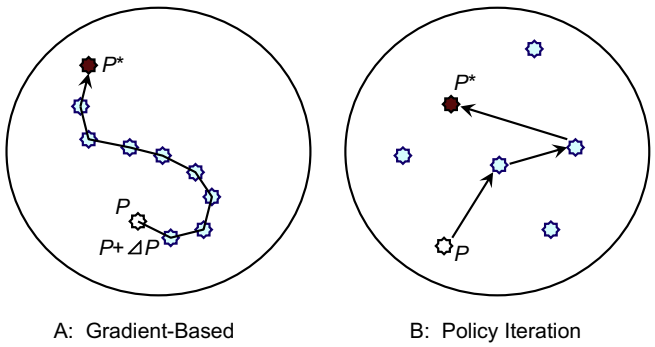

Fig. 5. Two kinds of sensitivity-based optimization methods.

use (18) to minimize η_T. Please note, this methodology is not applicable to the customer-average performance, such as the customer waiting time in queueing systems. For such performance metrics, we may resort to the approach in the next section. Moreover, there are other approaches, such as the RG-factorization approach (Li, 2010) for performance optimization, which will not be covered in this review paper.

In summary, with PA of queueing systems reviewed in Section 2, we get a gradient-based approach to the optimization of the customer-average performance with continuous parameters of queueing systems. With the approach reviewed in this section, we may derive the policy iteration to the optimization of the time-average performance with discrete policy space of queueing systems. In the next section, we will review how to develop the policy iteration approach to optimize the customer-average performance of queueing systems, based on the special queueing structures.

4. Extended perturbation analysis of queueing systems

The traditional PA of queueing systems focuses on how to efficiently obtain the performance derivatives with respect to parameters based on a single sample path of queueing systems, while the sensitivity-based approach of Markov systems focuses on extracting both the performance difference and performance derivative information based on a sample path of Markov systems.

Both these two approaches have the same basic idea: analyzing the effect of a single perturbation on the system performance to obtain the performance sensitivity information. The fundamental quantity is the perturbation realization factor (or performance potential), which measures the total effect of a perturbation on the system performance and can be used as building blocks for the two types of sensitivity formulas.

Since a queueing system with Markovian property can be modeled as a Markov process, the perturbation realization factors when it is viewed as a queueing system, $c^{(f)}(\boldsymbol{n},i)$, and those when it is modeled as a Markov system, $d(u,v)$, must have some relationship. With such relationship as a bridge, it is possible to link the approaches in Sections 2 and 3 together and many results with these two approaches may be extended to each other in parallel. For example, it is promising to derive the difference formula for the customer-average performance in queueing systems since we already have the difference formula for the time-average performance in Markov systems. With such extension, it may provide a new way to optimize the customer-average performance of queueing systems using policy iteration based algorithms.

However, the perturbation in queueing systems is an infinitesimal delay of service time, which is a continuous quantity; and the perturbation in Markov systems is a jump of system states, which is discrete in nature. These two kinds of perturbations are totally different and it is not straightforward to establish a relationship between them.

Recent study (Xia and Cao, 2006a) shows that such relationship does exist. Consider a closed Jackson network with M servers and N customers. Suppose that the service time of server i at state $\boldsymbol{n}$ gets an infinitesimal delay Δ at time $t = 0$. By PA of queueing systems, the average effect of this service time delay Δ can be quantified by $c^{(f)}(\boldsymbol{n},i)\Delta$, which is defined in (5). On the other hand, by the sensitivity-based optimization of Markov systems, such a service time delay will make the perturbed sample path $\boldsymbol{n}'(t)$ different from the original sample path $\boldsymbol{n}(t)$: by definition of a service delay, during the small period $[0,\Delta)$, server i on $\boldsymbol{n}'(t)$ cannot have any service completion, while server i on the original $\boldsymbol{n}(t)$ may have service completion with probability $\mu_{i,\boldsymbol{n}}\Delta$. After this period, this two sample paths will have exactly the same transition probabilities and random factors, since the Markov model has the memoryless property. But the initial state distribution for $\boldsymbol{n}'(t)$ and $\boldsymbol{n}(t)$ is different at time $t = \Delta$ because of the aforementioned difference between this two sample paths. Such a difference in initial states equals an initial state jump whose effect is quantified by the perturbation realization factor $d(u,v)$ in (16). Therefore, we can use $d(u,v)$ of Markov systems to measure the effect of such service time perturbation. This is the basic idea used by Xia and Cao (2006a) to derive the relationship between $c^{(f)}(\boldsymbol{n},i)$ and $d(u,v)$. With detailed derivations in probability, the following relationship is obtained (Xia and Cao, 2006a)

$$c^{(f)}(\boldsymbol{n},i) - c(\boldsymbol{n},i)\eta_T = \mathbf{1}(n_i)\mu_{i,\boldsymbol{n}}\sum_{j=1}^{M} q_{ij}d(\boldsymbol{n}_{ij},\boldsymbol{n}). \tag{20}$$

The left-hand side of (20) quantifies the effect of perturbations from PA of queueing systems; the right-hand side of (20) quantifies the effect of perturbations due to state jumps of server i in a time unit, which is a viewpoint from PA of Markov systems.

