Automatica 45 (2009) 1639–1648

Contents lists available at ScienceDirect

Automatica

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

Policy iteration for customer-average performance optimization of closed queueing systems☆

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

[a] IBM China Research Laboratory, ZhongGuanCun Software Park, Beijing 100193, China

[b] CFINS, Department of Automation, Tsinghua University, Beijing 100084, China

[c] Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 5 September 2007
Received in revised form
18 January 2009
Accepted 9 March 2009
Available online 22 April 2009

Keywords:
Perturbation analysis
Customer-average performance
Policy iteration

ABSTRACT

We consider the optimization of queueing systems with service rates depending on system states. The optimization criterion is the long-run customer-average performance, which is an important performance metric, different from the traditional time-average performance. We first establish, with perturbation analysis, a difference equation of the customer-average performance in closed networks with exponentially distributed service times and state-dependent service rates. Then we propose a policy iteration optimization algorithm based on this difference equation. This algorithm can be implemented on-line with a single sample path and does not require knowing the routing probabilities of queueing systems. Finally, we give numerical experiments which demonstrate the efficiency of our algorithm. This paper gives a new direction to efficiently optimize the "customer-centric" performance in queueing systems.

1. Introduction

Queueing systems are widely used in modeling modern engineering systems such as communication networks, manufacturing systems, transportation and logistic systems, etc. In a queueing system, customers arrive to service stations (called servers) and wait in queues to be served; each customer requires a random amount of service from a server which provides service with a certain speed called service rate. We may control the service rates of the servers. The system pays a cost to provide the service and to cover the losses due to the waiting of customers. We wish to minimize the cost by choosing the right service rates. The cost is the performance to be optimized. We quantify the cost per served customers and call it customer-average performance.

Our recent study (Cao & Chen, 1997) shows that many results in performance optimization of stochastic systems can be derived and explained with two types of performance sensitivity equations: performance derivative equations and performance difference equations (see (11) and (12) in Section 2). With performance derivatives, gradient-based optimization approaches can be developed for Markov systems, and with the special structure of the performance difference equations, policy iteration algorithms in Markov decision processes (MDP) can be developed (Cao & Chen, 1997).

Perturbation analysis (PA) provides efficient algorithms to estimate the performance gradients with respect to system parameters (e.g., the service rates of servers in queueing systems) by analyzing a single sample path (Cao, 1994; Cassandras & Lafortune, 1999; Glasserman, 1991; Gong & Ho, 1987; Ho & Cao, 1991; Ho, Cao, & Cassandras, 1983). Performance derivative equations have been developed for queueing systems (Cao, 1994). PA was extended from queueing systems to Markov systems in the past decade (Cao & Chen, 1997). Both the performance derivative and performance difference equations have been established for Markov systems. Thus, both gradient-based and policy iteration type of optimization algorithms have been developed for Markov systems. However, so far we do not have the performance difference equation for queueing systems and therefore, we do not have policy iteration type of algorithms for queueing systems.

The goal of this paper is to derive the performance difference equations for queueing systems with exponentially distributed service times, and based on these equations to develop policy iteration algorithms for performance optimization of such queueing systems. These policy iteration algorithms are similar to those in MDP and converge fast to the optimal policy.

Both PA of queueing systems and PA of Markov systems are based on a concept called perturbation realization factor (for

☆ This paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Bart De Schutter under the direction of Editor Ian R. Petersen.

* Corresponding author. Tel.: +852 2358 7048; fax: +852 2358 1485.
E-mail addresses: crlxiali@cn.ibm.com (L. Xia), bjchenxi@tsinghua.edu.cn (X. Chen), eecao@ust.hk (X.-R. Cao).

0005-1098/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2009.03.007

simplicity, we also call it realization factor). Roughly speaking, the realization factor of a state $\mathbf{n}$ measures the final effect of a small perturbation at this state on the performance. The perturbation realization factor for queueing systems is completely different from that for Markov systems. This is because the perturbations in queueing systems are infinitesimal delays of service times, which are continuous in nature, while the perturbations in Markov systems are the jumps of system states, which are discrete in the case of Markov process representations of queueing systems. However, since a closed queueing network with exponentially distributed service times can be modeled as a Markov process, it is natural to expect that the realization factors for queueing systems and those for Markov systems are related. Such a relationship was established in Xia and Cao (2006). Based on this relationship, we can develop similar results for both types of systems. Particularly, in this paper, we will develop the performance difference equations for queueing systems.

One important feature of our results is that the performance difference equations and optimization algorithms developed in this paper are for the customer-average performance, instead of the time-average performance used in many performance optimization problems. Customer-average performance quantifies the system performance averaged by the number of served customers and reflects the "customer-centric" view, which is popular in many service industries. In practice, many performance measurements belong to the customer-average performance. Examples include the average waiting time of each customer in a bank, the probability of a packet successfully reaching the destination node in a wireless communication network, and the average hop-count of a packet passing by before it reaches the destination in a packet switched network. However, to the best of our knowledge, most optimization algorithms usually concern the time-average performance and not much literature discusses the optimization of customer averages.

In summary, the contributions of this paper include: (1) We derive the customer-average performance difference equations for a state-dependent closed Jackson network (a network with exponentially distributed service times and state-dependent service rates, see Section 2 for a formal definition), (2) We develop a policy iteration optimization algorithm for customer-average performance of state-dependent closed Jackson networks; this algorithm is unique in the sense that it is based on perturbation realization factors, which brings new insights and may lead to new research topics such as event-based aggregation (Cao, 2005), and (3) We develop an on-line algorithm to estimate the perturbation realization factors and to implement the policy iteration algorithm. Numerical examples show that the policy iteration algorithm converges fast and the on-line estimation algorithm is accurate.

The rest of this paper is organized as follows. Section 2 gives a background review of PA theory in queueing systems and Markov systems. In Section 3, we derive the difference equation of customer-average performance in queueing systems. Furthermore, in Section 4, we propose the policy iteration algorithm for customer-average performance and the on-line estimation algorithm for perturbation realization factors. Numerical experiments are shown in Section 5 to demonstrate the convergence of the policy iteration algorithm and the accuracy of the on-line estimation algorithm. Finally, in Section 6, we give a brief discussion and conclusion of this paper.

2. Background on perturbation analysis theory

2.1. Perturbation analysis of queueing systems

Consider a closed Jackson network consisting of M servers Chen and Yao (2001), Gordon and Newell (1967). The number of total customers in the network is N. The customers are cycling among the servers. After the completion of the service of a customer at server i, the customer will depart from server i and enter server j with routing probabilities q_{ij}, $i, j = 1, 2, \ldots, M$. The service discipline in each server is FCFS (First-Come First-Served) and the buffer size is adequate (not smaller than N). Let n_i be the number of customers at server i, $\mathbf{n} = (n_1, n_2, \ldots, n_M)$ be the system state, and $\mathcal{S} = \{all\ \mathbf{n} : \sum_{i=1}^{M} n_i = N\}$ be the state space. The service rate of server i depends on system state $\mathbf{n}$ and is therefore denoted as $\mu_{i,\mathbf{n}}$, $i = 1, 2, \ldots, M$, $\mathbf{n} \in \mathcal{S}$. The service time of server i at state $\mathbf{n}$ is a random variable which is exponentially distributed with parameter $\mu_{i,\mathbf{n}}$. The distribution function is written as $F(t) = 1 - \mathrm{e}^{-t/\mu_{i,\mathbf{n}}}$, where t is the service time. When the system state $\mathbf{n}$ changes, the service rate $\mu_{i,\mathbf{n}}$ will change, therefore the service time of server i will also change since its distribution function $F(t)$ is changed. The details involve the generation of random variables and can be referred to the Chapter 4 of Cao (1994). Such type of service rates is called *state-dependent* service rates and such type of networks is also called *state-dependent* closed Jackson networks.

Let $\mathbf{n}(t)$ denote the system state at time t; $\mathbf{n}(t)$, $t \in [0, \infty)$ is a stochastic process. Let $f : \mathcal{S} \to \mathcal{R} = (-\infty, \infty)$ be the cost function. Let T_L be the time when the whole network has served L customers (i.e., the Lth service completion time of the network). Consider a sample path of $\mathbf{n}(t)$, the *time-average performance* on the sample path is defined as

$$\eta_T = \lim_{L\to\infty} \frac{\int_0^{T_L} f(\mathbf{n}(t))\mathrm{d}t}{T_L} = \lim_{L\to\infty} \frac{F_L}{T_L}, \quad w.p.1, \tag{1}$$

where $F_L := \int_0^{T_L} f(\mathbf{n}(t))\mathrm{d}t$ denotes the accumulated performance until T_L. On the other hand, we consider the *customer-average performance* which defines the average performance on each served customer as follows.

$$\eta^{(f)} = \lim_{L\to\infty} \frac{F_L}{L}, \quad w.p.1. \tag{2}$$

We assume that the state process $\mathbf{n}(t)$ is ergodic; thus, the two limits in (1) and (2) exist with probability one (w.p.1) and are independent of the sample path. This is true for a closed Jackson network if it is strongly connected: that is, customers in each server have a positive probability of visiting every other server in the network.

We now review the perturbation realization factors in PA theory. In a closed network, if the service completion time of a server is delayed by a small amount of time Δ, we say that the server has a perturbation. This perturbation will affect the system performance $\eta^{(f)}$. The effect of a single perturbation Δ of server k when the system is in state $\mathbf{n}$ at time 0 can be measured by the *perturbation realization factor*, which is defined as

$$\begin{aligned} c^{(f)}(\mathbf{n}, k) &= \lim_{L\to\infty} \lim_{\Delta\to 0} E\left\{\frac{\Delta F_L}{\Delta}\right\} \\ &= \lim_{L\to\infty} \lim_{\Delta\to 0} E\left\{\frac{F'_L - F_L}{\Delta}\right\} \\ &= \lim_{L\to\infty} \lim_{\Delta\to 0} E\left\{\frac{1}{\Delta}\left(\int_0^{T'_L} f(\mathbf{n}'(t))\mathrm{d}t - \int_0^{T_L} f(\mathbf{n}(t))\mathrm{d}t\right)\right\}, \end{aligned} \tag{3}$$

where $\mathbf{n}'(t)$ is the state of the perturbed sample path at time t, T'_L is the Lth service completion time on the perturbed sample path. We define the perturbation realization probability $c(\mathbf{n}, k)$ as a special $c^{(f)}(\mathbf{n}, k)$ with $f(\mathbf{n}) = I(\mathbf{n}) \equiv 1$ for all $\mathbf{n} \in \mathcal{S}$, i.e., $c(\mathbf{n}, k) := c^{(I)}(\mathbf{n}, k)$. It can be verified that $c(\mathbf{n}, k)$ is the probability that eventually every server in the network will have the same perturbation as that of server k at state $\mathbf{n}$ via propagation.

