# Perturbation Analysis, Dynamic Programming, and Beyond

**Chapter** · August 2013

DOI: 10.1142/9789814513012_0007



**2 authors:**

Li Xia
Tsinghua University
**43** PUBLICATIONS **183** CITATIONS
SEE PROFILE

Xi-Ren Cao
The Hong Kong University of Science and Tech…
**207** PUBLICATIONS **4,178** CITATIONS
SEE PROFILE

# Chapter 7

# Perturbation Analysis, Dynamic Programming, and Beyond

Li Xia

*Center for Intelligent and Networked Systems*
*Department of Automation, TNList, Tsinghua University*
*Beijing 100084, China*

Xi-Ren Cao

*Department of Finance and Department of Automation*
*Shanghai Jiao Tong University, China, and*
*Institute of Advanced Study*
*Hong Kong University of Science and Technology, China**

The main idea of perturbation analysis is that a sample path of a discrete event dynamic system contains information about not only the performance itself, but also its derivative. The development of recent 30+ years has testified the power of this insightful idea. It is now clear that a sample path of a Markov system under a policy contains some information about the performance of the system under other policies; and with this view, policy iteration is simply a discrete version of the gradient descent method, and each iteration can be implemented based on a single sample path. This view leads to the direct-comparison based approach to performance optimization, which may be applied to cases where dynamics programming does not work well. In this chapter dedicated to Prof. Ho's 80th birthday, we document the research along this direction, especially those beyond dynamic programming.

*This research was supported in part by the Collaborative Research Fund of the Research Grants Council, Hong Kong Special Administrative Region, China, under Grant No. HKUST11/CRF/10, the 111 International Collaboration Project (B06002), National Natural Science Foundation of China (61203039, 60736027), the Specialized Research Fund for the Doctoral Program of Higher Education (20120002120009), Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation.

## 7.1. Introduction

Perturbation analysis (PA) was first proposed for queueing systems by Prof. Y. C. Ho et al., in early 80's last century [29, 32] and its main idea was to *estimate the performance derivatives with respect to system parameters by analyzing a single sample path of a stochastic dynamic system*. It indicates that a system's sample path contains the information not only for the system performance, but also for the performance derivatives.

The development in the past three decades has testified the power of this highly innovative idea. In this chapter dedicated to Prof. Ho's 80th birthday, we briefly document the research development starting from PA that the authors have been involved, and we hope this may illustrate, from one angle, the deep influence of Prof. Ho's insightful initial work. The chapter represents our own understanding; and there are many other important works presented in other chapters.

The early works of PA focus on queueing systems and most of them are with a finite horizon performance; the approach was to develop an efficient algorithm to calculate the sample derivative as an estimate for the derivative of the average performance [29]. It was realized in mid 80's that for the PA-based sample derivative obtained from a single sample path to be an unbiased estimate, it requires conditions for exchanging the order of expectation and derivatives [2]. Since then, one of the major research direction is developing efficient algorithms for the sample derivatives for various systems, proving the unbiasedness of the sample derivative as the estimate of the derivatives of the average performance, and developing new techniques when the unbiasedness does not hold. There have been many excellent papers and books in this research direction, including [3, 5, 18, 19, 21–24, 26, 28, 30, 31], and we certainly cannot cite all of them. We will not review them in details except mentioning here that the idea is recently applied to analyze the performance of non-linear behavior in financial engineering, and a new property called *mono-linearity* is discovered, leading to some new insights to this subject [14].

In this chapter, we mainly focus on another research direction starting from PA. The main ideas leading to this direction are as follows.

First, the effect of a parameter change on a performance can be decomposed into the sum of that of a sequence of single perturbations, which can be measured by a quantity called perturbation realization factor. When the perturbation generated due to the parameter change is infinitesimal, this decomposition leads to the performance derivative, and when the

perturbation generated is finite, it leads to the performance difference. This idea applies to both queueing systems [2, 4, 6, 23, 29] and general Markov systems [13, 17], both finite and infinite horizon performance. For Markov systems, the realization factor of a perturbation, which represents a jump from one state to another, equals the difference of the performance potentials of these two states, the latter measure the "contribution" of a state to the system performance. This approach overcomes the difficulty involved with the exchangeability.

The second idea is a fundamental observation: Because the policy space is usually extremely large, exhaustive search for an optimal policy is infeasible. Therefore, if we do not know anything about the property of the performance curve on the policy space, to develop feasible methods for performance optimization we need to compare the performance of any two policies by analyzing one of them. (If comparing two policies requires to analyze both of them, then comparing all the policies requires to analyze all of them, equivalent to enumerating search in the policy space.) The realization factor approach provides a simple way to implement this single policy based comparison. The realization factors can be determined by analyzing one of the policies. With the realization factors, the original idea of PA, i.e., a system's sample path contains the information not only for the system performance, but also for the performance derivatives, extends to finite parameter changes, i.e., under some mild conditions such as the Markov structure, a system's sample path under a policy also contains some information about the performance under other policies. This establishes a new approach for the performance optimization, which is simply based on a *direct comparison* of the performance of any two policies. This approach is based on the performance difference equation, with no dynamic programming. Compared with the standard dynamic programming approach, this approach is simple and intuitive, and may be applied to problems where dynamic programming fails to work.

In this chapter, we mainly review the research along the direction with the direct-comparison based approach. This approach was first applied to Markov decision processes (MDPs) and it was shown that the standard results such as HJB optimization equations and policy iteration, etc., can be easily established, and Q-learning and other techniques can be derived; in addition, the study has led to a new topic, gradient-based learning (called policy gradient in the reinforcement learning community). Furthermore, because of its simplicity and intuitiveness, the approach also motivates the study of many problems with relatively complex performance, such

as the $N$-bias and the sample-path based variance. The $N$-bias optimality eventually leads to the Blackwell optimality as $N$ goes to infinity [12, 16, 39]; the final results are equivalent to the $N$-discount optimality [41, 49], but the approach is simple and more direct in the sense of no discounting.

The standard dynamic programming requires two assumptions, among others; it assumes that the actions taken at different states can be chosen independently, and that the performance to be optimized is time consistent; that is, the optimal policy for the performance starting from time $t+1$ is also optimal for the performance starting from $t$. However, many practical problems, such as problems involving aggregation or event-triggering actions, the independent-action assumption is violated. We have applied the direct-comparison based approach to study these problems. The study on time-inconsistent performance is underway.

The direct-comparison based approach also motivates some new ideas in the well-established subjects, such as the optimization of queueing systems. There are already rich studies on the optimal control of queueing systems. The traditional approaches usually model the optimization problem of Markovian queues as a Markov decision process and use the optimality equation to analyze and develop the related algorithms. However, for the complex queueing systems, such as the queueing networks, the associated optimality equation is too complicated for effective analysis. By applying the direct-comparison based approach, we can derive the difference equation for queueing systems, which is more efficient than the derivative equation in the traditional PA theory [44]. Direct comparison and difference equation of queueing systems give a new perspective and some new results are derived, which are difficult to obtain in the traditional queueing theory, such as the Max-Min optimality of service rate control [48] and the optimal admission control of queueing networks [46].

The structure of the relations of the above topics is illustrated in Fig. 7.1. The rest of this chapter is organized as follows. First, we will introduce the traditional PA theory in queueing systems. Then, we will briefly introduce the development on the performance optimization of Markov systems. Finally, we will discuss the further advancements on some emerging topics, including a new optimization framework called event-based optimization and some new ideas based on PA in financial engineering. The concept of perturbation realization factor (or performance potential for Markov systems) is the overall thread throughout the chapter.