The similar relationship formula for open Jackson networks is also derived by Xia and Cao (2006a). With the relationship formula (20), we may link the PA of queueing systems and the PA and policy iteration of Markov systems together, and the results in one system may be extended to the other. Therefore, it enriches the results of PA theory and opens a new research direction. For example, one of the direct consequence is that we can extend the difference formula in Markov systems to queueing systems, and with this difference formula we can develop policy iteration to optimize the customer-average performance of queueing systems.

As we know, time-average and customer-average are two important performance metrics in queueing systems. The first metric measures the system performance averaged over time as defined in (1), while the second one measures the system performance averaged over the number of served customers as defined in (2). These two metrics are naturally different and they describe the different aspects of system behaviors. The relationship between these two metrics is given by (3). For a clearer explanation, we use an M/M/1 queue for example. A typical customer-average performance is the (customer) average response time W, while the average queue length L is actually a time-average performance. Obviously, these two metrics are totally different and both are important for a queueing system. In this case, when $f(n) = n$, the relationship (3) becomes $W = L/\lambda$, where λ is the arrival rate of the queue and it also equals the throughput η_{th}. Therefore, this relationship is exactly *the Little's Law*.

In many situations, the optimization results for time-average η_T and customer-average η_C are generally different. It can be partially explained from their relationship formula (3). For example, in the service rate control problem of a closed Jackson network, the change of service rates will obviously affect the throughput of the network η_{th}, so the optimal service rates for η_T and η_C are usually different (Xia et al., 2009). Therefore, it is necessary to develop different optimization algorithms under different performance criteria.

For some (not all) queueing systems with Markovian property, such as the load-dependent Jackson networks, there exist some numerical methods to calculate the performance (e.g., Buzen algorithm). However, one still has to resort to optimization approaches, such as the gradient method or the exhaustive search method to obtain the optimal parameters. Therefore, it is meaningful to develop other efficient optimization methods for Markovian queueing systems, and such new methods shed new insights and may be extended to more non-Markovian systems. In addition, new analytical results can be obtained from these new methods and new insights, see Xia et al. (2009), some of them are reviewed below.

Consider a closed Jackson network with state-dependent service rates. The network parameters are the same as those in

Section 2. We further denote $\vec{\mu}_{\boldsymbol{n}} = \{\mu_{i,\boldsymbol{n}},\ i = 1, 2, \ldots, M\}$ as the service rates of all servers at state $\boldsymbol{n}$. The cost function depends on the service rate and is denoted as $f(\boldsymbol{n}, \vec{\mu}_{\boldsymbol{n}}),\ \boldsymbol{n} \in \mathcal{S}$. Suppose that the service rate of one server i at one state $\boldsymbol{n}$ is changed from $\mu_{i,\boldsymbol{n}}$ to $\mu'_{i,\boldsymbol{n}}$. We consider its effect on the customer-average performance η_C. Since a closed Jackson network can also be modeled as a Markov process, the specific difference of η_T can be obtained by (18). As the $d(u, v)$ and $c^{(f)}(\boldsymbol{n}, i)$ have the relationship (20), we can use $c^{(f)}(\boldsymbol{n}, i)$ to replace g in (18). With the specific values of P for closed Jackson networks, we rewrite the difference formula (18) as below.

$$\eta'_T - \eta_T = -\pi'(\boldsymbol{n})\frac{\Delta\mu_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}}[c^{(f)}(\boldsymbol{n}, i) - c(\boldsymbol{n}, i)\eta_T] + \pi'(\boldsymbol{n})h(\boldsymbol{n}), \tag{21}$$

where $\Delta\mu_{i,\boldsymbol{n}} = \mu'_{i,\boldsymbol{n}} - \mu_{i,\boldsymbol{n}}$ and $h(\boldsymbol{n}) = f(\boldsymbol{n}, \vec{\mu}'_{\boldsymbol{n}}) - f(\boldsymbol{n}, \vec{\mu}_{\boldsymbol{n}})$.

Since the throughput η_{th} (the reciprocal of η_l) can be viewed as a special case of time-average performance, we can similarly obtain the following difference formula for η_l

$$\eta'_l - \eta_l = -\eta'_l\pi'(\boldsymbol{n})\frac{\Delta\mu_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}}c(\boldsymbol{n}, i), \tag{22}$$

where some special properties of perturbation realization factors are used to simplify this difference formula (Xia et al., 2009). Based on (3), the difference of η_C can be written as

$$\eta'_C - \eta_C = \eta'_l(\eta'_T - \eta_T) + \eta_T(\eta'_l - \eta_l). \tag{23}$$

Substituting (21) and (22) into (23), we obtain the following difference formula for customer-average performance

$$\eta'_C - \eta_C = \eta'_l\pi'(\boldsymbol{n})\left\{\frac{-\Delta\mu_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}}c^{(f)}(\boldsymbol{n}, i) + h(\boldsymbol{n})\right\}. \tag{24}$$