With the realization factors $c^{(f)}(\mathbf{n}, k)$, the performance derivatives with respect to the service rates are derived as follows (Cao, 1994).

$$\frac{\mathrm{d}\eta^{(f)}}{\mathrm{d}\mu_{k,\mathbf{n}}} = -\frac{\eta^{(I)}}{\mu_{k,\mathbf{n}}}\pi(\mathbf{n})c^{(f)}(\mathbf{n}, k), \tag{4}$$

where $\eta^{(I)}$ is a special $\eta^{(f)}$ with $f(\mathbf{n}) = I(\mathbf{n}) \equiv 1$ for all $\mathbf{n} \in \mathcal{S}$, and $\pi(\mathbf{n})$ is the steady-state probability of state $\mathbf{n}$. Since $\eta^{(I)}$ is a special case of $\eta^{(f)}$, from (2) we have

$$\eta^{(I)} = \lim_{L\to\infty} \frac{\int_0^{T_L} I(\mathbf{n}(t))\mathrm{d}t}{L} = \lim_{L\to\infty} \frac{T_L}{L}. \tag{5}$$

It is easy to show that $\eta^{(I)}$ is the reciprocal of the average throughput of the network, i.e.,

$$\eta_{th} \coloneqq \lim_{L\to\infty} \frac{L}{T_L} = \frac{1}{\eta^{(I)}}, \tag{6}$$

where η_{th} denotes the throughput of the queueing network. With (1), (2) and (5), we have

$$\eta^{(f)} = \lim_{L\to\infty} \frac{F_L}{T_L}\frac{T_L}{L} = \lim_{L\to\infty} \frac{F_L}{T_L} \lim_{L\to\infty} \frac{T_L}{L} = \eta_T\eta^{(I)}, \tag{7}$$

which shows the relation between the customer-average performance $\eta^{(f)}$ and the time-average performance η_T. The second equality of (7) can be verified from the definition of limit and physical meanings of F_L and T_L. Eq. (7) has a strong affinity to the Little's Law in queueing systems. Especially, for an M/M/1 queue, if the cost function is $f(\mathbf{n}) = n$, then $\eta^{(f)}$ is the mean response time (waiting time + service time) of customers, η_T is the queue length (including the customer being served), and $\eta^{(I)}$ is the reciprocal of customer arrival rate. Thus, for this special case, (7) is exactly the Little's Law.

From (7) we can see, since the service rates will affect the values of both η_T and $\eta^{(I)}$, the optimal values of service rates for $\eta^{(f)}$ and η_T are generally different. The numerical experiments in Section 5.2 further demonstrate this point. Therefore, it is necessary to develop an algorithm to optimize the customer-average performance.

With (4), in order to obtain the performance derivative, we only need to estimate the values of $\eta^{(I)}$, $\pi(\mathbf{n})$, and $c^{(f)}(\mathbf{n}, k)$. From the physical meaning of $\eta^{(I)}$ and $\pi(\mathbf{n})$, we can estimate them easily based on the sample paths. While, the estimation of $c^{(f)}(\mathbf{n}, k)$ can also be implemented based on the definition of realization factors (3). The main idea is to simulate the perturbations propagated through the network and calculate the induced increment of system performance. Thus the realization factors can be obtained through the sample path. The details can be found in Algorithm 2 Section 4. With the estimated performance derivative, we can use the gradient-descent algorithms to optimize the system performance (Cao, 1994; Chong & Zak, 2001).

2.2. Perturbation analysis of Markov systems

Consider an ergodic Markov process $X = \{X_t, t \geq 0\}$, where X_t is the system state at time t. The state space is finite and denoted as $\mathcal{S} = \{1, 2, \ldots, S\}$. The infinitesimal generator (also called transition rate matrix) is $B = [b(u, v)]_{S\times S}$, where $b(u, v) \geq 0$ if $u \neq v$, $b(u, u) = -\sum_{v\in\mathcal{S}, v\neq u} b(u, v)$, $u, v \in \mathcal{S}$. We use a row vector π to denote the steady-state distribution and let e be a column vector whose elements are all one. We have $Be = 0$, $\pi e = 1$, $\pi B = 0$. Let the column vector f be the cost function with $f(u)$ denoting the cost at state u, $u \in \mathcal{S}$. Then, the time-average performance is $\eta_T = \pi f$.

In PA of Markov systems, we define the *performance potential* g to measure the "potential" contribution of a state to the total system performance η_T (Cao & Chen, 1997). g is a column vector whose element $g(u)$, $u \in \mathcal{S}$, is defined as

$$g(u) = \lim_{T\to\infty} E\left\{\int_0^T [f(X_t) - \eta_T]\mathrm{d}t \mid X_0 = u\right\}. \tag{8}$$

From (8), we can see that g is also known as the bias in Markov decision theory. We can further derive the Poisson equation as below (Cao & Chen, 1997).

$$Bg = -f + \eta_T e. \tag{9}$$

The *perturbation realization factor* in PA of Markov systems is defined as the difference between the performance potentials of any two states.

$$\begin{aligned}\mathrm{d}(u, v) &= g(v) - g(u)\\ &= \lim_{T\to\infty} E\left\{\int_0^T [f(X'_t) - f(X_t)]\mathrm{d}t | X_0 = u, X'_0 = v\right\}.\end{aligned} \tag{10}$$

As we may see, $\mathrm{d}(u, v)$ is the difference of the contributions of two states to the total system performance. It measures the effect on the system's transient performance if the initial state is perturbed from u to v.

The performance difference equation in Markov systems can be derived from the Poisson equation (9). Consider two Markov processes with infinitesimal generators B and B', and cost functions f and f', respectively. Let π and π' be the steady-state distributions and $\eta_T = \pi f$ and $\eta'_T = \pi' f'$ be the time-average performance of the two Markov processes, respectively, and g be the performance potential of the system with parameters (B, f). Let $\Delta B = B' - B$ and $h = f' - f$. The following performance difference equation is derived.

$$\eta'_T - \eta_T = \pi'(\Delta B g + h). \tag{11}$$

Next, consider a randomized policy which adopts B' with probability δ and adopts B with probability $1 - \delta$. The infinitesimal generator of this system is $B^{(\delta)} = (1 - \delta)B + \delta B' = B + \delta\Delta B$. When $\delta \ll 1$, we denote the cost function under this policy as $f^{(\delta)} = f + \delta h$ with $h = \frac{\partial f^{(\delta)}}{\partial \delta}$. The time-average performance of this system with parameters $(B^{(\delta)}, f^{(\delta)})$ is denoted as $\eta_T^{(\delta)}$. We get the derivative of $\eta_T^{(\delta)}$ with respect to δ as below.

$$\frac{\mathrm{d}\eta_T^{(\delta)}}{\mathrm{d}\delta} = \pi(\Delta B g + h). \tag{12}$$

Eqs. (11) and (12) are the fundamental formulas for the PA theory of Markov systems. For example, we can derive the policy iteration algorithm directly from the performance difference equation (11). The details can be found in Cao and Chen (1997).

As explained in Section 3, a closed queueing network with exponentially distributed service times can be modeled by a Markov process, whose infinitesimal generator B can be determined by $\mu_{i,\mathbf{n}}$ and q_{ij}. Therefore, the optimization of a queueing network can be performed using techniques from both PA of queueing systems and Markov systems. From the definitions (3) and (10), we can see there are two types of realization factors in queueing systems and Markov systems, i.e., $c^{(f)}(\mathbf{n}, k)$ and $\mathrm{d}(u, v)$ respectively. The first one measures the effect of an infinitesimal continuous perturbation of service time on the customer-average performance $\eta^{(f)}$, while the second one describes the effect of a discrete jump of system initial state on performance η_T. Although they describe different characteristics of the systems, it seems they must have some relations. Recently, the following relationship formula has been established (Xia & Cao, 2006).

$$c^{(f)}(\mathbf{n}, k) - c(\mathbf{n}, k)\eta_T = \epsilon(n_k)\mu_{k,\mathbf{n}} \sum_{j=1}^{M} q_{kj}\mathrm{d}(\mathbf{n}_{kj}, \mathbf{n}), \tag{13}$$

where $\mathbf{n}_{kj} = (n_1, \ldots, n_k - 1, \ldots, n_j + 1, \ldots, n_M)$ is a neighboring state of $\mathbf{n}$ with $n_k > 0$, $\epsilon(n_k)$ is an indicator function which is defined as: if $n_k > 0$, $\epsilon(n_k) = 1$; otherwise $\epsilon(n_k) = 0$. This formula builds up a bridge between PA of queueing systems and Markov systems. One of the direct consequences is that we can establish parallel results between these two theories. For example, similar to the difference equation (11) in Markov systems, we may establish the difference equation with realization factors $c^{(f)}(\mathbf{n}, k)$ for queueing systems. This leads to the policy iteration algorithm for customer-average performance of queueing systems, which is new in the literature. We will discuss the details in the next two sections.

3. Difference equation of customer-average performance

In this section, we first derive a difference equation for the case when the cost function is set as $f(\mathbf{n}) = \mu(\mathbf{n}) := \sum_{k=1}^{M} \epsilon(n_k)\mu_{k,\mathbf{n}}$ and a single service rate is changed. Then we extend it to an arbitrary cost function and allow multiple service rates to change.

In order to obtain the difference equation, we give the following theorem.