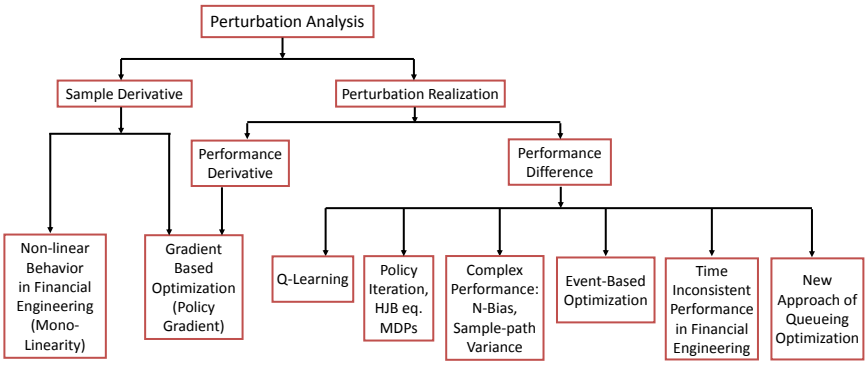


Fig. 7.1. Summary of the chapter: Some researches motivated by PA.

## 7.2. Perturbation Analysis of Queueing Systems Based on Perturbation Realization Factors

The PA theory in queueing systems reviewed in this chapter is based on perturbation realization factor, which measures the long-term effect of a single perturbation on the system average performance [4, 6]. With the perturbation realization factor as building blocks, the performance derivative equation and difference equation with respect to system parameters are derived, and the gradient-based algorithm and the policy iteration algorithm can be developed, respectively. In this section, we will mainly introduce these two types of sensitivity equations based on perturbation realization factors. Other parts of the PA theory, such as IPA, FPA, SPA, PA for fluid model, etc., are omitted for the limit of the space and can be referred to other parts of this book, or in the literature [6, 19, 24–26, 28, 31].

### 7.2.1. *Performance gradient*

Below, we use a typical queueing model, the closed Jackson network, to introduce the key results of the traditional PA theory in queueing systems. Please note, the similar principles of PA theory may be also applied to other queueing systems [6, 21, 28, 35]. Consider a closed Jackson network (also called Gordon-Newell network [27]) with $M$ servers and $N$ customers. The total number of customers in the network is a constant $N$. That is, there is no customer arrival to or departure from the network. The customer transits among the servers and receives service. The service time at every server obeys an exponential distribution. We denote the service rate of

server $i$ as $\mu_i$, $i = 1, 2, \ldots, M$. When a customer finishes its service at server $i$, it will transit to server $j$ with a routing probability $q_{ij}$, $i, j = 1, 2, \ldots, M$. Obviously, we have $\sum_{j=1}^{M} q_{ij} = 1$ for all $i$. The service discipline of servers is first come first serve (FCFS). The waiting capacity of servers is adequate and there are no customer overflows or losses. The number of customers (including the customer being served) at server $i$ is denoted as $n_i$. The system state is denoted as $\boldsymbol{n} := (n_1, n_2, \ldots, n_M)$. The state space is defined as $\mathcal{S} := \{\text{all } \boldsymbol{n} : \sum_{i=1}^{M} n_i = N\}$. 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 $t$, $i = 1, 2, \ldots, M$, $t \geq 0$. Define $T_l$ as the $l$th state transition time of the stochastic process $\boldsymbol{n}(t)$. $T_l$ also indicates the time when the entire network has served $l$ customers.

The cost function of the system is denoted as $f(\boldsymbol{n})$, $\boldsymbol{n} \in \mathcal{S}$. We define the *time-average performance* $\eta$ of the system as below.

$$\eta := \lim_{T\to\infty} \frac{1}{T} \int_0^T f(\boldsymbol{n}(t))dt = \lim_{T\to\infty} \frac{F_T}{T}, \tag{7.1}$$

where $F_T := \int_0^T f(\boldsymbol{n}(t))dt$ is defined as the accumulated performance until $T$. As a comparison, we define another performance metric called the *customer-average performance* $\eta_C$ as below.

$$\eta_C := \lim_{l\to\infty} \frac{1}{l} \int_0^{T_l} f(\boldsymbol{n}(t))dt = \lim_{l\to\infty} \frac{F_{T_l}}{l}. \tag{7.2}$$

When the network is strongly connected (any customer may visit every server in the network), $\boldsymbol{n}(t)$ is ergodic. Thus, the limits in (7.1) and (7.2) exist with probability 1. It is clear that $\eta$ is the system performance averaged for each unit of time, while $\eta_C$ is the system performance averaged for each customer. These two performance metrics are both important for queueing systems and they have the following relation

$$\eta_C = \lim_{l\to\infty} \frac{T_l}{l} \lim_{T_l\to\infty} \frac{1}{T_l} \int_0^{T_l} f(\boldsymbol{n}(t))dt = \frac{\eta}{\eta_{th}} = \eta\eta_I, \tag{7.3}$$

where $\eta_{th} := \lim_{l\to\infty} \frac{l}{T_l}$ is the average throughput of the entire network, $\eta_I$ is a special case of $\eta_C$ when $f(\boldsymbol{n}) = I(\boldsymbol{n}) \equiv 1$ for all $\boldsymbol{n}$. That is, we have

$$\eta_I := \lim_{l\to\infty} \frac{1}{l} \int_0^{T_l} 1dt = \lim_{l\to\infty} \frac{T_l}{l} = \frac{1}{\eta_{th}}. \tag{7.4}$$

Generally speaking, the optimizations for $\eta$ and $\eta_C$ are different and we have to develop specific approaches for them respectively [44].

Suppose that the service rate of server $i$ has a very small change $\Delta\mu_i$. That is, the service rate is changed from $\mu_i$ to $\mu_i + \Delta\mu_i$, where $\Delta\mu_i \to 0$. With the inverse transform method, we have $s = -\frac{1}{\mu_i}\ln\zeta$, where $\zeta$ is a uniformly distributed random number in $[0, 1]$ and $s$ is the random service time obeying exponential distribution with rate $\mu_i$. For the same randomness $\zeta$, when the service rate is changed from $\mu_i$ to $\mu_i + \Delta\mu_i$, the random service time $s$ will be changed to $s' = -\frac{1}{\mu_i+\Delta\mu_i}\ln\zeta = -\frac{1-\Delta\mu_i/\mu_i}{\mu_i}\ln\zeta + o(\Delta)$. The perturbation of the random service time can be written as

$$\Delta s = s' - s = -\frac{\Delta\mu_i}{\mu_i}s. \tag{7.5}$$

Therefore, during the simulation process, the service time of each customer at server $i$ will have a perturbation whose amount is proportional to its original service time with a scalar $-\frac{\Delta\mu_i}{\mu_i}$.