When the service rates of server i at all states $\boldsymbol{n}$ are changed from $\mu_{i,\boldsymbol{n}}$ to $\mu'_{i,\boldsymbol{n}}, \boldsymbol{n} \in \mathcal{S}$, the difference formula for η_C is obtained as below with a similar analysis.

$$\eta'_C - \eta_C = \eta'_l\sum_{\boldsymbol{n}\in\mathcal{S}}\pi'(\boldsymbol{n})\left\{\frac{-\Delta\mu_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}}c^{(f)}(\boldsymbol{n}, i) + h(\boldsymbol{n})\right\}. \tag{25}$$

Furthermore, when the service rates of all servers i at all states $\boldsymbol{n}$ are changed from $\mu_{i,\boldsymbol{n}}$ to $\mu'_{i,\boldsymbol{n}},\ i = 1, 2, \ldots, M,\ \boldsymbol{n} \in \mathcal{S}$, the difference formula will have the following form after some transformations with (7)

$$\eta'_C - \eta_C = \eta'_l\sum_{\boldsymbol{n}\in\mathcal{S}}\pi'(\boldsymbol{n})\left\{f(\boldsymbol{n}, \vec{\mu}'_{\boldsymbol{n}}) - \sum_{i=1}^{M}\frac{\mu'_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}}c^{(f)}(\boldsymbol{n}, i)\right\}. \tag{26}$$

Now, let us consider the service rate control problem in a state-dependent closed Jackson network. Denote $\mathcal{L} = \{\vec{\mu}_{\boldsymbol{n}}, \boldsymbol{n} \in \mathcal{S}\}$ as a policy. Suppose that the feasible values of service rates are discrete and all the possible $\mathcal{L}$'s form a policy space Ψ, that is $\Psi = \{all\ \mathcal{L}\}$.

From (26), we can easily derive the optimality (H–J–B) equation: a set of service rates $\vec{\mu}^*_{\boldsymbol{n}} = \{\mu^*_{i,\boldsymbol{n}},\ i = 1, 2, \ldots, M\}, \boldsymbol{n} \in \mathcal{S}$, is optimal (for minimization problem) if and only if it satisfies the optimality (H–J–B) equation

$$\min_{all\vec{\mu}_{\boldsymbol{n}}}\left\{f(\boldsymbol{n}, \vec{\mu}_{\boldsymbol{n}}) - \sum_{i=1}^{M}\frac{\mu_{i,\boldsymbol{n}}}{\mu^*_{i,\boldsymbol{n}}}c^{(f)}(\boldsymbol{n}, i)^*\right\} = 0, \quad \text{for all } \boldsymbol{n} \in \mathcal{S}, \tag{27}$$

where $c^{(f)}(\boldsymbol{n}, i)^*$ is the perturbation realization factors corresponding to the policy $\mathcal{L}^* = \{\vec{\mu}^*_{\boldsymbol{n}},\ \boldsymbol{n} \in \mathcal{S}\}$.

Based on the difference formula (26) and the optimality equation (27), we have the following policy iteration algorithm for minimizing the customer-average performance of queueing systems.

Algorithm 3. Policy iteration algorithm for optimization of customer-average performance of state-dependent closed Jackson networks

1. *Initialization:* Choose an arbitrary policy $\mathcal{L}_0 \in \Psi$ as an initial policy, and set $k = 0$.
2. *Evaluation:* At the kth iteration with the policy denoted as $\mathcal{L}_k = \{\vec{\mu}_{\boldsymbol{n}}, \boldsymbol{n} \in \mathcal{S}\}$, calculate or estimate the perturbation realization factors $c^{(f)}(\boldsymbol{n}, i),\ i = 1, 2, \ldots, M,\ \boldsymbol{n} \in \mathcal{S}$, under policy $\mathcal{L}_k$.
3. *Improvement:* Choose the policy of the next ($k + 1$) th) iteration as $\mathcal{L}_{k+1} = \{\vec{\mu}'_{\boldsymbol{n}},\ \boldsymbol{n} \in \mathcal{S}\}$ with

$$\vec{\mu}'_{\boldsymbol{n}} = \arg\min_{\vec{\mu}'_{\boldsymbol{n}}}\left\{f(\boldsymbol{n}, \vec{\mu}'_{\boldsymbol{n}}) - \sum_{i=1}^{M}\frac{\mu'_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}}c^{(f)}(\boldsymbol{n}, i)\right\}, \boldsymbol{n} \in \mathcal{S}. \tag{28}$$

If at a state $\boldsymbol{n}$, $\vec{\mu}_{\boldsymbol{n}}$ already reaches the minimum of the large bracket above, then set $\vec{\mu}'_{\boldsymbol{n}} = \vec{\mu}_{\boldsymbol{n}}$.

4. *Stopping Rule:* If $\mathcal{L}_{k+1} = \mathcal{L}_k$, stop; otherwise, set $k = k + 1$ and go to step 2.