Theorem 3.1. *In a state-dependent closed Jackson network, if $f(\mathbf{n}) = \mu(\mathbf{n}) := \sum_{k=1}^{M} \epsilon(n_k)\mu_{k,\mathbf{n}}$, the perturbation realization factor is $c^{(f)}(\mathbf{n}, k) = \epsilon(n_k)\mu_{k,\mathbf{n}}$, $k = 1, 2, \ldots, M$, $\mathbf{n} \in \mathcal{S}$.*

Proof. The detailed proof can be found in the Appendix. □

Theorem 3.1 means that when the cost function $f(\mathbf{n})$ is equal to the sum of "active" service rates $\mu(\mathbf{n})$, the perturbation realization factors will have a special form $c^{(f)}(\mathbf{n}, k) = \epsilon(n_k)\mu_{k,\mathbf{n}}$. With this cost function, the time-average performance is the average throughput of the network: $\eta_T = \pi f = \sum_{\mathbf{n}\in\mathcal{S}} \pi(\mathbf{n})\mu(\mathbf{n}) = \sum_{\mathbf{n}\in\mathcal{S}} \pi(\mathbf{n}) \sum_{k=1}^{M} \epsilon(n_k)\mu_{k,\mathbf{n}}$, i.e., $\eta_T = \eta_{th}$, when $f(\mathbf{n}) = \mu(\mathbf{n})$.

Based on Theorem 3.1, we first derive the performance difference equation in queueing systems when the cost function is $f(\mathbf{n}) = \mu(\mathbf{n})$, for all $\mathbf{n} \in \mathcal{S}$. A state-dependent closed Jackson network can be modeled by a Markov process with infinitesimal generator B, which can be determined by $\mu_{i,\mathbf{n}}$ and q_{ij}, $i, j = 1, 2, \ldots, M$ and $\mathbf{n} \in \mathcal{S}$.

In order to more clearly show the structure of matrix B, we give an example network with $M = 3$, $N = 2$. Thus the system has 6 states: $\mathbf{n}_1, \ldots, \mathbf{n}_6$, which are (2, 0, 0), (1, 1, 0), (1, 0, 1), (0, 2, 0), (0, 1, 1), (0, 0, 2) respectively. With the same order of above states, B can be written as in Box I.

The elements of matrix B are obtained through a standard analysis of this stochastic system, e.g., the first element $-\mu_{1,\mathbf{n}_1}$ means that the transition rate from state (2, 0, 0) to other states. For a general network, the structure of matrix B is similar.

We pick a particular state, denoted as $\mathbf{n}$, and assume that the service rate of a particular server, denoted as server k, changes from $\mu_{k,\mathbf{n}}$ to $\mu_{k,\mathbf{n}} + \Delta\mu_{k,\mathbf{n}}$; the infinitesimal generator will change from B to $B' = B + \Delta B$. We will establish the difference equation for η_{th} by using (11). From the structure of matrix B, all the elements of matrix $\Delta B = B' - B$ are zero except the row corresponding to state $\mathbf{n}$. We denote this non-zero row as $\Delta B(\mathbf{n}, *)$. For this row, the value of the diagonal element is $\Delta B(\mathbf{n}, \mathbf{n}) = -\epsilon(n_k)\Delta\mu_{k,\mathbf{n}}$; the value of the neighboring states of $\mathbf{n}$ is $\Delta B(\mathbf{n}, \mathbf{n}_{kj}) = \epsilon(n_k)q_{kj}\Delta\mu_{k,\mathbf{n}}$, $j = 1, 2, \ldots, M$; and all the other elements are zero. Because $f(\mathbf{n}) = \mu(\mathbf{n})$, the cost function f changes by $h = f' - f = \mu' - \mu$, where μ is a vector consisting of $\mu(\mathbf{n})$. We have $h(\mathbf{n}) = \epsilon(n_k)\Delta\mu_{k,\mathbf{n}}$ and $h(\mathbf{n}') = 0$ for all $\mathbf{n}' \neq \mathbf{n}$. Therefore, following (11), we obtain

$$\begin{aligned}
\eta'_{th} - \eta_{th} &= \pi'(\Delta B g + h) \\
&= \pi'(\mathbf{n})\left[\sum_{j=1}^{M} \epsilon(n_k)q_{kj}\Delta\mu_{k,\mathbf{n}} g(\mathbf{n}_{kj}) - \epsilon(n_k)\Delta\mu_{k,\mathbf{n}} g(\mathbf{n}) + h(\mathbf{n})\right] \\
&= \pi'(\mathbf{n})\left[\sum_{j=1}^{M} \epsilon(n_k)q_{kj}\Delta\mu_{k,\mathbf{n}} g(\mathbf{n}_{kj}) - \epsilon(n_k)\Delta\mu_{k,\mathbf{n}} g(\mathbf{n}) \sum_{j=1}^{M} q_{kj} + \epsilon(n_k)\Delta\mu_{k,\mathbf{n}}\right] \\
&= \pi'(\mathbf{n})\left[\epsilon(n_k)\Delta\mu_{k,\mathbf{n}} \sum_{j=1}^{M} q_{kj}\left(g(\mathbf{n}_{kj}) - g(\mathbf{n})\right) + \epsilon(n_k)\Delta\mu_{k,\mathbf{n}}\right] \\
&= \pi'(\mathbf{n})\left[-\epsilon(n_k)\Delta\mu_{k,\mathbf{n}} \sum_{j=1}^{M} q_{kj}\mathrm{d}(\mathbf{n}_{kj}, \mathbf{n}) + \epsilon(n_k)\Delta\mu_{k,\mathbf{n}}\right] \qquad (14)
\end{aligned}$$

Substituting the relationship (13) into (14), we get

$$\begin{aligned}
\eta'_{th} - \eta_{th} &= -\pi'(\mathbf{n})\frac{\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}\left[c^{(f)}(\mathbf{n}, k) - c(\mathbf{n}, k)\eta_{th}\right] \\
&\quad + \pi'(\mathbf{n})\epsilon(n_k)\Delta\mu_{k,\mathbf{n}}. \qquad (15)
\end{aligned}$$

From Theorem 3.1, $c^{(f)}(\mathbf{n}, k) = \epsilon(n_k)\mu_{k,\mathbf{n}}$. Substituting it into (15) obtains the performance difference equation for η_{th}

$$\begin{aligned}
\eta'_{th} - \eta_{th} &= -\pi'(\mathbf{n})\frac{\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}\left[\epsilon(n_k)\mu_{k,\mathbf{n}} - c(\mathbf{n}, k)\eta_{th}\right] \\
&\quad + \pi'(\mathbf{n})\epsilon(n_k)\Delta\mu_{k,\mathbf{n}} \\
&= \pi'(\mathbf{n})\frac{\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}} c(\mathbf{n}, k)\eta_{th}. \qquad (16)
\end{aligned}$$

Substituting (6) into (16), we have

$$\frac{1}{\eta'^{(l)}} - \frac{1}{\eta^{(l)}} = \frac{\eta^{(l)} - \eta'^{(l)}}{\eta^{(l)}\eta'^{(l)}} = \pi'(\mathbf{n})\frac{\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}\frac{c(\mathbf{n}, k)}{\eta^{(l)}}. \qquad (17)$$

Thus, we have the following theorem about the performance difference equation of $\eta^{(l)}$.

Theorem 3.2. *For a state-dependent closed Jackson network, if a particular service rate changes from $\mu_{k,\mathbf{n}}$ to $\mu_{k,\mathbf{n}} + \Delta\mu_{k,\mathbf{n}}$, the performance difference of $\eta^{(l)}$ is*

$$\eta'^{(l)} - \eta^{(l)} = -\eta'^{(l)}\pi'(\mathbf{n})\frac{\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}} c(\mathbf{n}, k). \qquad (18)$$

Based on (18), we can further derive the performance difference equation of $\eta^{(f)}$ for an arbitrary cost function f. We assume that for a particular state $\mathbf{n}$, the service rate of a particular server k changes from $\mu_{k,\mathbf{n}}$ to $\mu_{k,\mathbf{n}} + \Delta\mu_{k,\mathbf{n}}$, and therefore, the infinitesimal generator changes from B to B'. We also assume that the cost function changes from $f(\mathbf{n})$ to $f'(\mathbf{n})$. Thus, all the elements of $h = f' - f$ are zero except for the state $\mathbf{n}$, $h(\mathbf{n}) = f'(\mathbf{n}) - f(\mathbf{n})$. Since we have already determined the value of matrix ΔB, the difference of η'_T and η_T can be obtained by using (11). With a similar analysis to (14) and (15), we have

$$\begin{aligned}
\eta'_T - \eta_T &= -\pi'(\mathbf{n})\frac{\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}\left[c^{(f)}(\mathbf{n}, k) - c(\mathbf{n}, k)\eta_T\right] \\
&\quad + \pi'(\mathbf{n})h(\mathbf{n}). \qquad (19)
\end{aligned}$$

With (7), we have

$$\begin{aligned}
\eta'^{(f)} - \eta^{(f)} &= \eta'_T\eta'^{(l)} - \eta_T\eta^{(l)} \\
&= \eta'_T\eta'^{(l)} - \eta_T\eta'^{(l)} + \eta_T\eta'^{(l)} - \eta_T\eta^{(l)} \\
&= \eta'^{(l)}(\eta'_T - \eta_T) + \eta_T(\eta'^{(l)} - \eta^{(l)}). \qquad (20)
\end{aligned}$$

Substituting (18) and (19) into (20), we get the following theorem.

$$B = \begin{bmatrix} -\mu_{1,\mathbf{n}_1} & \mu_{1,\mathbf{n}_1}q_{12} & \mu_{1,\mathbf{n}_1}q_{13} & 0 & 0 & 0 \\ \mu_{2,\mathbf{n}_2}q_{21} & -\mu_{1,\mathbf{n}_2}-\mu_{2,\mathbf{n}_2} & \mu_{2,\mathbf{n}_2}q_{23} & \mu_{1,\mathbf{n}_2}q_{12} & \mu_{1,\mathbf{n}_2}q_{13} & 0 \\ \mu_{3,\mathbf{n}_3}q_{31} & \mu_{3,\mathbf{n}_3}q_{32} & -\mu_{1,\mathbf{n}_3}-\mu_{3,\mathbf{n}_3} & 0 & \mu_{1,\mathbf{n}_3}q_{12} & \mu_{1,\mathbf{n}_3}q_{13} \\ 0 & \mu_{2,\mathbf{n}_4}q_{21} & 0 & -\mu_{2,\mathbf{n}_4} & \mu_{2,\mathbf{n}_4}q_{23} & 0 \\ 0 & \mu_{3,\mathbf{n}_5}q_{31} & \mu_{2,\mathbf{n}_5}q_{21} & \mu_{3,\mathbf{n}_5}q_{32} & -\mu_{2,\mathbf{n}_5}-\mu_{3,\mathbf{n}_5} & \mu_{2,\mathbf{n}_5}q_{23} \\ 0 & 0 & \mu_{3,\mathbf{n}_6}q_{31} & 0 & \mu_{3,\mathbf{n}_6}q_{32} & -\mu_{3,\mathbf{n}_6} \end{bmatrix}$$

Box I.