After the perturbations are generated, they are propagated throughout the entire network according to some specific rules [6]. The effect of such single perturbation on the accumulated performance can be quantified by a quantity called perturbation realization factor. Consider a single perturbation $\Delta$ of service time of server $i$ when the system is at state $\boldsymbol{n}$. The perturbation realization factor is denoted as $c^{(f)}(\boldsymbol{n}, i)$, $\boldsymbol{n} \in \mathcal{S}$, $i = 1, \ldots, M$, and it measures the long-term effect of this perturbation on the total accumulated performance $F_{T_l}$. Theoretically, $c^{(f)}(\boldsymbol{n}, i)$ is defined as below.

$$\begin{aligned} c^{(f)}(\boldsymbol{n}, i) &:= \lim_{l\to\infty}\lim_{\Delta\to 0} E\left\{\frac{\Delta F_{T_l}}{\Delta}\right\} = \lim_{l\to\infty}\lim_{\Delta\to 0} E\left\{\frac{F'_{T'_l} - F_{T_l}}{\Delta}\right\} \\ &= \lim_{l\to\infty}\lim_{\Delta\to 0} E\left\{\frac{1}{\Delta}\left[\int_0^{T'_l} f(\boldsymbol{n}'(t))dt - \int_0^{T_l} f(\boldsymbol{n}(t))dt\right]\right\}, \end{aligned} \tag{7.6}$$

where $\boldsymbol{n}'(t)$ is the stochastic process of the state of the perturbed system (with the perturbation $\Delta$ of service time of server $i$ at time 0) at time $t$, $T'_l$ is the $l$th service completion time of the perturbed system.

The value of $c^{(f)}(\boldsymbol{n}, i)$ can be numerically obtained by solving the following set of linear equations [6]

$$\text{If } n_i = 0, \text{ then } c^{(f)}(\boldsymbol{n}, i) = 0; \tag{7.7}$$

$$\sum_{i=1}^{M} c^{(f)}(\boldsymbol{n}, i) = f(\boldsymbol{n}); \tag{7.8}$$

$$\left\{\sum_{k=1}^{M} 1_{n_k>0}\mu_k\right\} c^{(f)}(\boldsymbol{n},i) = \sum_{k=1}^{M}\sum_{j=1}^{M} 1_{n_k>0}\mu_k q_{kj} c^{(f)}(\boldsymbol{n}_{-k,+j},i)$$
$$+\sum_{j=1}^{M}\mu_i q_{ij}\left\{\left[1-1_{n_j>0}\right] c^{(f)}(\boldsymbol{n}_{-i,+j},j) + f(\boldsymbol{n}) - f(\boldsymbol{n}_{-i,+j})\right\}; \quad (7.9)$$

where $\boldsymbol{n}_{-i,+j} := (n_1,\ldots,n_i-1,\ldots,n_j+1,\ldots,n_M)$ is a neighboring state of $\boldsymbol{n}$ with $n_i > 0$, $1_{n_i>0}$ is an indicator function defined as, if $n_i > 0$, $1_{n_i>0} = 1$; otherwise $1_{n_i>0} = 0$. On the other hand, $c^{(f)}(\boldsymbol{n},i)$ can also be estimated statistically from a single sample path. For more details, readers can refer to the literature [6, 45].

With $c^{(f)}(\boldsymbol{n},i)$ as building blocks, we can derive the performance derivative equation of PA theory in queueing systems in an intuitive way as follows. Suppose the service rate $\mu_i$ is decreased to $\mu_i - \Delta\mu_i$, where $\Delta\mu_i$ is an infinitesimal amount. According to the inverse transform method, every service time at server $i$ will have a delay $\frac{\Delta\mu_i}{\mu_i}s$, where $s$ is the random service time originally generated. During a period $T_l \gg 1$, the total amount of service time delay at state $\boldsymbol{n}$ is $T_l\pi(\boldsymbol{n})\frac{\Delta\mu_i}{\mu_i}$, where $\pi(\boldsymbol{n})$ is the steady-state probability that the system is at $\boldsymbol{n}$. Since the effect of a unit delay at $(\boldsymbol{n},i)$ is quantified by $c^{(f)}(\boldsymbol{n},i)$, the total effect of $-\Delta\mu_i$ on the accumulated performance $F_{T_l}$ is

$$\Delta F_{T_l} = \sum_{\boldsymbol{n}\in\mathcal{S}} \frac{\Delta\mu_i}{\mu_i} T_l\pi(\boldsymbol{n}) c^{(f)}(\boldsymbol{n},i). \quad (7.10)$$

Dividing $-\Delta\mu_i l$ on both sides and letting $l\to\infty$ and $\Delta\to 0$, we derive the following performance derivative equation

$$\frac{\mu_i}{\eta_I}\frac{d\eta_C}{d\mu_i} = -\sum_{\boldsymbol{n}\in\mathcal{S}} \pi(\boldsymbol{n}) c^{(f)}(\boldsymbol{n},i). \quad (7.11)$$

With (7.11), we can develop the gradient-based optimization algorithm to optimize the system parameters, such as the service rates, arrival rates, or routing probabilities. The performance gradients can be directly estimated from the sample path, or indirectly obtained by utilizing the estimates of $c^{(f)}(\boldsymbol{n},i)$'s and (7.11). In fact, the traditional PA approach in queueing systems can be viewed as an efficient implementation of the above derivative equation. Details can be found in the literature [6, 24, 30, 44, 45].

### 7.2.2. *Policy iteration*

The gradient-based optimization suffers its intrinsic deficiencies, such as the trapping into local optimum and the difficulty of selecting proper step-sizes. Fortunately, in the recent studies we find that the perturbation realization factors contain not only the derivative sensitivity information, but also the difference sensitivity information. For the optimization of service rates in a closed Jackson network, the following difference equation is derived by [44] when only one particular service rate $\mu_{i,\boldsymbol{n}}$ is changed to $\mu'_{i,\boldsymbol{n}}$.

$$\eta'_C - \eta_C = \eta'_I \pi'(\boldsymbol{n}) \left\{ \frac{-\Delta\mu_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}} c^{(f)}(\boldsymbol{n}, i) + h(\boldsymbol{n}) \right\}, \tag{7.12}$$

where $h(\boldsymbol{n}) := f'(\boldsymbol{n}) - f(\boldsymbol{n})$, $\eta'_I$, $\pi'(\boldsymbol{n})$, and $f'(\boldsymbol{n})$ are the corresponding parameters of the perturbed system with new service rate $\mu'_{i,\boldsymbol{n}}$. For the service rate control problem, we can further write $f'(\boldsymbol{n}) = f(\boldsymbol{n}, \vec{\mu}'_{\boldsymbol{n}})$, where $\vec{\mu}'_{\boldsymbol{n}} := (\mu_{1,\boldsymbol{n}}, \ldots, \mu'_{i,\boldsymbol{n}}, \ldots, \mu_{M,\boldsymbol{n}})$ in this scenario.

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 equation of $\eta_C$ is obtained similarly as below.

$$\eta'_C - \eta_C = \eta'_I \sum_{\boldsymbol{n}\in\mathcal{S}} \pi'(\boldsymbol{n}) \left\{ \frac{-\Delta\mu_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}} c^{(f)}(\boldsymbol{n}, i) + h(\boldsymbol{n}) \right\}. \tag{7.13}$$

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 equation is derived as follows by combining (7.8).

$$\eta'_C - \eta_C = \eta'_I \sum_{\boldsymbol{n}\in\mathcal{S}} \pi'(\boldsymbol{n}) \left\{ f(\boldsymbol{n}, \vec{\mu}'_{\boldsymbol{n}}) - \sum_{i=1}^{M} \frac{\mu'_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}} c^{(f)}(\boldsymbol{n}, i) \right\}, \tag{7.14}$$

where $\vec{\mu}'_{\boldsymbol{n}} := (\mu'_{1,\boldsymbol{n}}, \mu'_{2,\boldsymbol{n}}, \ldots, \mu'_{M,\boldsymbol{n}})$ in this scenario.