Please note that $\pi'(\boldsymbol{n})$ is not needed in every iteration. This is because $\pi'(\boldsymbol{n})$ is always positive in a strongly connected queueing system and we only need to minimize the value of big brackets in (26). This is a crucial advantage because if $\pi'(\boldsymbol{n})$ is needed then it may become exhaustive search. In general, policy iteration algorithm is more efficient than the gradient-based algorithm, just like what Fig. 5 illustrates.

It is worth mentioning that step 3 is actually a sub-optimization problem. However, in this sub-optimization problem there is only M variables, and while the original problem has $|\mathcal{S}| \times M$ variables. This greatly reduces the computation. Moreover, the cost function $f(\boldsymbol{n}, \vec{\mu}_{\boldsymbol{n}})$ usually satisfies some special conditions, e.g., $f(\boldsymbol{n}, \vec{\mu}_{\boldsymbol{n}})$ may be a linear function of $\mu_{i,\boldsymbol{n}}$; in this case, (28) is simply a linear programming problem of M variables. Furthermore, to implement policy iteration, it is even not necessary to solve the sub-optimization problem (28); in fact, we may use any $\vec{\mu}'_{\boldsymbol{n}}$ with $f(\boldsymbol{n}, \vec{\mu}'_{\boldsymbol{n}}) - \sum_{i=1}^{M}\frac{\mu'_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}}c^{(f)}(\boldsymbol{n}, i) < 0$ as the policy in the next step.

In many situations, we may discretize $\mu_{i,\boldsymbol{n}}$ into a number of discrete values to make the problem a discrete one. In this case, this sub-optimization problem is much simpler compared to the original problem, because the state $\boldsymbol{n}$ is fixed and we only need to enumerate all possible (a small number) $\mu_{i,\boldsymbol{n}}$, $i = 1, 2, \ldots, M$, to get the best one (Xia et al., 2009). The total search space of the optimization algorithm is greatly reduced through such decompositions (from $|A|^{|\mathcal{S}|\times M}$ to $|\mathcal{S}| \times |A|^M$, where $|A|$ is the number of possible discrete values of each $\mu_{i,\boldsymbol{n}}$). This is one of the key points of policy iteration method. It is worth pointing out that although this is a great reduction, the problem is not solved completely since $|\mathcal{S}| \times |A|^M$ is still large especially for large-scale problems. Learning methods and other approximation methods should be developed (Cao, 2007). Therefore, Algorithm 3 gives a new way to optimize the customer-average performance of queueing systems, additional to the traditional gradient-based PA theory. Along with the derivative formulas (10) and (11), the difference formulas make the performance optimization of queueing systems with PA theory more complete.

In order to implement the policy iteration, it requires to obtain the perturbation realization factors $c^{(f)}(\boldsymbol{n}, i)$. As explained in Section 2, $c^{(f)}(\boldsymbol{n}, i)$ can be estimated by Algorithm 2. Therefore, the policy iteration algorithm above can be online implemented based on the estimates of $c^{(f)}(\boldsymbol{n}, i)$ from sample paths. The procedure can be understood as follows. Observing the system behavior from sample paths, we extract the related information to obtain the value of $c^{(f)}(\boldsymbol{n}, i)$. Then we can use the difference formula to get a better policy and improve the system performance; the difference formula (or the improvement scheme) is derived by utilizing the special properties of queueing systems. Continue this procedure until the policy cannot be improved anymore. In words, we efficiently extract the sensitivity information from sample paths and increasingly improve the system performance based on the structure of

queueing systems. This can also be viewed as a learning and improving procedure.

This performance optimization theory may be applied to other queueing systems. For example, consider a closed Jackson network where the service rate of a server only depends on the queue length at this server, which is called load-dependent service rate $\mu_{i,n_i},\ i=1,2,\ldots,M,\ n_i=1,2,\ldots,N$. Utilizing the property of product-form solution in closed Jackson networks, we can get the following difference formula when the service rates of a server i change from μ_{i,n_i} to μ'_{i,n_i} (Xia and Cao, 2006b)

$$\eta'_C-\eta_C=\eta'_l\sum_{n_i=1}^{N}\pi'(n_i)\left\{\frac{-\Delta\mu_{i,n_i}}{\mu_{i,n_i}}\tilde{c}^{(f)}(n_i,i)+\tilde{h}(n_i,i)\right\}, \tag{29}$$

where $\tilde{c}^{(f)}(n_i,i)$ is called the aggregated perturbation realization factor which is the average of $c^{(f)}(\boldsymbol{n},i)$ weighted by the distribution of $\pi(\boldsymbol{n}|n_i)$ and its definition is

$$\tilde{c}^{(f)}(n_i,i)=\sum_{\boldsymbol{n}\in\mathcal{S}_{n_i}}\pi(\boldsymbol{n}|n_i)c^{(f)}(\boldsymbol{n},i), \tag{30}$$

where $\mathcal{S}_{n_i}$ is a subset of state space where the queue length of server i equals n_i, and $\tilde{h}(n_i,i)=\sum_{\boldsymbol{n}\in\mathcal{S}_{n_i}}\pi(\boldsymbol{n}|n_i)h(\boldsymbol{n})$ is the aggregated difference of cost functions. Similarly, we can estimate the aggregated perturbation realization factors based on a sample path of system and develop the policy iteration to optimize the service rates of server i.