Theorem 3.3. *For a state-dependent closed Jackson network, if a particular service rate changes from $\mu_{k,\mathbf{n}}$ to $\mu_{k,\mathbf{n}} + \Delta\mu_{k,\mathbf{n}}$, the performance difference of $\eta^{(f)}$ is*

$$\eta'^{(f)} - \eta^{(f)} = \eta'^{(l)}\pi'(\mathbf{n})\left\{\frac{-\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}c^{(f)}(\mathbf{n},k) + h(\mathbf{n})\right\}. \tag{21}$$

It is obvious that (18) is a special case of (21) when the cost function $f(\mathbf{n}) = l(\mathbf{n})$ for all $\mathbf{n} \in \mathcal{S}$.

In the previous analysis, we assume that the service rate $\mu_{k,\mathbf{n}}$ changes only at one particular state $\mathbf{n}$. Now, we consider the situation where the service rates of server k change at all the states $\mathbf{n}$, i.e., $\mu_{k,\mathbf{n}}$ changes to $\mu_{k,\mathbf{n}} + \Delta\mu_{k,\mathbf{n}}$ for all $\mathbf{n} \in \mathcal{S}$. Suppose that the cost function also changes from $f(\mathbf{n})$ to $f'(\mathbf{n})$ for all $\mathbf{n} \in \mathcal{S}$. Following the same analysis as above, we obtain the following difference equation for $\eta^{(f)}$.

$$\eta'^{(f)} - \eta^{(f)} = \eta'^{(l)}\sum_{\mathbf{n}\in\mathcal{S}}\pi'(\mathbf{n})\left\{\frac{-\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}c^{(f)}(\mathbf{n},k) + h(\mathbf{n})\right\}, \tag{22}$$

where $h(\mathbf{n}) = f'(\mathbf{n}) - f(\mathbf{n})$ for all $\mathbf{n} \in \mathcal{S}$.

Furthermore, we consider the situation where the service rates of all servers change from $\mu_{k,\mathbf{n}}$ to $\mu_{k,\mathbf{n}} + \Delta\mu_{k,\mathbf{n}}$, $k = 1, 2, \ldots, M$, $\mathbf{n} \in \mathcal{S}$. With a similar analysis, we obtain the performance difference equation

$$\eta'^{(f)} - \eta^{(f)} = \eta'^{(l)}\sum_{\mathbf{n}\in\mathcal{S}}\pi'(\mathbf{n})\left\{\sum_{k=1}^{M}\frac{-\Delta\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}c^{(f)}(\mathbf{n},k) + h(\mathbf{n})\right\}. \tag{23}$$

Eq. (23) can be further simplified as follows.

$$\begin{aligned}\eta'^{(f)} - \eta^{(f)} &= \eta'^{(l)}\sum_{\mathbf{n}\in\mathcal{S}}\pi'(\mathbf{n})\left\{\sum_{k=1}^{M}\frac{\mu_{k,\mathbf{n}} - \mu'_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}c^{(f)}(\mathbf{n},k) + f'(\mathbf{n}) - f(\mathbf{n})\right\} \\ &= \eta'^{(l)}\sum_{\mathbf{n}\in\mathcal{S}}\pi'(\mathbf{n})\left\{\sum_{k=1}^{M}c^{(f)}(\mathbf{n},k) - \sum_{k=1}^{M}\frac{\mu'_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}c^{(f)}(\mathbf{n},k) + f'(\mathbf{n}) - f(\mathbf{n})\right\}.\end{aligned} \tag{24}$$

Substituting (32) of the Appendix into (24), we have

$$\eta'^{(f)} - \eta^{(f)} = \eta'^{(l)}\sum_{\mathbf{n}\in\mathcal{S}}\pi'(\mathbf{n})\left\{f'(\mathbf{n}) - \sum_{k=1}^{M}\frac{\mu'_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}c^{(f)}(\mathbf{n},k)\right\}. \tag{25}$$

So far, we have assumed that the service rates are state-dependent. If the service rates are state-independent, e.g., $\mu_{k,\mathbf{n}} = \mu_k$ for all $\mathbf{n} \in \mathcal{S}$, $k = 1, 2, \ldots, M$, we can also derive the performance difference equations with a similar procedure. The specific forms of difference equations are the same as those in state-dependent networks except that all the $\mu_{k,\mathbf{n}}$ are replaced by μ_k, e.g., Eq. (22) will take the following form

$$\begin{aligned}\eta'^{(f)} - \eta^{(f)} &= \eta'^{(l)}\sum_{\mathbf{n}\in\mathcal{S}}\pi'(\mathbf{n})\left\{\frac{-\Delta\mu_k}{\mu_k}c^{(f)}(\mathbf{n},k) + h(\mathbf{n})\right\} \\ &= \eta'^{(l)}\left\{\frac{-\Delta\mu_k}{\mu_k}\sum_{\mathbf{n}\in\mathcal{S}}\pi'(\mathbf{n})c^{(f)}(\mathbf{n},k) + \sum_{\mathbf{n}\in\mathcal{S}}\pi'(\mathbf{n})h(\mathbf{n})\right\}.\end{aligned} \tag{26}$$

Until now, we have derived the difference equation for $\eta^{(f)}$. Both the derivative equation (4) and the difference equations (21)–(26), are expressed in terms of the realization factors $c^{(f)}(\mathbf{n},k)$. They have nothing to do with the realization factors of perturbed systems $c'^{(f)}(\mathbf{n},k)$. Therefore, we only need to estimate $c^{(f)}(\mathbf{n},k)$ under the current system parameters to implement the policy iteration optimization (see the next section). From (21)–(23), the performance difference is linear with respect to the realization factors, and the non-linear property is reflected by $\pi'(\mathbf{n})$. This is an interesting characteristic for queueing systems.

Finally, we emphasize again that this performance metric $\eta^{(f)}$ defined in (2) is a customer-average performance. In the standard MDP formulation, the average performance to be optimized is a time average defined in (1) (see (11) for its difference equation). Therefore, the algorithms of MDP cannot be directly used to optimize the customer-average performance. This is illustrated by the experiments in Section 5.

4. Policy iteration of customer-average performance

In this section, we develop a policy iteration algorithm for performance optimization of a state-dependent closed Jackson network, where the action is the service rate of each server at each state. We denote $\mathcal{U} = \{\mu_{k,\mathbf{n}}, k = 1, 2, \ldots, M, \mathbf{n} \in \mathcal{S}\}$ as a stationary policy based on parameters $\mu_{k,\mathbf{n}}$. The total policy space is denoted as $\Psi = \{all\ \mathcal{U}\}$, where different $\mu_{k,\mathbf{n}}$'s represent different policies. The cost function associated with policy $\mathcal{U}$ is denoted as $f^{\mathcal{U}}(\mathbf{n})$. With this notation, the cost functions with two policies $\mathcal{U}'$ and $\mathcal{U}$ are $f^{\mathcal{U}'}(\mathbf{n})$ and $f^{\mathcal{U}}(\mathbf{n})$ respectively; for simplicity, they are denoted as $f'(\mathbf{n})$ and $f(\mathbf{n})$ in the previous two sections.

In this section, we just consider a special form of $f^{\mathcal{U}}(\mathbf{n})$, in which the cost function at state $\mathbf{n}$ depends only on the action taken at $\mathbf{n}$, not on the actions taken at other states. We denote it as $f(\mathbf{n}, \vec{\mu}_{\mathbf{n}})$, where $\vec{\mu}_{\mathbf{n}} := (\mu_{1,\mathbf{n}}, \mu_{2,\mathbf{n}}, \ldots, \mu_{M,\mathbf{n}})$ is the action at state $\mathbf{n}$. With this notation, a policy can be denoted as $\mathcal{U} = \{\vec{\mu}_{\mathbf{n}}, \mathbf{n} \in \mathcal{S}\}$. The objective is to minimize the customer-average performance $\eta^{(f)}$.

We show that with the performance difference equations derived in Section 3, a policy iteration algorithm for customer-average performance can be easily developed. The algorithm is based on the realization factors $c^{(f)}(\mathbf{n},k)$, which can be estimated on sample paths, and hence the policy iteration algorithm can be implemented on-line. First, we have the following lemma.

Lemma 4.1. *If $f(\mathbf{n}, \vec{\mu}'_{\mathbf{n}}) - \sum_{k=1}^{M}\frac{\mu'_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}}c^{(f)}(\mathbf{n},k) \le 0$ for all $\mathbf{n} \in \mathcal{S}$, then $\eta'^{(f)} \le \eta^{(f)}$. In addition, if the inequality strictly holds for at least one $\mathbf{n} \in \mathcal{S}$, then $\eta'^{(f)} < \eta^{(f)}$. The lemma holds if we change $\le$ to $\ge$ and $<$ to $>$.*

Since $\eta'^{(l)} > 0$ and $\pi'(\mathbf{n}) > 0$ for all $\mathbf{n} \in \mathcal{S}$, this lemma can be directly obtained from the performance difference equation (25), where $f'(\mathbf{n})$ is replaced by $f(\mathbf{n}, \vec{\mu}'_{\mathbf{n}})$. With Lemma 4.1, we can derive the necessary and sufficient condition for a policy to be optimal.