From (7.14), we can easily derive the optimality (HJB) equation for the service rate control problem: A set of service rates $\vec{\mu}^*_{\boldsymbol{n}} = (\mu^*_{1,\boldsymbol{n}}, \mu^*_{2,\boldsymbol{n}}, \ldots, \mu^*_{M,\boldsymbol{n}})$, $\boldsymbol{n} \in \mathcal{S}$, is optimal (for the minimization problem) if and only if it satisfies the optimality (HJB) equation

$$f(\boldsymbol{n}, \vec{\mu}_{\boldsymbol{n}}) - \sum_{i=1}^{M} \frac{\mu_{i,\boldsymbol{n}}}{\mu^*_{i,\boldsymbol{n}}} c^{(f)}(\boldsymbol{n}, i)^* \geq 0, \quad \forall\ \vec{\mu}_{\boldsymbol{n}}, \boldsymbol{n}, \tag{7.15}$$

where $c^{(f)}(\boldsymbol{n}, i)^*$ is the perturbation realization factors corresponding to the policy $\mathcal{L}^* := \{\vec{\mu}^*_{\boldsymbol{n}}, \boldsymbol{n} \in \mathcal{S}\}$ and the policy space is denoted as $\Psi := \{\text{all } \mathcal{L}\}$. The correctness of (7.15) is straightforward based on (7.14) since $\pi'(\boldsymbol{n})$ and $\eta'_I$ are always positive.

Based on the difference equation (7.14) and the optimality equation (7.15), we have the following policy iteration algorithm for minimizing the customer-average performance of queueing systems.

**Algorithm 1.** Policy iteration algorithm for the optimization of customer-average performance of state-dependent closed Jackson networks.

(1) *Initialization:* Choose an arbitrary policy $\mathcal{L}_0 \in \Psi$ as the initial policy, and set $k = 0$.
(2) *Evaluation:* At the $k$th 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 (the $(k+1)$th) iteration as $\mathcal{L}_{k+1} = \{\vec{\mu}'_{\boldsymbol{n}}, \boldsymbol{n} \in \mathcal{S}\}$ with

$$\vec{\mu}'_{\boldsymbol{n}} = \arg\min_{\vec{\mu}'_{\boldsymbol{n}}} \left\{ f(\boldsymbol{n}, \vec{\mu}'_{\boldsymbol{n}}) - \sum_{i=1}^{M} \frac{\mu'_{i,\boldsymbol{n}}}{\mu_{i,\boldsymbol{n}}} c^{(f)}(\boldsymbol{n}, i) \right\}, \ \boldsymbol{n} \in \mathcal{S}. \tag{7.16}$$

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

Compared with the gradient-based optimization, the policy iteration is much more efficient. During each iteration, the policy iteration always finds a strictly better policy (usually it is much better). While the gradient-based optimization searches within a near neighboring space. Thus, the policy iteration usually has a much faster convergence speed than that of the gradient-based optimization. Moreover, the policy iteration converges to the global optimum, while the gradient-based optimization is always trapped into a local optimum. This is a prominent improvement compared to the gradient-based optimization in the traditional PA theory.

As we see, (7.14) initiates a new approach of the performance optimization of queueing systems. Since (7.14) provides a very clear relation between the performance and the parameters, it may solve some problems which are difficult for the traditional approaches of queueing theory. For example, the service rate control problem is richly studied in the queueing theory, but it is difficult to solve in a complicated queueing network, such as a closed Jackson network. This is because the traditional approach usually uses the formulation of MDP. The associated optimality equation of this MDP model is too complicated to do effective analysis. However, based

on (7.14), we easily derive the Max-Min optimality for this service rate control problem. That is, when the cost function is convex with respect to the service rate, the optimal service rate is either maximal or minimal. This result is further more profound than those in the existing literature. This progress benefits from the difference equation (7.14). Details can be referred to in [48].

In summary, the perturbation realization factor plays a key role to build the performance derivative equation (7.11) and the performance difference equation (7.14). It contains the information of not only the performance derivative nearby the current policy, but also the performance difference under other policies. The perturbation realization factor can be efficiently estimated from a single sample path. Therefore, both the gradient-based optimization and the policy iteration can be implemented based on a single sample path, which makes the algorithms online implementable.

## 7.3. Performance Optimization of Markov Systems Based on Performance Potentials

During the recent decades, rich studies have been conducted to further extend the idea of PA theory from queueing systems to Markov systems [9, 17, 21].

In the previous section, we see that the parameter changing is decomposed into a series of perturbations and the total effect equals a weighted sum of perturbation realization factors. This idea is similarly applied to the Markov system, where we use the performance potential as the building blocks and the performance derivative and difference equation are derived thereafter. This gives a new perspective to study the performance optimization of Markov systems, besides the classical MDP theory. Below, we use a discrete time Markov chain to briefly introduce this theory (it has also been extended to continuous time Markov processes, multi-chain Markov systems, semi-Markov processes, etc. [8, 17, 49]). More detailed introduction and review can be referred to the literature [11, 12].

### 7.3.1. *Performance gradients and potentials*

Consider an ergodic discrete-time Markov chain $\boldsymbol{X} = \{X_0, X_1, X_2, \dots\}$, where $X_l$ is the system state at time $l$, $l = 0, 1, 2, \dots$. The system space is finite and denoted as $\mathcal{S} = \{1, 2, \dots, S\}$, where $S$ is the size of the state space.

The transition probability is denoted as $P = [p(j|i)]_{i,j=1}^{S}$. The steady-state distribution is denoted as a row vector $\pi = (\pi(1), \pi(2), \ldots, \pi(S))$. Obviously, we have $Pe = e$ and $\pi P = \pi$, where $e$ is an $S$-dimensional column vector whose elements are all 1. The cost function is denoted as a column vector $f = (f(1), f(2), \ldots, f(S))^T$. The long-run average performance of the Markov system is denoted as $\eta$ and we have

$$\begin{aligned}\eta = E\{f(X_l)\} &= \sum_{i=1}^{S} \pi(i) f(i) = \pi f \\ &= \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} f(X_l) = \lim_{L\to\infty} \frac{F_L}{L}, \end{aligned} \tag{7.17}$$

where $E$ denotes the expectation over steady-state distribution and $F_L := \sum_{l=0}^{L-1} f(X_l)$. At each state $i$, we have to choose an action $a$ from the action space $\mathcal{A}$. Different action $a$ determines different state transition probability $p^a(j|i)$, $i, j \in \mathcal{S}$ and $a \in \mathcal{A}$. A policy $d$ is a mapping from the state space $\mathcal{S}$ to the action space $\mathcal{A}$. That is, $d : \mathcal{S} \to \mathcal{A}$. The policy space is denoted as $\mathcal{D}$ and here we consider only the deterministic and stationary policies. Since the policy $d$ determines both the transition probability matrix $P$ and the cost function $f$, for simplicity, we also use $(P, f)$ to represent a policy in Markov systems in some situations. The optimization problem is to choose a proper $(P, f)$ to make the corresponding $\eta$ optimal.

Similar to the parameter perturbation discussed in the previous section, we consider that the transition probability matrix $P$ of Markov systems has perturbations. We assume that the states $X_l$ are observable, $l = 0, 1, \ldots$. For simplicity, we assume the cost function $f$ is irrelevant to the policy $d$. Different policy has different $P$. We also use $P$ to represent the policy of Markov systems. We want to obtain the performance gradients around the current policy in the policy space by analyzing the system's behavior under this policy $P$.