In order to demonstrate how to use the gradient-based and policy iteration based optimization for queueing systems, we give a numerical example. Consider a state-dependent closed Jackson network with $M=3$ and $N=5$. The routing probability matrix is $Q=[0,0.7,0.3;0.4,0,0.6;0.8,0.2,0]$. The cost function is $f(\boldsymbol{n})=n_1+\sum_{i=1}^{3}\mu_{i,\boldsymbol{n}}$, where the first term of cost is the holding cost which reflects the (customer) average response time at server 1 and the second term is the operating cost for providing certain service rates. The value domains of all service rates are assumed to be identical as [0.1,3]. The objective is to minimize the customer-average performance η_C by adjusting the service rates $\mu_{i,\boldsymbol{n}}$, $i=1,2,3$ and $\boldsymbol{n}\in\mathcal{S}$. For simplicity, the initial service rates $\mu_{i,\boldsymbol{n}}$ are all set as 0.1. As $f(\boldsymbol{n})$ is a linear function of $\mu_{i,\boldsymbol{n}}$, it is proved that the optimal values of service rates can be either maximum or minimum of the corresponding feasible values (Ma and Cao, 1994; Xia and Shihada, submitted for publication; Yao and Schechner, 1989). Thus, the number of feasible values of each service rate is only 2. The size of the policy space is $2^{|\mathcal{S}|\times M}=2^{21\times3}\approx10^{19}$, which is still very large.

First, Algorithm 1 is used to estimate the performance gradient with respect to service rates. Then gradient-based optimization is applied to update the service rates as: $\mu_{i,\boldsymbol{n}}^{(k+1)}=\mu_{i,\boldsymbol{n}}^{(k)}-\gamma\times\nabla_{\mu_{i,\boldsymbol{n}}}\eta_C$, where $\mu_{i,\boldsymbol{n}}^{(k)}$ is the service rate at the kth iteration, $i=1,2,3,\ \boldsymbol{n}\in\mathcal{S}$, γ is the step-size and set as 0.1, and $\nabla_{\mu_{i,\boldsymbol{n}}}\eta_C$ is the gradient estimated by Algorithm 1. The first *20* iterations of gradient-based optimization are illustrated by the stars in Fig. 6 and we can see that the convergence rate is slow. How to choose the step-size is a key problem of gradient-based approaches and heavily affects the convergence rate. Choosing a decreasing step size may improve the convergence speeds, and this example simply serves for an illustrative purpose.

As a comparison, Algorithm 3 is used to demonstrate the efficiency of policy iteration approach. As illustrated by the simulation result in Fig. 6, the algorithm only iterates 4 times to reach the stopping criterion. This shows that Algorithm 3 converges fast and it over-performs the gradient-based approach Algorithm 1, as shown by Fig. 5. The optimal service rates obtained by Algorithm 3 are listed in Table 1.

It is worth pointing out again that the approach in this section focuses on the customer-average performance. This metric cannot be optimized by the traditional MDP method which focuses on the time-average performance. The approach in this section is based on the difference formulas and perturbation realization factors in queueing systems, and it provides another new perspective to study the performance optimization of queueing systems.

In summary, with the relationship formula (20), PA of queueing systems and PA of Markov systems are linked together. The performance difference formula and policy iteration in Markov systems are extended to queueing systems for the optimization of the customer-average performance. Together with the performance derivative formula of traditional PA, this completes the PA theory of queueing systems. The performance derivative formulas (10) and (11), and the performance difference formulas (24)–(26) are the basis for performance optimization of queueing systems. Based on performance derivatives, gradient-based algorithm can be developed. Based on performance differences, policy iteration algorithm can be developed, and the optimality (H–J–B) equation (27) can be derived using the perturbation realization factors.

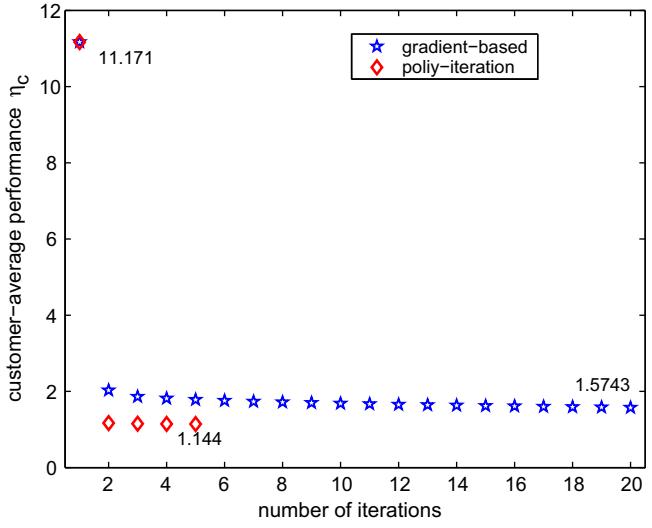

Fig. 6. Comparison of optimization approaches with gradient-based and policy iteration.