Theorem 4.1. *A policy $\mathcal{U} = \{\vec{\mu}_{\mathbf{n}}, \mathbf{n} \in \mathcal{S}\}$ is optimal if and only if*

$$f(\mathbf{n}, \vec{\mu}'_{\mathbf{n}}) - \sum_{k=1}^{M} \frac{\mu'_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}}} c^{(f)}(\mathbf{n}, k) \geq 0, \tag{27}$$

for all $\mathcal{U}' = \{\vec{\mu}'_{\mathbf{n}}, \mathbf{n} \in \mathcal{S}\} \in \Psi$ and $\mathbf{n} \in \mathcal{S}$, where $c^{(f)}(\mathbf{n}, k)$, $k = 1, 2, \ldots, M$, $\mathbf{n} \in \mathcal{S}$, are the realization factors for policy $\mathcal{U}$.

This theorem follows directly from Lemma 4.1. With Lemma 4.1 and Theorem 4.1, we can develop a policy iteration algorithm to optimize the service rates of queueing systems. The procedure of the algorithm is described as follows.

Algorithm 1 (*Policy iteration algorithm for customer-average performance of queueing systems*).

(1) *Initialization:* Choose an arbitrary policy $\mathcal{U}^0 \in \Psi$ as an initial policy.
(2) *Improvement:* At the jth iteration with the policy denoted as $\mathcal{U}^j = \{\vec{\mu}_{\mathbf{n}}, \mathbf{n} \in \mathcal{S}\}$, calculate or estimate the realization factors $c^{(f)}(\mathbf{n}, k)$, $k = 1, 2, \ldots, M$, $\mathbf{n} \in \mathcal{S}$, under policy $\mathcal{U}^j$. Choose the policy of the next (the $(j+1)$th) iteration as $\mathcal{U}^{j+1} = \{\vec{\mu}'_{\mathbf{n}}, \mathbf{n} \in \mathcal{S}\}$ with

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

If at a state $\mathbf{n}$, $\vec{\mu}_{\mathbf{n}}$ already reaches the minimum of above large bracket, then set $\vec{\mu}'_{\mathbf{n}} = \vec{\mu}_{\mathbf{n}}$.
(3) *Stopping Rule:* If $\mathcal{U}^{j+1} \neq \mathcal{U}^j$, set $j = j + 1$ and return to step 2; otherwise, stop.

In most cases the service rates $\vec{\mu}_{\mathbf{n}}$ (actions) are discretized and the number of possible action values is not too large, thus the minimization in step 2 can be easily carried out. In addition, even if the realization factors are not accurately estimated, the resulting $\vec{\mu}'_{\mathbf{n}}$ is the same as long as the imprecision does not affect the order of the terms in the large bracket in step 2.

It follows from Lemma 4.1 that the system performance improves in each iteration step. By Theorem 4.1, the policy is optimal when the algorithm stops. If the service rates are chosen from a set of discrete values, i.e., the action space is finite, then the policy space Ψ is also finite. It is easy to show that the iteration algorithm will stop within a finite number of steps since the performance always improves in each iteration, i.e., the algorithm converges to the optimal policy within a finite number of iterations.

In Algorithm 1, the realization factors $c^{(f)}(\mathbf{n}, k)$ are required in each iteration. From the Appendix we can see the realization factors can be obtained by solving a set of linear equations (31)–(33). But, the problem is that the dimension of (33) is $|\mathcal{S}| \times M = \binom{M+N-1}{M-1} \times M$, where the first factor is the number of different ways of putting N customers into M servers, which may be very large in practical systems. Therefore, it is desirable to develop learning algorithms. Moreover, the learning algorithm is also model-free, i.e., it does not require to know the routing probabilities in order to apply the policy iteration algorithm. We observe that the realization factors $c^{(f)}(\mathbf{n}, k)$ can be estimated from a sample path of the queueing network. From the definition of realization factors (3) and PA theory in queueing systems (Cao, 1994), we give the following on-line algorithm for estimating $c^{(f)}(\mathbf{n}, k)$. For simplicity, we only discuss the realization factors of a particular server k. The situations for other servers are similar.

Algorithm 2 (*On-line algorithm for estimating realization factors of server k*).

(1) *Initialization:* Set $\Delta(\mathbf{n}, i) = 0, T(\mathbf{n}) = 0, \Delta F_L(\mathbf{n}) = 0$, for all $\mathbf{n} \in \mathcal{S}, i = 1, 2, \ldots, M$. For a particular state $\mathbf{n}$, $T(\mathbf{n})$ stores the total original perturbations generated at server k when the state is $\mathbf{n}$; $\Delta(\mathbf{n}, i)$ denotes the perturbations of server i, which are propagated to server i from the perturbations originally generated at server k when the state is $\mathbf{n}$; $\Delta F_L(\mathbf{n})$ is the total accumulated perturbations of F_L due to the perturbations generated at server k when the state is $\mathbf{n}$.
(2) *Perturbation generation:* At the service completion time of each server, denote the system state before the customer transition as $\mathbf{n}_1$. The perturbation is generated as follows. Set $\Delta(\mathbf{n}_1, k) = \Delta(\mathbf{n}_1, k) + t(\mathbf{n}_1)$ and $T(\mathbf{n}_1) = T(\mathbf{n}_1) + t(\mathbf{n}_1)$, where $t(\mathbf{n}_1)$ is the time that the system has stayed in state $\mathbf{n}_1$. (Note: actually the generated perturbations should be $\delta t(\mathbf{n}_1)$ where $\delta \ll 1$ is an infinitesimal. But this δ will be eliminated at the final calculation. Thus, for simplicity, we just ignore the infinitesimal δ, see Cao (1994)).
(3) *Perturbation propagation:* When a server i, $1 \leq i \leq M$, completes its current service, denote the system states before and after the customer transition as $\mathbf{n}_1$ and $\mathbf{n}_2$, respectively. For all $\mathbf{n} \in \mathcal{S}$, the perturbation is propagated as follows. If the customer from server i enters server j and terminates the idle period of server j, set $\Delta(\mathbf{n}, j) = \Delta(\mathbf{n}, i)$; otherwise, set $\Delta(\mathbf{n}, j) = (1 - \kappa_j(\mathbf{n}_1, \mathbf{n}_2))\Delta(\mathbf{n}, i) + \kappa_j(\mathbf{n}_1, \mathbf{n}_2)\Delta(\mathbf{n}, j)$, $j = 1, 2, \ldots, M$, where $\kappa_j(\mathbf{n}_1, \mathbf{n}_2) = \mu_{j,\mathbf{n}_1}/\mu_{j,\mathbf{n}_2}$.
(4) *Perturbation realization:* At the same time of step 3, set $\Delta F_L(\mathbf{n}) = \Delta F_L(\mathbf{n}) + [f(\mathbf{n}_1) - f(\mathbf{n}_2)]\,\Delta(\mathbf{n}, i)$, for all $\mathbf{n} \in \mathcal{S}$.

This process continues until the system has served $L \gg 1$ customers. In step 4, there is a minor point to note: When a server i finishes the service of the Lth customer, we simply set $\Delta F_L(\mathbf{n}) = \Delta F_L(\mathbf{n}) + f(\mathbf{n}_1)\Delta(\mathbf{n}, i)$, for all $\mathbf{n} \in \mathcal{S}$. This is because at time T_L, the sample path ends and $\mathbf{n}_2$ does not exist. Finally, the realization factors can be obtained as follows.

$$c^{(f)}(\mathbf{n}, k) = \lim_{L \to \infty} \frac{\Delta F_L(\mathbf{n})}{T(\mathbf{n})}, \quad \text{for all } \mathbf{n} \in \mathcal{S}. \tag{28}$$

The details of this algorithm can be explained using (3) and (33). Readers who are interested may do so by some careful thinking or may refer to Chapter 4 of Cao (1994). With this algorithm, the realization factors can be estimated based on a single sample path. The algorithm does not require the explicit knowledge of the routing probabilities of queueing networks. Combining this algorithm and the policy iteration algorithm together, we can implement the policy iteration for customer-average performance on-line.

With the Algorithms 1 and 2, this paper proposes a policy iteration algorithm on the customer-average performance optimization in queueing systems with realization factors. This links perturbation analysis with policy iteration and creates a new research direction. Our new policy iteration algorithm is expected to converge in a small number of iterations, similarly to how policy iteration algorithms in Markov systems have been observed to converge in a small number of iterations under the time-average or discounted cost criteria. Moreover, we implement this policy iteration algorithm in an on-line optimization manner.

However, similar to the policy iteration in classical MDP theory, this algorithm also suffers some problems. The first one is the curse of dimensionality. When the system size increases, the state space will grow exponentially and the step 2 of Algorithm 1 will be computation intensive. Moreover, for a large scale problem, the number of realization factors will be very large and the on-line estimation Algorithm 2 will be not very precise, especially when the sample path is not long enough.