In the policy space, along the direction from $P$ to $P'$, we consider a randomized policy $P^\delta$: at state $i$ the system transits according to $p'(j|i)$, $i, j \in \mathcal{S}$, with probability $\delta$, and transits according to $p(j|i)$, $i, j \in \mathcal{S}$, with probability $1 - \delta$. That is, the transition probability matrix of the randomized policy is $P^\delta = (1 - \delta)P + \delta P'$, where $0 \le \delta \le 1$. Policy $P$ and $P'$ can be any two policies in the policy space. Let $\pi^\delta$ and $\eta^\delta$ be the steady-state probability and the long-run average performance associated with $P^\delta$. The performance derivative at policy $P$ along the direction $\Delta P := P' - P$

(from $P$ to $P'$) is

$$\frac{d\eta^\delta}{d\delta}\Big|_{\delta=0} = \lim_{\delta\to 0}\frac{\eta^\delta - \eta}{\delta}. \tag{7.18}$$

Different $P'$'s represent different directional derivatives in the policy space.

Consider the policy $P$ is slightly changed to $P^\delta$, $\delta \ll 1$. This slight change in $P$ will generate a series of perturbations on the sample path under $P$. Different from the perturbation (small delay of service time) in the queueing system, the perturbation in the Markov system is a "jump" from one state $i$ to another state $j$, $i, j \in \mathcal{S}$. That is, at time $l$, suppose the Markov chains with $P$ and $P^\delta$ are both at state $k$, $k \in \mathcal{S}$; at time $l+1$, the original Markov chain with $P$ may transit to state $X_{l+1} = i$; however, because of the slight change in the transition probability $p(\cdot|k)$, the Markov chain with $P^\delta$ may transit to state $X'_{l+1} = j$. Thus, there is a state "jump" from $i$ to $j$ if we compare these two sample paths. The long-term effect of such a state "jump" from $i$ to $j$ on the accumulated performance $F_L$ can be quantified by *perturbation realization factor* $\gamma(i, j)$, which is defined as

$$\gamma(i,j) := E\left\{\sum_{l=0}^{\infty}[f(X'_l) - f(X_l)]\Big|X'_0 = j, X_0 = i\right\}, \tag{7.19}$$

where $X'_l$ is the perturbed sample path with initial state $j$. $X_l$ and $X'_l$ can be viewed as two replications of a stochastic process with different initial states. Please note, $\gamma(i, j)$ in Markov systems is different from $c^f(\boldsymbol{n}, i)$ in queueing systems, although they are both called perturbation realization factors. Actually, these two kinds of perturbation realization factors have a relation as follows [42].

$$c^{(f)}(\boldsymbol{n}, i) - c(\boldsymbol{n}, i)\eta = 1_{n_i>0}\mu_{i,\boldsymbol{n}}\sum_{j=1}^{M} q_{ij}\gamma(\boldsymbol{n}_{-i,+j}, \boldsymbol{n}), \tag{7.20}$$

where $c(\boldsymbol{n}, i)$ is called perturbation realization probability and it is a special case of $c^{(f)}(\boldsymbol{n}, i)$ with $f(\boldsymbol{n}) \equiv 1$ for all $\boldsymbol{n}$.

The perturbation realization factor $\gamma(i, j)$ can be further written as below.

$$\gamma(i,j) = g(j) - g(i), \quad \forall i, j \in \mathcal{S}, \tag{7.21}$$

where $g(i)$, $i \in \mathcal{S}$, is called *performance potential*. Performance potential $g(i)$ quantifies the long-term contribution of initial state $i$ to the

accumulated performance and it is defined as

$$g(i) := E\left\{\sum_{l=0}^{\infty}[f(X_l)-\eta]\Big|X_0=i\right\}. \tag{7.22}$$

Performance potential is a fundamental quantity in Markov systems since it can compose the perturbation realization factor $\gamma(i,j)$. Decomposing the first step of the summation in (7.22), we can derive the following Poisson equation in a matrix form

$$(I-P)g+\eta e=f, \tag{7.23}$$

where $g=(g(1),g(2),\ldots,g(S))^T$ is the potential vector.

From the definition of $g(i)$ in (7.22), we can see that the effect of a jump from state $i$ to $j$ on the long-run accumulated performance $F_L$ can be measured by $\gamma(i,j)=g(j)-g(i)$. Finally, the effect of a small (infinitesimal) change in a Markov chain's transition probability matrix (from $P$ to $P^\delta$) on the long-run accumulated performance $F_L$ can be decomposed into the sum of the effects of all the single perturbations (jumps on a sample path) induced by the change in the transition probability matrix. With these principles, we can intuitively derive the equations for the performance derivative along any direction $\Delta P$ (from $P$ to any $P'$) as follows: During a long enough period $L\gg 1$, the period that the system stays at state $i$ is $\pi(i)L$; When the system is at state $i$, the probability that the sample path has a state jump from $i$ to $j$ caused by $P^\delta$ is $p^\delta(j|i)-p(j|i)$; The number of state jumps from $i$ to $j$ is $\pi(i)L[p^\delta(j|i)-p(j|i)]$; The effect of each state jump from $i$ to $j$ is measured by $\gamma(i,j)$; The total effect of all of these perturbations is summated as

$$\begin{aligned}F_L^\delta-F_L&=\sum_{i\in\mathcal{S}}\pi(i)L\sum_{j\in\mathcal{S}}[p^\delta(j|i)-p(j|i)]\gamma(i,j)\\&=\delta\sum_{i\in\mathcal{S}}\pi(i)L\sum_{j\in\mathcal{S}}[p'(j|i)-p(j|i)]g(j).\end{aligned} \tag{7.24}$$

Dividing $L$ on both sides of the above equation and combining (7.18), we have the following performance derivative equation

$$\frac{d\eta^\delta}{d\delta}\Big|_{\delta=0}=\pi(P'-P)g=\pi\Delta Pg. \tag{7.25}$$

This equation can be also easily derived from the Poisson equation, as we will see later. However, the PA principles provide a clear and intuitive explanation for the performance potential and the derivative equation, and it can be easily extended to other non-standard problems for which the

Poisson equation may not exist. For a more rigorous analysis to construct this derivative equation (7.25), see the literature [12].

### 7.3.2. *Policy iteration and HJB equation*

Besides the performance derivative equation (7.25), we also obtain the difference equation in Markov systems, which is similar to (7.14) in queueing systems. The derivation of difference equation can also be constructed by analyzing a series of perturbations on the sample paths [10], similar to the analysis process of (7.25). However, for the purpose of rigorous analysis and providing another perspective, we theoretically derive the difference equation based on the Poisson equation as follows.