Table 1
The optimal service rates for the minimal customer-average performance.

n	(0,0,5)	(0,1,4)	(0,2,3)	(0,3,2)	(0,4,1)	(0,5,0)	(1,0,4)	(1,1,3)	(1,2,2)	(1,3,1)
$\mu^*_{1,n}$	0.1	0.1	0.1	0.1	0.1	0.1	3.0	3.0	3.0	3.0
$\mu^*_{2,n}$	0.1	0.1	3.0	3.0	3.0	3.0	0.1	0.1	0.1	0.1
$\mu^*_{3,n}$	3.0	3.0	3.0	3.0	0.1	0.1	0.1	0.1	0.1	0.1
n	(1,4,0)	(2,0,3)	(2,1,2)	(2,2,1)	(2,3,0)	(3,0,2)	(3,1,1)	(3,2,0)	(4,0,1)	(4,1,0)	(5,0,0)
$\mu^*_{1,n}$	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
$\mu^*_{2,n}$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
$\mu^*_{3,n}$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

With the sample path based estimation algorithm of perturbation realization factors, the optimization algorithms can be implemented online without interrupting the operation of queueing systems. However, the approach in this section currently only deals with the service rate control problem. There is no literature studying the difference formula and policy iteration of customer-average performance when other parameters have changes. How to extend this approach to optimize other parameters, such as routing probabilities, buffer sizes, is still a future research topic. Other new research topics in the framework of Markov systems, such as the event-based optimization (Cao, 2007), value function approximation and aggregation (Bertsekas and Tsitsiklis, 1996; Marbach et al., 2000), may also be extended to queueing systems as future research directions.

5. Other perturbation analysis related approaches

During the history of development of PA theory in queueing systems, many other variants of PA are proposed to deal with the unbiasedness issue for some special problems. In this section, we introduce the finite PA, smoothed PA, and PA with stochastic fluid model. Other research topics about PA, such as the rare PA, structural PA, and PA applications to various scenarios, are widely included in the books (Cao, 1994; Fu and Hu, 1997; Glasserman, 1991; Ho and Cao, 1991) and we will not discuss them in this paper.

As reviewed in Section 2, the traditional PA of queueing systems mostly requires that the perturbation of parameters be infinitesimal. This kind of PA theory is also called IPA (Infinitesimal PA). However, in some cases the parameter perturbations are not infinitesimal. For example, the average service time of a server may increase by 20 percent amount, or the perturbed parameters and their changes are discrete (e.g., buffer size). Such perturbations may change the event occurrence orders compared to the nominal ones and IPA is not applicable (Cao, 1985; Heidelberger et al., 1989). To address this type of problems, finite PA (FPA) (Heidergott, 2000; Ho et al., 1983) is developed. Because the perturbations are not small enough, it may occur that some original idle periods of servers are eliminated or some new idle periods are created compared to the original sample path. Therefore, the perturbation propagation rule in FPA is different from that in Section 2 and it can be much more complicated. With the new version of propagation rule, FPA algorithm may provide a rough estimate of performance sensitivity for finite perturbations based on a single sample path. More accurate FPA estimates require even more complicated propagation rules and this makes the implementation of FPA too difficult in practice.

Smoothed PA (called SPA (Fu and Hu, 1994; Glasserman and Gong, 1990; Gong and Ho, 1987; Heidergott, 1998)) is proposed to handle the problem that the sample performance function $L(\theta,\xi)$ may be discontinuous with respect to parameters θ, where ξ represents the random factors of the sample path. This discontinuity may make IPA estimates biased (Cao, 1985; Heidelberger et al., 1989). The key idea of SPA is to find a random characterization $z(\theta,\xi)$ which can make the conditional performance $E_\xi\{L(\theta,\xi)|z\}$ be a continuous function of θ. In other words, the characterization z can "smooth-out" the discontinuity of the sample performance function $L(\theta,\xi)|z$ and make it continuous. Similar techniques of IPA can be used to estimate the performance gradient under this condition z. Then, we can average the estimates above under the distribution of z to get the final estimate for the performance derivative. SPA extends the applicability of PA theory. The disadvantage of SPA is that finding a proper characterization z heavily depends on the specific problem and the user's experience, and this limits its application.

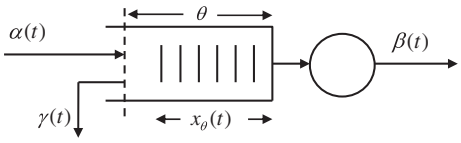

Fig. 7. The stochastic fluid model for a single queue with a limited buffer.

PA method with stochastic fluid model (SFM) applies PA to a wider situation besides the classical queueing systems. The high-rate communication network has events occurring heavily frequently (packets arrival or departure) and it will consume large computation resources under the simulation framework of discrete event systems. Actually, the dynamics of traffic flow can be approximated well as a flow of fluid and SFM can be used to model such kind of systems. SFM ignores the dynamics of individual customers and treats the system state as continuous variables. Therefore, the complex dynamics of discrete states are approximately represented by the evolution of continuous states. This is a much simpler model than the originally rigorous queueing model.