Table 1
The numerical results of the on-line estimation Algorithm 2.

n	$\mu_{1,\mathbf{n}}$	$\mu_{2,\mathbf{n}}$	$\mu_{3,\mathbf{n}}$	$c^{(f)}(\mathbf{n},1)$	$c^{(f)}(\mathbf{n},2)$	$c^{(f)}(\mathbf{n},3)$	$\hat{c}^{(f)}(\mathbf{n},1)$	$\hat{c}^{(f)}(\mathbf{n},2)$	$\hat{c}^{(f)}(\mathbf{n},3)$	SD of $\hat{c}^{(f)}(\mathbf{n},1)$
(0,0,5)	0.10	0.10	0.40	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(0,1,4)	0.10	0.10	0.35	0.0000	−0.0413	0.0413	0.0000	−0.0348	0.0348	0.0000
(0,2,3)	0.10	0.20	0.30	0.0000	0.0308	−0.0308	0.0000	0.0350	−0.0350	0.0000
(0,3,2)	0.10	0.30	0.25	0.0000	0.2183	−0.2183	0.0000	0.2189	−0.2189	0.0000
(0,4,1)	0.10	0.35	0.10	0.0000	0.1581	−0.1581	0.0000	0.1570	−0.1570	0.0000
(0,5,0)	0.10	0.50	0.10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(1,0,4)	0.10	0.10	0.45	1.0002	0.0000	−0.0002	1.0012	0.0000	−0.0012	0.0103
(1,1,3)	0.15	0.15	0.35	1.4039	−0.3195	−0.0844	1.4049	−0.3191	−0.0857	0.0084
(1,2,2)	0.20	0.20	0.30	1.7893	−0.2815	−0.5077	1.7881	−0.2808	−0.5073	0.0084
(1,3,1)	0.15	0.35	0.15	1.4635	−0.1052	−0.3583	1.4622	−0.1035	−0.3587	0.0069
(1,4,0)	0.10	0.40	0.10	1.0697	−0.0697	0.0000	1.0698	−0.0698	0.0000	0.0055
(2,0,3)	0.15	0.10	0.30	1.7549	0.0000	0.2451	1.7596	0.0000	0.2404	0.0114
(2,1,2)	0.20	0.15	0.25	2.2307	−0.1250	−0.1057	2.2298	−0.1254	−0.1044	0.0082
(2,2,1)	0.20	0.25	0.15	2.2143	0.0907	−0.3050	2.2126	0.0929	−0.3055	0.0052
(2,3,0)	0.15	0.35	0.10	1.7149	0.2851	0.0000	1.7153	0.2847	0.0000	0.0053
(3,0,2)	0.30	0.10	0.20	2.9543	0.0000	0.0457	2.9545	0.0000	0.0455	0.0061
(3,1,1)	0.35	0.10	0.10	3.3355	−0.1061	−0.2294	3.3352	−0.1061	−0.2292	0.0067
(3,2,0)	0.30	0.20	0.10	3.0622	−0.0622	0.0000	3.0619	−0.0619	0.0000	0.0062
(4,0,1)	0.40	0.10	0.10	4.1934	0.0000	−0.1934	4.1939	0.0000	−0.1939	0.0070
(4,1,0)	0.35	0.15	0.10	4.0255	−0.0255	0.0000	4.0292	−0.0292	0.0000	0.0052
(5,0,0)	0.45	0.10	0.10	5.0000	0.0000	0.0000	5.0000	0.0000	0.0000	0.0000

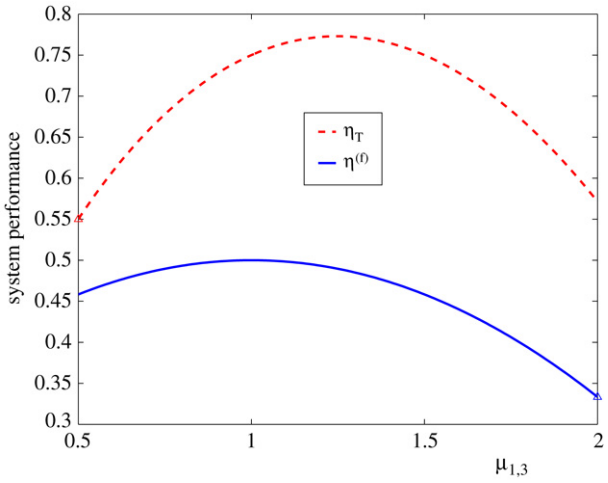


Fig. 1. The difference between customer-average performance and time-average performance.

Such imprecision of realization factors may affect the optimization result of Algorithm 1. Furthermore, our algorithm is to optimize the state-dependent service rates. If the service rates do not depend on states, but have other constraints, the optimization will be much more complex. For example, if the service rates are load-dependent, the action is to choose the service rates $\mu_{k,n_k}, k = 1, 2, \ldots, M$ and $n_k = 1, 2, \ldots, N$. The service rates of a server are identical at many states when the numbers of customers at this server are identical. It is equivalent to introducing constraints on the control variables and is similar to the crucial problem in MDP where the actions at different states are dependent (Puterman, 1994). The policy iteration algorithm developed in this section is not applicable for this situation, and we may apply the event-based optimization approach proposed in Cao (2005). All these problems are common in MDP and will be the future research topics.

5. Numerical experiments

In this section, we give numerical experiments to demonstrate the efficiency of the policy iteration algorithm and the on-line estimation algorithm for realization factors.

5.1. Experiments of on-line algorithm for estimating realization factors

Consider a state-dependent closed Jackson network with $M = 3$ and $N = 5$. The routing probability matrix is

$$Q = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.8 & 0 & 0.2 \\ 0.3 & 0.7 & 0 \end{bmatrix}.$$

The service rates $\mu_{k,\mathbf{n}}$ are listed in Table 1. The cost function is defined as $f(\mathbf{n}) = n_1$. Let L_1 be the number of customers served by server 1 in the period $[0, T_L)$. It is easy to show that the customer-average performance is

$$\eta^{(f)} = \lim_{L\to\infty} \frac{F_L}{L} = \lim_{L\to\infty} \frac{L_1}{L} \lim_{L\to\infty} \frac{F_L}{L_1} = v_1 \bar{\tau}_1, \tag{29}$$

where v_1 is the visit ratio of server 1, $\bar{\tau}_1$ is the (customer-)average response time of each customer at server 1. For any $i, j = 1, \ldots, M$, v_i/v_j is the ratio of the numbers that a particular customer visiting server i and server j in a long period of time. The vector of visit ratios v can be determined by the equations $vQ = v$ and $ve = 1$ (Chen & Yao, 2001), it has nothing to do with the service rates. So, from (29) we can see that the customer-average performance $\eta^{(f)}$ is equal to $\bar{\tau}_1$ multiplied by a fixed number v_1, thus $\eta^{(f)}$ is proportional to the average response time of each customer at server 1, which is an important performance metric of queueing systems. Actually, for the parameter setting of this experiment, we can solve the equations $vQ = v$ and $ve = 1$, and obtain $v = (0.3723, 0.3680, 0.2597)$, thus $\eta^{(f)} = 0.3723\bar{\tau}_1$. On the other hand, if we consider the time-average performance, we can see

$$\eta_T = \lim_{L\to\infty} \frac{F_L}{T_L} = \bar{n}_1, \tag{30}$$

where $\bar{n}_1$ is the average queue length at server 1. $\bar{n}_1$ is also an important metric of queueing systems. Therefore, these two performance metrics, $\bar{\tau}_1$ and $\bar{n}_1$, are both important for queueing systems. The optimal solutions for these two performance metrics are not the same. This will be illustrated by the following numerical examples in Section 5.2.

We use two methods to obtain the realization factors. One is to solve the linear equations (31)–(33) in the Appendix. The other is to use the on-line estimation Algorithm 2 based on a single sample path. The total simulation duration is $L = 100{,}000$. The results are listed in Table 1. The mean value $(\hat{c}^{(f)}(\mathbf{n}, k), k = 1, 2, 3)$

and standard deviations (SD) are the results of 10 replications. For simplicity, we only list the SD of $\hat{c}^{(f)}(\mathbf{n}, 1)$. The simulation results demonstrate that the on-line Algorithm 2 has a quite good estimation accuracy.

5.2. Experiments of policy iteration algorithm for customer-average performance

In this section, we implement the policy iteration algorithm for the customer-average performance of queueing systems. The parameter setting is the same as that of Section 5.1. The cost function is $f(\mathbf{n}) = n_1 + \sum_{k=1}^{3} \mu_{k,\mathbf{n}}$, which reflects the cost for the average response time at server 1 and the cost for providing service rates $\mu_{k,\mathbf{n}}$, $k = 1, 2, 3$. The objective is to minimize the customer-average performance $\eta^{(f)}$ through adjusting the service rates $\mu_{k,\mathbf{n}}$, $k = 1, 2, 3$ and $\mathbf{n} \in \mathcal{S}$. The value domains of service rates are listed as $D_{\mu_{k,\mathbf{n}}}$ in Table 2. The initial service rates $\mu_{k,\mathbf{n}}$ are preset as the values in Table 1. In fact, based on the difference equation (21) we can prove that the optimal values of service rates can be either maximal or minimal. Thus, we only need to choose $\mu^{max}_{k,\mathbf{n}}$ or $\mu^{min}_{k,\mathbf{n}}$, $k = 1, 2, 3$ and $\mathbf{n} \in \mathcal{S}$. The size of the policy space is $2^{|\mathcal{S}|\times M} = 2^{21\times 3} \approx 10^{19}$, which is still very large.

With the estimated realization factors, we implement Algorithm 1 to optimize the service rates. From the simulation results, we find that the algorithm only iterates 4 times to obtain the optimal service rates (we have verified that this optimized service rates are really the optimum, the same as the theoretical values). This illustrates that Algorithm 1 has a very fast convergence speed. The optimal service rates $\mu^{*}_{k,\mathbf{n}}$ are listed in Table 2 and the optimal customer-average performance is $\eta^{*(f)} = 1.1827$.

As a comparison, we use the traditional MDP algorithms to optimize the time-average performance of this problem. In this situation, we obtain the optimal service rates $\mu'_{k,\mathbf{n}}$ which are listed in Table 2. The corresponding $\eta^{(f)} = 1.4119$, which is worse than the optimal value $\eta^{*(f)} = 1.1827$. The service rates $\mu'_{k,\mathbf{n}}$ are also different from the optimal service rates $\mu^{*}_{k,\mathbf{n}}$ for customer-average performance. This example illustrates that the algorithm for time-average performance cannot be directly used to optimize the customer-average performance.

Furthermore, in order to more clearly illustrate the difference between customer-average performance and time-average performance, we give another numerical example as follows. We consider a cyclic network with 2 servers. The number of customers is $N = 3$. For simplicity, we can use n_1 to represent the system state $\mathbf{n}$. The service rates of server 2 is fixed at $\mu_{2,n_1} = 1$, $n_1 = 0, 1, 2, 3$. The service rates of server 1 is $\mu_{1,n_1} = 1$ when $n_1 = 0, 1, 2$, and $\mu_{1,n_1} \in [0.5, 2]$ when $n_1 = 3$. The cost function is $f(0) = f(1) = f(2) = 1, f(3) = -\mu_{1,3}(1 - \mu_{1,3})^2$. The objective is to choose an optimal $\mu_{1,3}$ to minimize the system performance $\eta^{(f)}$. With numerical computation, we obtain the performance curve as Fig. 1. It is obvious that the optimal $\mu_{1,3}$ is 2 for $\eta^{(f)}$, while the optimal $\mu_{1,3}$ is 0.5 for η_T. This example clearly illustrates the difference between these two performance metrics.