Suppose the policy of a Markov system is changed from $(P, f)$ to $(P', f')$. The steady-state distribution of the perturbed system with $(P', f')$ is denoted as $\pi'$ and the corresponding system performance is $\eta' = \pi' f'$. Multiplying both sides of (7.23) with $\pi'$ and after some transformations, we derive the performance difference equation as follows.

$$\eta' - \eta = \pi'[(P' - P)g + (f' - f)] = \pi'(\Delta P g + h), \tag{7.26}$$

where $\Delta P = P' - P$ and $h = f' - f$. With (7.26), we can directly compare the performance difference of any two policies based only on the information of the current policy. This is the key idea of the direct-comparison based approach. The policy iteration directly follows as below. Based on the sample path of the current policy, we calculate or estimate the value of $g$ of the current system. We choose a proper $(P', f')$ which makes every element of the column vector $P'g+f'$ minimal component-wisely. Since $\pi'$ is always positive for ergodic systems, we directly have $\eta' \leq \eta$ and the system performance is improved for the minimization problem. Repeating this process, we can find the optimal policy within a finite number of iterations. This is the basic idea of policy iteration in Markov systems. Moreover, based on (7.26), we can directly derive the optimality (HJB) equation for Markov systems as follows.

$$Pg^* + f \succeq P^* g^* + f^*, \quad \forall P, f, \tag{7.27}$$

where $\succeq$ is $\geq$ component-wisely and $g^*$ is the performance potential under the optimal policy $(P^*, f^*)$.

The detailed algorithm of policy iteration can be referred to in [12, 39]. With (7.26), it is very easy to obtain the performance derivative equation. For the situation where the cost function is changed under different policies,

the performance derivative equation (7.25) is further extended as

$$\frac{d\eta^{\delta}}{d\delta} = \pi(\Delta P g + h). \tag{7.28}$$

With the performance derivative, the policy-gradient based approach can be developed for the continuous parameters optimization problem of Markov systems [12, 36].

In summary, the performance potential plays a key role in the direct comparison theory of Markov systems. The effect of a change in the policy applied to a Markov system can be intuitively decomposed into a series of state "jumps" and the effects of these state "jumps" can be quantified with the performance potentials. This analysis with perturbation decompositions is similar to that in the PA theory of queueing systems. With performance potentials as building blocks, we can construct the performance difference equation and the performance derivative equation. With the difference equation, the policy iteration algorithm is proposed. With the derivative equation, the policy-gradient based algorithm is proposed. The direct-comparison based approach provides a new perspective to study the performance optimization of Markov systems, besides the dynamic programming of the classic MDP theory. It is possible to develop other new approaches to handle the emerging problems which are difficult for the classical MDP theory, such as the event-based optimization. This is what we will discuss in the next section.

## 7.4. Beyond Dynamic Programming

In this section, we will first introduce some problems solved by the direct-comparison based approach in Markov systems, while these problems are difficult for the traditional dynamic programming method. These problems include the $N$-bias optimization problem and the sample-path variance minimization problem in MDP. Next we will introduce a new optimization framework called event-based optimization. Then we will introduce some new results in the financial engineering. The last two problems are solved by following the similar idea of the direct-comparison based approach in Markov systems. It is infeasible to use the dynamic programming to solve these two problems.

### 7.4.1. *New results based on direct comparison*

In this subsection, we introduce some new results based on the direct-comparison approach. These results are derived naturally and intuitively from the difference equation. They were not obtained by using the traditional dynamic programming, probably because applying dynamic programming to long-run average performance usually requires discounting with discounting factor approaching zero; this may loss intuitive insights.

#### 7.4.1.1. *N-bias optimality in MDP*

Consider a discrete-time multichain Markov chain $\boldsymbol{X} = \{X_0, X_1, \dots\}$, where $X_l$ is the system state at time $l$. The notations are the same as those in Section 7.3. The transition probability matrix under policy $d$ is denoted as $P^d$ and the associated cost function is denoted as $f^d$. The average performance under policy $d$ is denoted as a column vector $g_0^d$ with the component as

$$g_0^d(i) = \lim_{L\to\infty} \frac{1}{L} E\left\{ \sum_{l=0}^{L-1} f(X_l, d(X_l)) \Big| X_0 = i \right\}, \quad i \in \mathcal{S}. \tag{7.29}$$

From the above equation, we see that the average performance $g_0^d(i)$ is relevant to the initial state $i$ since $\boldsymbol{X}$ is a multichain.

We denote $(P^d)^*$ as the Cesaro limit of the transition probability matrix $P^d$, that is,

$$(P^d)^* := \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} (P^d)^l. \tag{7.30}$$

The *bias* under policy $d$ is denoted as a column vector $g_1^d$ with the component as

$$g_1^d(i) = \lim_{L\to\infty} E\left\{ \sum_{l=0}^{L} [f(X_l, d(X_l)) - g_0^d(i)] \Big| X_0 = i \right\}, \quad i \in \mathcal{S}. \tag{7.31}$$

From the above equation, we see that the bias $g_1^d(i)$ quantifies the contribution of initial state $i$ on the accumulated performance and $g_1^d(i)$ is equivalent to the performance potential $g(i)$ in (7.22) when the Markov chain is a unichain.

Furthermore, we define the $(n+1)$th bias under policy $d$ as a column vector with components

$$g_{n+1}^d(i) = -E\left\{\sum_{l=0}^{\infty}\{g_n^d(X_l) - [(P^d)^* g_n^d](i)\}\Big| X_0 = i\right\}, \quad i \in \mathcal{S}, \tag{7.32}$$

where $(P^d)^* g_n^d$ is the steady-state value of the $n$th bias, $i \in \mathcal{S}$, and $n \geq 1$.

With the above $n$th biases, we further define the *$n$th-bias optimality*, $n \geq 0$. The optimal $n$th bias is denoted as

$$g_n^*(i) := \min_{d \in \mathcal{D}_{n-1}} g_n^d(i), \quad i \in \mathcal{S}, \tag{7.33}$$

where $\mathcal{D}_{n-1} := \{d \in \mathcal{D}_{n-2} : g_{n-1}^d = g_{n-1}^*\}$ is the set of all $(n-1)$th-bias optimal policies and $\mathcal{D}_{-1} := \mathcal{D}$ is the original policy space. Therefore, if $d \in \mathcal{D}_{n-1}$, then $g_k^d = g_k^*$ for all $k = 0, 1, \ldots, n-1$. A policy $d^* \in \mathcal{D}_{n-1}$ is called $n$th-bias optimal if it satisfies

$$g_n^{d^*}(i) = g_n^*(i), \quad i \in \mathcal{S}. \tag{7.34}$$

Therefore, the $n$th-bias optimization problem is to find the $n$th-bias optimal policy, $n = 0, 1, \ldots$. When $n = 0$, it is equivalent to the classical MDP with the long-run average performance criterion. The classical MDP theory handles the average and bias optimality, but no work have been done for the $n$th-bias optimality. This is because the classical MDP theory usually has to utilize the $n$-discount optimality (with the discount approaching to 1) and it would be complicated to analyse. As we will show later, we can derive the performance difference equation for the $n$th-bias optimality, similar to the idea of the direct-comparison based approach in Markov systems. Based on the difference equations, it is very natural and direct to derive the policy iteration for the $n$th-bias optimality and the optimality equation. This benefits from the advantage of difference equation which directly compares the performance difference under any two policies. More details can be found in the literature [16, 49].

Similar to the difference equation (7.26), we can derive the difference equation of the $n$th biases under any two policies $d$ and $b$, $d, b \in \mathcal{D}_{n-1}$. When $n = 0$, the $n$th bias is the performance average and we derive the following difference equation

$$g_0^b - g_0^d = (P^b)^*[(f^b + P^b g_1^d) - (f^d + P^d g_1^d)] + [(P^b)^* - I] g_0^d. \tag{7.35}$$

When the Markov system is a unichain, the above equation is exactly the same as (7.26).