Fig. 7 is an example of SFM for a single queue with buffer size limit θ. $\alpha(t)$ is the inflow rate of customers and $\beta(t)$ is the outflow rate of the server. $x_\theta(t)$ is the queue length of the server and $\gamma(t)$ is the overflow rate when $x_\theta(t)$ reaches the buffer size limit θ. Therefore, the dynamics of this queue can be formulated as the following differential equation

$$\frac{dx_\theta(t)}{dt}=\begin{cases}0, & \text{if } x_\theta(t)=0 \quad \text{and } \alpha(t)\leqslant\beta(t),\\ 0, & \text{if} x_\theta(t)=\theta \quad \text{and } \alpha(t)\geqslant\beta(t),\\ \alpha(t)-\beta(t), & \text{otherwise}\end{cases} \tag{31}$$

The overflow rate can be obviously represented as, if $x_\theta(t)=\theta$ and $\alpha(t)\geqslant\beta(t)$, $\gamma(t)=\alpha(t)-\beta(t)$; otherwise $\gamma(t)=0$.

The idea of PA theory is applied to the SFM to analyze the derivative of system performance. This approach is particularly suitable for the performance sensitivity analysis caused by the changes of buffer sizes in communication networks, such as the derivative of packet loss rate with respect to buffer size. It is known that the change of buffer size in queueing systems is not an infinitesimal perturbation and the traditional IPA theory is not applicable for this kind of problems. As the SFM gives the dynamic differential equation such as (31), we can analytically study the effect of changed parameters on the system performance. It is possible to obtain a simple expression of derivative estimator under certain scenarios. Research results also show that such derivative estimates are unbiased. Although SFM is an approximation of the original discrete event system, it can also be viewed as a general approach which may include other PA methods in certain sense. The original essence of discrete events can be recovered from the SFM with some proper modifications. But PA approach with SFM still has some limitations. This approach is valid currently only for some kinds of performance metrics and parameters under certain queueing systems. For more details and application cases, audience can refer to Cassandras et al. (2003), Cassandras et al. (2002), Liu and Gong (2002), Panayiotou and Cassandras (2006) and Wardi et al. (2002).

6. Discussion

This paper reviews the performance optimization theory and methodologies for queueing systems from a sensitivity-based view. Compared with the classical queueing theory for system performance, the sensitivity-based approach gives a new and efficient way to optimize the performance by extracting the sensitivity information from sample paths. Perturbation realization factor is the fundamental concept in this approach and it reflects the special

structure of queueing systems with regard to performance optimization and serves as the building blocks for analyzing the effects of parameter changes on the system performance. Based on the perturbation realization, two kinds of sensitivity formulas, performance derivative and performance difference, are derived. With the performance derivative formula, gradient-based optimization algorithms can be developed; and with the performance difference formula, policy iteration based optimization algorithms and the optimality (H–J–B) equations can be derived. These two kinds of sensitivity formulas are fundamental to the optimization theory and provide a clear picture for the performance optimization of queueing systems. Sample path based estimation and learning methods can be combined with the optimization algorithms and make this optimization framework online implementable.

This paper gives an overview about performance optimization of queueing systems via perturbation realization. We must emphasize again: the performance difference formula is a totally new finding for the performance optimization of queueing systems and it has much significance for developing the optimization algorithms. Performance difference can provide much more sensitivity information than performance gradients, since the performance difference may approach the first order derivative when the difference is infinitesimal. Actually, performance difference may also induce the high order derivatives as we will show it in our future work. With these new sensitivity information, the optimization algorithms may perform much better than before. Performance difference formula and policy iteration based approach give a new direction for the performance optimization of queueing systems.

There are still many research problems worth further investigations. For example, the extension of difference formula and policy iteration to a more general queueing system besides the closed Jackson network, the difference formula for the changes in routing probabilities or buffer sizes, event-based optimization for the situation where the standard MDP theory is not applicable, etc. As this paper indicates, all these aforementioned problems can be probably understood more intuitively from a sensitivity based view with perturbation realization and it may deserve the attentions from the research community of queueing theory.

References

Bertsekas, D.P., Tsitsiklis, J.N., 1996. Neuro-Dynamic Programming. Athena Scientific, Belmont, MA.

Cao, X.R., 1985. Convergence of parameter sensitivity estimates in a stochastic experiment. IEEE Transactions on Automatic Control 30, 834–843.

Cao, X.R., 1987. Realization probability in closed Jackson queueing networks and its application. Advances in Applied Probability 19, 708–738.

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

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

Cao, X.R., 2003. From perturbation analysis to Markov decision processes and reinforcement learning. Discrete Event Dynamic Systems: Theory and Applications 13, 9–39.

Cao, X.R., 2007. Basic ideas for event-based optimization of Markov systems. Discrete Event Dynamic Systems: Theory and Applications 15, 169–197.

Cao, X.R., 2007. Stochastic Learning and Optimization – A Sensitivity-Based Approach. Springer, New York.

Cao, X.R., Chen, H.F., 1997. Perturbation realization, potentials, and sensitivity analysis of Markov processes. IEEE Transactions on Automatic Control 42, 1382–1393.