6. Discussion and conclusion

In this paper, we first derived the performance difference equations and then developed a policy iteration algorithm for customer-average performance in state-dependent closed Jackson networks. We also developed a sample path based algorithm for estimating the realization factors so that the policy iteration algorithm can be implemented on-line.

Policy iteration based on realization factors is new in the customer-average performance optimization of queueing systems and may bring new insights for future research. For example, aggregation techniques such as the event-based optimization approach (Cao, 2005) may be explored in the setting of queueing systems. This may lead to a solution to the optimization of load-dependent closed Jackson networks, where the service rates of a sever depend on the number of customers at the server.

Acknowledgments

Second author was supported in part by the National Natural Science Foundation (60574064) of China. Third author was supported in part by a grant from Hong Kong RGC.

Appendix. Proof of Theorem 3.1

First, we give a set of linear equations which are proved to be able to uniquely determine the realization factors $c^{(f)}(\mathbf{n}, k)$ (Cao, 1994).

$$\text{If } n_k = 0, \quad \text{then } c^{(f)}(\mathbf{n}, k) = 0; \tag{31}$$

$$\sum_{k=1}^{M} c^{(f)}(\mathbf{n}, k) = f(\mathbf{n}); \tag{32}$$

and

$$\begin{aligned}
&\left\{\sum_{i=1}^{M} \epsilon(n_i)\mu_{i,\mathbf{n}}\right\} c^{(f)}(\mathbf{n}, k) \\
&= \sum_{i=1, i\neq k}^{M} \sum_{j=1}^{M} \epsilon(n_i)\mu_{i,\mathbf{n}} q_{ij} \kappa_k(\mathbf{n}, \mathbf{n}_{ij}) c^{(f)}(\mathbf{n}_{ij}, k) \\
&\quad + \sum_{j=1}^{M} \mu_{k,\mathbf{n}} q_{kj} \Bigg\{ c^{(f)}(\mathbf{n}_{kj}, k) + \sum_{i=1, i\neq k}^{M} \left[1 - \kappa_i(\mathbf{n}, \mathbf{n}_{kj})\right] \epsilon(n_i) \\
&\quad c^{(f)}(\mathbf{n}_{kj}, i) + \left[1 - \epsilon(n_j)\right] c^{(f)}(\mathbf{n}_{kj}, j) + f(\mathbf{n}) - f(\mathbf{n}_{kj}) \Bigg\},
\end{aligned} \tag{33}$$

where $\mathbf{n}_{kj} = (n_1, \ldots, n_k - 1, \ldots, n_j + 1, \ldots, n_M)$ is a neighboring state of $\mathbf{n}$ with $n_k > 0$, $\kappa_i(\mathbf{n}, \mathbf{n}_{kj}) = \mu_{i,\mathbf{n}}/\mu_{i,\mathbf{n}_{kj}}$, $\epsilon(n_i)$ is an indicator function which is defined as: if $n_i > 0$, $\epsilon(n_i) = 1$; otherwise $\epsilon(n_i) = 0$.

Eq. (31) means that when the server k is idle, the perturbation of its service time will have no effect on the whole network. Eq. (32) means that if the all the service time of servers have a same delay Δ, it equals the effect of the whole sample path being shifted by time Δ, thus the induced performance perturbation is $f(\mathbf{n})\Delta$ and the sum of realization factors are $f(\mathbf{n})$. Eq. (33) is more complex and it describes how the perturbation of a server k is propagated to other servers. The first term of right-hand side of (33) measures the system effect if other servers (not server k) first finished its current customer's service. The second term of right-hand side of (33) measures the performance effect if server k first finished its current customer's service. The detailed explanations relate to the principle of perturbation propagation in PA theory, which can be found in Theorem 4.1 of Cao (1994).

In order to prove Theorem 3.1, we only need to check that $c^{(f)}(\mathbf{n}, k) = \epsilon(n_k)\mu_{k,\mathbf{n}}$, $k = 1, 2, \ldots, M$, $\mathbf{n} \in \mathcal{S}$, satisfy the above Eqs. (31)–(33).

1. If $n_k = 0$, then $c^{(f)}(\mathbf{n}, k) = \epsilon(n_k)\mu_{k,\mathbf{n}} = 0$. Thus, (31) holds.

2. We have $\sum_{k=1}^{M} c^{(f)}(\mathbf{n}, k) = \sum_{k=1}^{M} \epsilon(n_k)\mu_{k,\mathbf{n}} = f(\mathbf{n})$. Thus, (32) holds.

3. For Eq. (33), we substitute $c^{(f)}(\mathbf{n}, k) = \epsilon(n_k)\mu_{k,\mathbf{n}}$ into this equation and verify that both sides are indeed equal. The first term

Table 2
The optimal solution of the policy iteration Algorithm 1.

n	$D_{\mu_{1,\mathbf{n}}}$	$D_{\mu_{2,\mathbf{n}}}$	$D_{\mu_{3,\mathbf{n}}}$	$\mu^*_{1,\mathbf{n}}$	$\mu^*_{2,\mathbf{n}}$	$\mu^*_{3,\mathbf{n}}$	$\mu'_{1,\mathbf{n}}$	$\mu'_{2,\mathbf{n}}$	$\mu'_{3,\mathbf{n}}$
(0,0,5)	[0.01,1.5]	[0.04,1.0]	[0.09,4.0]	0.01	0.04	4.00	0.01	0.04	0.09
(0,1,4)	[0.01,1.5]	[0.03,1.0]	[0.06,4.5]	0.01	0.03	4.50	0.01	0.03	0.06
(0,2,3)	[0.02,1.0]	[0.05,1.5]	[0.03,3.5]	0.02	0.05	3.50	0.02	0.05	0.03
(0,3,2)	[0.03,1.5]	[0.07,1.0]	[0.02,2.5]	0.03	0.07	2.50	0.03	0.07	0.02
(0,4,1)	[0.04,2.0]	[0.09,1.8]	[0.05,3.0]	0.04	0.09	3.00	0.04	0.09	0.05
(0,5,0)	[0.05,1.0]	[0.05,2.6]	[0.01,2.0]	0.05	2.60	0.01	0.05	0.05	0.01
(1,0,4)	[0.05,2.0]	[0.01,2.0]	[0.04,1.0]	2.00	0.01	0.04	2.00	0.01	0.04
(1,1,3)	[0.05,1.5]	[0.02,2.5]	[0.09,1.5]	1.50	0.02	0.09	1.50	0.02	0.09
(1,2,2)	[0.02,1.8]	[0.03,2.0]	[0.05,1.5]	1.80	0.03	0.05	1.80	0.03	0.05
(1,3,1)	[0.06,2.0]	[0.05,3.0]	[0.02,2.5]	2.00	0.05	0.02	2.00	0.05	0.02
(1,4,0)	[0.04,2.4]	[0.06,1.5]	[0.08,4.5]	2.40	0.06	0.08	2.40	0.06	0.08
(2,0,3)	[0.08,1.8]	[0.03,3.0]	[0.10,2.0]	1.80	0.03	0.10	1.80	0.03	0.10
(2,1,2)	[0.09,1.0]	[0.05,4.5]	[0.07,2.0]	1.00	0.05	2.00	1.00	0.05	0.07
(2,2,1)	[0.10,2.0]	[0.10,1.0]	[0.03,3.0]	2.00	0.10	3.00	2.00	0.10	0.03
(2,3,0)	[0.07,2.5]	[0.03,3.5]	[0.10,4.5]	2.50	0.03	0.10	2.50	0.03	0.10
(3,0,2)	[0.03,3.0]	[0.05,2.0]	[0.01,2.0]	3.00	0.05	2.00	3.00	0.05	0.01
(3,1,1)	[0.05,3.5]	[0.10,2.0]	[0.08,4.5]	3.50	0.10	4.50	3.50	0.10	0.08
(3,2,0)	[0.09,4.0]	[0.06,1.5]	[0.03,2.5]	4.00	0.06	0.03	4.00	0.06	0.03
(4,0,1)	[0.10,3.0]	[0.09,2.5]	[0.06,1.0]	3.00	0.09	1.00	3.00	0.09	0.06
(4,1,0)	[0.08,4.5]	[0.05,3.5]	[0.09,1.5]	4.50	0.05	0.09	4.50	0.05	0.09
(5,0,0)	[0.09,5.0]	[0.04,2.0]	[0.02,1.0]	5.00	0.04	0.02	5.00	0.04	0.02

of the right-hand side of (33) is

$$\sum_{i=1,i\neq k}^{M}\sum_{j=1}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}q_{ij}\kappa_k(\mathbf{n},\mathbf{n}_{ij})c^{(f)}(\mathbf{n}_{ij},k)$$
$$=\sum_{i=1,i\neq k}^{M}\sum_{j=1}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}q_{ij}\frac{\mu_{k,\mathbf{n}}}{\mu_{k,\mathbf{n}_{ij}}}\epsilon(\mathbf{n}_{ij(k)})\mu_{k,\mathbf{n}_{ij}}, \tag{34}$$

where $\mathbf{n}_{ij(k)}$ is the number of customers at server k when the system state is $\mathbf{n}_{ij}$. In fact, $\mathbf{n}_{(k)}=n_k$. Since $n_k>0$ and $i\neq k$ in (34), it is obvious that $\epsilon(\mathbf{n}_{ij(k)})=1$. So, the first term of the right-hand side of (33) is

$$\sum_{i=1,i\neq k}^{M}\sum_{j=1}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}q_{ij}\kappa_k(\mathbf{n},\mathbf{n}_{ij})c^{(f)}(\mathbf{n}_{ij},k)$$
$$=\sum_{i=1,i\neq k}^{M}\sum_{j=1}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}q_{ij}\mu_{k,\mathbf{n}}$$
$$=\sum_{i=1,i\neq k}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}\mu_{k,\mathbf{n}}. \tag{35}$$