When $n = 1$ and $g_0^b = g_0^d$, we consider the bias optimality of MDP and the difference equation of the bias is as follows.

$$g_1^b - g_1^d = [I - P^b + (P^b)^*]^{-1}[(f^b + P^b g_1^d) - (f^d + P^d g_1^d)] + (P^b)^*(P^b - P^d)g_2^d. \tag{7.36}$$

When $n > 1$ and $g_{n-1}^b = g_{n-1}^d$, the difference equation of the $n$th-biases under policy $b$ and $d$ is as follows.

$$g_n^b - g_n^d = [I - P^b + (P^b)^*]^{-1}(P^b - P^d)g_n^d + (P^b)^*(P^b - P^d)g_{n+1}^d. \tag{7.37}$$

Therefore, we obtain the difference equation of the $n$th biases, $n = 0, 1, \ldots$. These equations above are more complicated than (7.26) because here we consider the multichain Markov system. The performance of multichain is a vector, while the performance of unichain is a scalar. It is not straightforward for the classical MDP theory to handle these $n$th-bias optimality problems [39, 41]. Based on the difference equations (7.35), (7.36), and (7.37), we can directly derive the policy iteration for the $n$th-bias optimality. The basic idea is similar to those in Section 7.3. Here we only give a very brief discussion.

We use $n = 1$ as an example to discuss the policy iteration to find the $n$th-bias optimal policy. Suppose the current policy $d$ is average optimal. We can choose a new policy $b$ from the set of the average optimal policy $D_0$, which satisfies the following conditions. First, we have to guarantee $f^b + P^b g_1^d \leq f^d + P^d g_1^d$ component-wisely. Second, we have to guarantee $(P^b g_2^d)(i) \leq (P^d g_2^d)(i)$ when $f^b(i) + (P^b g_1^d)(i) = f^d(i) + (P^d g_1^d)(i)$ for some $i \in \mathcal{S}$. With these conditions and the structure of $(P^b)^*$, we can directly have $g_1^b \leq g_1^d$ based on (7.36). Thus, the bias of the system is improved. Repeating this process, we can find the bias optimal policy finally. The scenario for a general $n$th-bias optimality is similar and the details can be referred to in [16, 49].

#### 7.4.1.2. *Optimization of sample-path variance in MDP*

In the classical MDP theory, we often discuss the optimization under the long-run average performance criterion. However, in many practical systems, such as the controller design in automatic control or the portfolio management in finance, the variability of the system performance is also an important metric. When the long-run average performance already attains optimum, how to choose a policy with a minimal variance is an interesting problem. More specially, the variance considered here is the limiting average variance along the sample path, we call it *sample-path variance*

which is different from the traditional definition of variance of a stochastic process [33].

The sample-path variance is difficult to handle because it is the time average of the square of the total costs deviating from the average cost. In an ergodic Markov chain, denote $\mathcal{D}_0$ as the set of optimal policies under the long-run average criterion. Under a policy $d \in \mathcal{D}_0$, the sample-path variance is defined as

$$\sigma_{sp}^d := \lim_{L\to\infty} \frac{1}{L} E\left\{ \left[ \sum_{l=0}^{L-1} (f^d(X_l) - \eta^*) \right]^2 \right\}, \tag{7.38}$$

where $\eta^*$ is the optimal long-run average performance. The sample-path variance minimization problem is to find the optimal policy $d^* \in \mathcal{D}_0$, which makes $\sigma_{sp}^{d^*}$ minimal, that is

$$d^* = \arg\min_{d\in\mathcal{D}_0} \sigma_{sp}^d. \tag{7.39}$$

Similar to the basic idea of the direct-comparison based approach in Markov systems, we can derive the following difference equation of the sample-path variances under any two policies $d$ and $b$, with $d, b \in \mathcal{D}_0$ [33]

$$\sigma_{sp}^b - \sigma_{sp}^d = \pi^b \left[ (P^b - P^d) g_{sp}^d + (f_{sp}^b - f_{sp}^d) \right], \tag{7.40}$$

where $f_{sp}^d$ and $g_{sp}^d$ are the performance function and the sample-path variance potential, respectively. The forms of $f_{sp}^d$ and $g_{sp}^d$ are special and defined as follows.

$$f_{sp}^d(i) := 2[f^d(i) - \eta^*] g^d(i) - [f^d(i) - \eta^*]^2, \quad \forall i \in \mathcal{S}, \tag{7.41}$$

where $g^d(i)$ is the performance potential under the long-run average performance criterion and its definition is (7.22).

The sample-path variance potential $g_{sp}^d$ is defined with the performance function $f_{sp}^d$ as below.

$$g_{sp}^d(i) := \lim_{L\to\infty} E\left\{ \sum_{l=0}^{L} [f_{sp}^d(X_l) - \pi^d f_{sp}^d] \Big| X_0 = i \right\}, \quad \forall i \in \mathcal{S}. \tag{7.42}$$

It is proved that $\pi^d f_{sp}^d = \sigma_{sp}^d$ [37].

With (7.40), it is easy to do the performance analysis for the sample-path variance minimization problem. Some conclusions, such as the direct comparison and the sufficient condition of the optimal policy, naturally follow. This gives valuable inspirations to handle such problems and more details can be referred to in [33].

### 7.4.2. *Event-based optimization*

In a standard MDP, the decision maker has to select the action at every state. We can call it state-based control. However, in many practical systems, the control decisions are made only when certain events happen. We call it event-based control. Since the sequence of events usually does not have a Markov property, how to optimize such kind of problems is a new challenge and it does not fit the optimization framework of MDP theory. In this subsection, we discuss a new optimization framework called *event-based optimization* to solve this kind of problems.

In the work of [11, 12, 15, 34, 47], the event-based optimization is studied with a sensitivity-based approach, which can be viewed as a consequence of Section 7.3. In this approach, the performance optimization is based on the difference equation and the derivative equation. The policy iteration and the gradient-based optimization follow directly along these two sensitivity equations. These approaches may be applied to cases where the dynamic programming fails to work. Below, we first introduce the mathematical formulation of the event-based optimization. Then we will briefly introduce the theory of event-based optimization based on direct comparison.

Consider a discrete time Markov chain $\boldsymbol{X} = \{X_0, X_1, \dots\}$, where $X_l$ is the system state at time $l$, $l = 0, 1, \dots$. The notations of the basic elements of $\boldsymbol{X}$ are the same as those in Section 7.3. An *event* $e$ is defined as a set of state transitions which possess certain common properties. That is, $e := \{\langle i, j\rangle : i, j \in \mathcal{S}$ and $\langle i, j\rangle$ has common properties$\}$, where $\langle i, j\rangle$ denotes the state transition from $i$ to $j$. All of the events which can trigger control compose the event space $\mathcal{E}$. Assume the event space is finite and the total number of types of events is $V$. We define $\mathcal{E} := \{e_\phi, e_1, e_2, \dots, e_V\}$, where $e_\phi$ indicates the "no action" event which means when it occurs, no action is taken. The input state of event $e$, $e \in \mathcal{E}$, is defined as $I(e) := \{$all $i \in \mathcal{S} :$ $\langle i, j\rangle \in e$ for some j$\}$. The output state of event $e$ at input state $i$ is defined as $O_i(e) := \{$all $j \in \mathcal{S} : \langle i, j\rangle \in e\}$. From the definition of event $e$, we see that the event includes more information than the system state. The event includes not only the current state, but also the next state. With an event, we may have some information about the future dynamics of the system. For example, if an event $e$ happens, we know that the current state must be in the set $I(e)$ and the next state must be in the set $O_i(e)$, $i \in I(e)$. The formulation of events can capture some structure information of the problem.