Cao, X.R., Yuan, X.M., Qiu, L., 1996. A single sample path-based performance sensitivity formula for Markov chains. IEEE Transactions on Automatic Control 41, 1814–1817.

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

Cassandras, C.G., Wardi, Y., Melamed, B., Sun, G., Panayiotou, C.G., 2002. Perturbation analysis for online control and optimization of stochastic fluid models. IEEE Transactions on Automatic Control 47, 1234–1248.

Chen, J.T., 2008. Queueing models of certain manufacturing cells under product-mix sequencing rules. European Journal of Operational Research 188, 826–837.

Fu, M.C., Hu, J.Q., 1994. Smoothed perturbation analysis derivative estimation for Markov chains. Operations Research Letters 15, 241–251.

Fu, M.C., 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, MA.

Glasserman, P., Gong, W.B., 1990. Smoothed perturbation analysis for a class of discrete event system. IEEE Transactions on Automatic Control 35, 1218–1230.

Gong, W.B., Ho, Y.C., 1987. Smoothed (conditional) perturbation analysis for discrete event dynamic systems. IEEE Transactions on Automatic Control 32, 858–866.

Gordon, W.J., Newell, G.F., 1967. Closed queueing systems with Exponential Servers. Operations Research 15, 252–265.

Heidelberger, P., Cao, X.R., Zazanis, M.A., Suri, R., 1989. Convergence properties of infinitesimal perturbation analysis estimates. Management Science 34, 1281–1302.

Heidergott, B., 1998. Optimisation of a single-component maintenance system: a smoothed perturbation analysis approach. European Journal of Operational Research 119, 181–190.

Heidergott, B., 2000. Customer-oriented finite perturbation analysis for queueing networks. Discrete Event Dynamic Systems: Theory and Applications 10, 201–232.

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

Ho, Y.C., Cao, X.R., 1991. Perturbation Analysis of Discrete Event Systems. Kluwer Academic Publishers., Norwell.

Ho, Y.C., Cao, X.R., Cassandras, C.G., 1983. Infinitesimal and finite perturbation analysis for queueing networks. Automatica 19, 439–445.

Kitaev, Y.M., Rykov, V.V., 1995. Controlled Queueing Systems. CRC Press, New York.

Li, Q.L., 2010. Constructive Computation in Stochastic Models with Applications: The RG-Factorizations. Springer and Tsinghua Press.

Liu, Y., Gong, W.B., 2002. Perturbation analysis for stochastic fluid queueing systems. Discrete Event Dynamic Systems: Theory and Applications 12, 391–416.

Ma, D.J., Cao, X.R., 1994. A direct approach to decentralized control of service rates in a closed Jackson network. IEEE Transactions on Automatic Control 39, 1460–1463.

Marbach, P., Mihatsch, O., Tsitsiklis, J.N., 2000. Call admission control and routing in integrated services networks using neuro-dynamic programming. IEEE Journal On Selected Areas In Communications 18, 197–208.

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

Panayiotou, C., Cassandras, C.G., 2006. Infinitesimal perturbation analysis and optimization for make-to-stock manufacturing systems based on stochastic fluid models. Discrete Event Dynamic Systems: Theory and Applications 16, 109–142.

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

Rappold, J.A., Roo, B.D.V., 2009. Designing multi-echelon service parts networks with finite repair capacity. European Journal of Operational Research 199, 781–792.

Sennott, L.I., 1999. Stochastic Dynamic Programming and the Control of Queueing Systems. Wiley, New York.

Stidham, S., Weber, R., 1993. A survey of Markov decision models for control of networks of queues. Queueing systems 13, 291–314.

Wardi, Y., Melamed, B., Cassandras, C.G., Panayiotou, C.G., 2002. Online IPA gradient estimators in stochastic continuous fluid models. Journal of Optimization Theory and Applications 115, 369–405.

Woensel, T.V., Kerbache, L., Peremans, H., Vandaele, N., 2008. Vehicle routing with dynamic travel times: a queueing approach. European Journal of Operational Research 186, 990–1007.

Xia, L., Cao, X.R., 2006a. Relationship between perturbation realization factors with queueing models and Markov models. IEEE Transactions on Automatic Control 51, 1699–1704.

Xia, L., Cao, X.R., 2006b. Aggregation of perturbation realization factors and service rate-based policy iteration for queueing systems. In: the Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, pp. 1063–1068.

Xia, L., Chen, X., Cao, X.R., 2009. Policy iteration for customer-average performance optimization of closed queueing systems. Automatica 45, 1639–1648.

Xia, L., Shihada, B., submitted for publication. Max–Min optimality of service rate control in closed queueing networks. IEEE Transactions on Automatic Control.

Yao, D.D., Schechner, Z., 1989. Decentralized control of service rates in a closed Jackson network. IEEE Transactions on Automatic Control 34, 236–240.

Zhang, J., Cao, X.R., 2009. Continuous-time Markov decision processes with nth-bias optimality criteria. Automatica 45, 1628–1638.