For the second term of the right-hand side of (33), we have

$$\sum_{j=1}^{M}\mu_{k,\mathbf{n}}q_{kj}\Big\{c^{(f)}(\mathbf{n}_{kj},k)+\sum_{i=1,i\neq k}^{M}\big[1-\kappa_i(\mathbf{n},\mathbf{n}_{kj})\big]\epsilon(n_i)$$
$$c^{(f)}(\mathbf{n}_{kj},i)+\big[1-\epsilon(n_j)\big]c^{(f)}(\mathbf{n}_{kj},j)+f(\mathbf{n})-f(\mathbf{n}_{kj})\Big\}$$
$$=\sum_{j=1}^{M}\mu_{k,\mathbf{n}}q_{kj}\Big\{\epsilon(\mathbf{n}_{kj(k)})\mu_{k,\mathbf{n}_{kj}}+\sum_{i=1,i\neq k}^{M}\Big[1-\frac{\mu_{i,\mathbf{n}}}{\mu_{i,\mathbf{n}_{kj}}}\Big]$$
$$\epsilon(n_i)\epsilon(\mathbf{n}_{kj(i)})\mu_{i,\mathbf{n}_{kj}}+\big[1-\epsilon(n_j)\big]\epsilon(\mathbf{n}_{kj(j)})\mu_{j,\mathbf{n}_{kj}}$$
$$+\sum_{i=1}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}-\sum_{i=1}^{M}\epsilon(\mathbf{n}_{kj(i)})\mu_{i,\mathbf{n}_{kj}}\Big\}$$
$$=\sum_{j=1}^{M}\mu_{k,\mathbf{n}}q_{kj}\Big\{\epsilon(\mathbf{n}_{kj(k)})\mu_{k,\mathbf{n}_{kj}}+\sum_{i=1,i\neq k}^{M}\Big[1-\frac{\mu_{i,\mathbf{n}}}{\mu_{i,\mathbf{n}_{kj}}}\Big]$$
$$\epsilon(n_i)\mu_{i,\mathbf{n}_{kj}}+\big[1-\epsilon(n_j)\big]\mu_{j,\mathbf{n}_{kj}}$$
$$+\sum_{i=1}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}-\sum_{i=1}^{M}\epsilon(\mathbf{n}_{kj(i)})\mu_{i,\mathbf{n}_{kj}}\Big\}$$
$$=\sum_{j=1}^{M}\mu_{k,\mathbf{n}}q_{kj}\Big\{\epsilon(\mathbf{n}_{kj(k)})\mu_{k,\mathbf{n}_{kj}}+\sum_{i=1,i\neq k}^{M}\big[\mu_{i,\mathbf{n}_{kj}}-\mu_{i,\mathbf{n}}\big]$$
$$\epsilon(n_i)+\big[1-\epsilon(n_j)\big]\mu_{j,\mathbf{n}_{kj}}+\sum_{i=1,i\neq k}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}$$
$$+\epsilon(n_k)\mu_{k,\mathbf{n}}-\sum_{i=1,i\neq k}^{M}\epsilon(\mathbf{n}_{kj(i)})\mu_{i,\mathbf{n}_{kj}}-\epsilon(\mathbf{n}_{kj(k)})\mu_{k,\mathbf{n}_{kj}}\Big\}$$
$$=\sum_{j=1}^{M}\mu_{k,\mathbf{n}}q_{kj}\Big\{\sum_{i=1,i\neq k}^{M}\big[\mu_{i,\mathbf{n}_{kj}}-\mu_{i,\mathbf{n}}\big]\epsilon(n_i)$$
$$+\big[1-\epsilon(n_j)\big]\mu_{j,\mathbf{n}_{kj}}+\sum_{i=1,i\neq k}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}+\mu_{k,\mathbf{n}}$$
$$-\sum_{i=1,i\neq k,j}^{M}\epsilon(\mathbf{n}_{kj(i)})\mu_{i,\mathbf{n}_{kj}}-\epsilon(\mathbf{n}_{kj(j)})\mu_{j,\mathbf{n}_{kj}}\Big\}$$
$$=\sum_{j=1}^{M}\mu_{k,\mathbf{n}}q_{kj}\Big\{\sum_{i=1,i\neq k}^{M}\big[\mu_{i,\mathbf{n}_{kj}}-\mu_{i,\mathbf{n}}\big]\epsilon(n_i)$$
$$+\big[1-\epsilon(n_j)\big]\mu_{j,\mathbf{n}_{kj}}+\sum_{i=1,i\neq k}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}+\mu_{k,\mathbf{n}}$$
$$-\sum_{i=1,i\neq k,j}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}_{kj}}-\mu_{j,\mathbf{n}_{kj}}\Big\}$$
$$=\sum_{j=1}^{M}\mu_{k,\mathbf{n}}q_{kj}\Big\{\sum_{i=1,i\neq k}^{M}\big[\mu_{i,\mathbf{n}_{kj}}-\mu_{i,\mathbf{n}}\big]\epsilon(n_i)$$
$$+\sum_{i=1,i\neq k}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}+\mu_{k,\mathbf{n}}-\sum_{i=1,i\neq k}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}_{kj}}\Big\}$$
$$=\sum_{j=1}^{M}\mu_{k,\mathbf{n}}q_{kj}\mu_{k,\mathbf{n}}=\mu_{k,\mathbf{n}}\mu_{k,\mathbf{n}}. \tag{36}$$

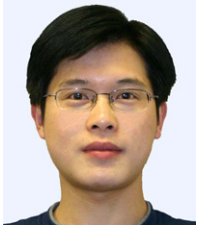
The summation of the two terms of the right-hand side of (33) is $\sum_{i=1,i\neq k}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}\mu_{k,\mathbf{n}}+\mu_{k,\mathbf{n}}\mu_{k,\mathbf{n}}=\sum_{i=1}^{M}\epsilon(n_i)\mu_{i,\mathbf{n}}\mu_{k,\mathbf{n}}$, which is equal to the left-hand side of (33). So, $c^{(f)}(\mathbf{n},k)=\epsilon(n_k)\mu_{k,\mathbf{n}}$, $k=1,2,\ldots,M$, $\mathbf{n}\in\mathcal{S}$, also satisfy (33), and Theorem 3.1 is proved. □

References

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

Cao, X. R. (2005). Basic ideas for event-based optimization of Markov systems. *Discrete Event Dynamic Systems: Theory and Applications*, *15*, 169–197.

Cao, X. R., & Chen, H. F. (1997). Potentials, perturbation realization, and sensitivity analysis of Markov processes. *IEEE Transactions on Automatic Control*, *42*, 1382–1393.
Cassandras, C., & Lafortune, S. (1999). *Introduction to discrete event systems*. Boston, MA: Kluwer Academic Publishers.
Chen, H., & Yao, D. D. (2001). *Fundamentals of queueing networks: Performance, asymptotics, and optimization*. New York: Springer Verlag.
Chong, E. K. P., & Zak, S. H. (2001). *An introduction to optimization* (2nd ed.). New York: John Wiley & Sons.
Glasserman, P. (1991). *Gradient estimation via perturbation analysis*. Norwell, MA: Kluwer Academic Publisher.
Gong, W. B., & Ho, Y. C. (1987). Smoothed (conditional) perturbation analysis of discrete event dynamical 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.
Ho, Y. C., & Cao, X. R. (1991). *Perturbation analysis of discrete-event dynamic systems*. Boston, MA: Kluwer Academic Publisher.
Ho, Y. C., Cao, X. R., & Cassandras, C. (1983). Infinitesimal and finite perturbation analysis for queueing networks. *Automatica*, *19*, 439–445.
Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York: John Wiley & Sons.
Xia, L., & Cao, X. R. (2006). Relationship between perturbation realization factors with queueing models and Markov models. *IEEE Transactions on Automatic Control*, *51*, 1699–1704.



Li Xia received the B.E. degree in automation, in July 2002, and the Ph.D. degree in control science and engineering, in July 2007, both from Tsinghua University, Beijing, China. He is currently a member of research staff at IBM China Research Laboratory, Beijing, China. His research interests include discrete event dynamic systems (DEDS) theory and applications, simulation-based optimization techniques, and customer relationship management.

Xi Chen received her B. Sc. and M. Eng. from Nankai University, Tianjin, China, in 1986 and 1989, respectively. After graduation, she worked in the Software Engineering Institute at Beijing University of Aeronautics and Astronautics for seven years. From October 1996 she studied in the Chinese University of Hong Kong and received her Ph.D. in 2000. Then she worked as a post-doctoral fellow in Information Communication Institute of Singapore and in the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong. Since July 2003, she has worked in the Center for Intelligent and Networked Systems (CFINS), Department of Automation, Tsinghua University, Beijing, China. Her research interests include wireless sensor networks and stochastic control.

Xi-Ren Cao received the M.S. and Ph.D. degrees from Harvard University, in 1981 and 1984, respectively, where he was a research fellow from 1984 to 1986. He then worked as consultant engineer/engineering manager at Digital Equipment Corporation, U.S.A, until October 1993. Then he joined the Hong Kong University of Science and Technology (HKUST), where he is currently Chair Professor, and Director of the Research Center for Networking.

Dr. Cao owns three patents in data- and telecommunications and published three books in the area of performance optimization and discrete event dynamic systems. He received the Outstanding Transactions Paper Award from the IEEE Control System Society in 1987, the Outstanding Publication Award from the Institution of Management Science in 1990, and the Outstanding Service Award from IFAC in 2008. He is a Fellow of IEEE, a Fellow of IFAC, and is/was the Chairman of IEEE Fellow Evaluation Committee of IEEE Control System Society, Editor-in-Chief of Discrete Event Dynamic Systems: Theory and Applications, Associate Editor at Large of IEEE Transactions of Automatic Control, and Board of Governors of IEEE Control Systems Society and on the Technical Board of IFAC. His current research areas include discrete event dynamic systems, stochastic learning and optimization, performance analysis of communication systems, signal processing, and financial engineering.