When an event happens, we have to take actions. The action space is assumed finite and it is denoted as $\mathcal{A}$. At event $e$, when an action $a$ is adopted, the state transition probability is denoted as $p^a(j|i,e)$, where $a \in \mathcal{A}$, $i,j \in \mathcal{S}$, and $e \in \mathcal{E}$. Here we consider the event-based policy $d$ and assume $d$ is Markovian and deterministic. That is, $d$ is a mapping from the event space $\mathcal{E}$ to the action space $\mathcal{A}$ and $d : \mathcal{E} \to \mathcal{A}$. At time $l$, we have to determine the policy $d_l$, $l = 0, 1, \ldots$. We further assume the policy is stationary, that is $d_l \equiv d$ for all $l$. Therefore, we only consider the stationary and deterministic policy and the all of the possible policies compose the event-based policy space denoted as $\mathcal{D}_e$. The cost function under policy $d$ is denoted as $f^d = [f^d(i)]_{i\in\mathcal{S}}$ and the associated long-run average performance is denoted as $\eta^d$. The stead-state distribution under policy $d$ is denoted as $\pi^d$ and we have $\eta^d = \pi^d f^d$. The goal of event-based optimization is to find the optimal event-based policy $d$ which makes the long-run average performance $\eta^d$ minimal, where $d \in \mathcal{D}_e$. That is,

$$d^* = \arg\min_{d\in\mathcal{D}_e} \eta^d = \arg\min_{d\in\mathcal{D}_e} \lim_{L\to\infty} E\left\{\frac{1}{L}\sum_{l=0}^{L-1} f^d(X_l)\right\}. \tag{7.43}$$

The framework of event-based optimization captures the structure of event-triggered control in many practical systems. Moreover, in the practice, the event occurrence is much rarer that the state transitions and the event space size is much less than the state space size, i.e., $|\mathcal{E}| \ll |\mathcal{S}|$. Therefore, the complexity of the event-based optimization is much less than that of the standard MDP. Below, we introduce how to follow the direct comparison based approach in Section 7.3 to solve this event-based optimization problem.

First, we study the performance difference equation of the event-based optimization. Under policy $d$, the state transition probability of the Markov system is denoted as $p^d(j|i,e)$, where $i,j \in \mathcal{S}$, $e \in \mathcal{E}$, $d \in \mathcal{D}_e$, and $\langle i,j\rangle \in e$. With the law of total probability and the formula of conditional probability, we have

$$p^d(j|i) = \sum_{e=1}^{V} \frac{\pi(i,j,e)}{\pi^d(i)} = \sum_{e=1}^{V} \frac{p^d(j|i,e)\pi^d(i,e)}{\pi^d(i)} = \sum_{e=1}^{V} p^d(j|i,e)\pi^d(e|i), \tag{7.44}$$

where $\pi^d(e|i)$ is the conditional probability that event $e$ happens at the current state $i$. Usually, this probability $\pi^d(e|i)$ is determined by the

definition of the event and it is irrelevant to the policy $d$ [12]. Therefore, we rewrite (7.44) as below.

$$p^d(j|i) = \sum_{e=1}^{V} p^d(j|i,e)\pi(e|i). \tag{7.45}$$

With (7.45), we apply the difference equation (7.26) to this event-based optimization. Consider two event-based policies $d$ and $b$, $d, b \in \mathcal{D}_e$. The performance difference under these two policies is quantified by (7.26) as follows.

$$\begin{aligned}
\eta^b - \eta^d &= \pi^b[(P^b - P^d)g^d + (f^b - f^d)] \\
&= \sum_{i=1}^{S} \pi^b(i) \Bigg\{ \sum_{j=1}^{S} \sum_{e=1}^{V} \pi(e|i)[p^{b(e)}(j|i,e) - p^{d(e)}(j|i,e)]g^d(j) \\
&\quad + [f^b(i) - f^d(i)] \Bigg\} \\
&= \sum_{i=1}^{S} \Bigg\{ \sum_{j=1}^{S} \sum_{e=1}^{V} \pi^b(i,e)[p^{b(e)}(j|i,e) - p^{d(e)}(j|i,e)]g^d(j) \\
&\quad + \pi^b(i)[f^b(i) - f^d(i)] \Bigg\} \\
&= \sum_{e=1}^{V} \pi^b(e) \sum_{i \in I(e)} \pi^b(i|e) \Bigg\{ \sum_{j \in O_i(e)} [p^{b(e)}(j|i,e) - p^{d(e)}(j|i,e)]g^d(j) \\
&\quad + [f^b(i) - f^d(i)] \Bigg\},
\end{aligned} \tag{7.46}$$

The difference equation (7.46) clearly describes how the system performance is changed under any two event-based policies. We notice that (7.46) has relation to $\pi^b(i|e)$ which is a probability of the system under the new policy $b$. In order to choose a better policy $b$, it is computationally exhaustive to enumerate $\pi^b(i|e)$ for all of the possible $b$. Fortunately, under some conditions, we may overcome this difficulty. If the event-based optimization problem satisfies the following property

$$\pi^b(i|e) = \pi^d(i|e), \quad \forall\ i \in I(e), e \in \mathcal{E}, \text{ for any two policies } b \text{ and } d, \tag{7.47}$$

by substituting (7.47) into (7.46), we have the following difference equation for the event-based optimization

$$\eta^b - \eta^d = \sum_{e=1}^{V} \pi^b(e) \sum_{i \in I(e)} \pi^d(i|e) \Bigg\{ \sum_{j \in O_i(e)} [p^{b(e)}(j|i,e) - p^{d(e)}(j|i,e)] g^d(j) + [f^b(i) - f^d(i)] \Bigg\}. \tag{7.48}$$

As we know, $\pi^b(e)$ is always positive for ergodic systems although it is difficult to enumerate its value under all of the possible $b$. $\pi^d(i|e)$ and $g^d(j)$ can be calculated or estimated based on the sample path under the current policy $d$. With (7.48), if we choose

$$b(e) := \arg\min_{a \in \mathcal{A}} \sum_{i \in I(e)} \pi^d(i|e) \Big\{ \sum_{j \in O_i(e)} p^a(j|i,e) + f(i,a) \Big\}, \quad \forall e \in \mathcal{E}, \tag{7.49}$$

then we have $\eta^b \leq \eta^d$. With this update procedure, the performance of the policy will be repeatedly improved until we find the optimal policy $d^*$. This optimization procedure is similar to the policy iteration in the direct-comparison based approach of Markov systems. The optimality equation of the event-based optimization can be also derived similar to (7.27). There are more studies about the implementation of the above optimization procedure in the event-based optimization, such as the online estimation of $\pi^d(i|e)$, $g^d(i)$, and the Q-factor based optimization algorithms. For more details, please refer to the literature [11, 12, 34, 47].

However, the optimality of the above procedure requires the condition (7.47), which is satisfied in categories of problems [11, 40, 43, 46, 48]. That is, the steady-state conditional distribution of state $i$ given event $e$ should be irrelevant to the event-based policy $d$. When (7.47) does not hold, we cannot guarantee that the above procedure can find the optimal policy. For such situation, we can develop the gradient-based optimization. With (7.48), it is easy to derive the following derivative equation

$$\frac{\partial \eta^d}{\partial \delta} = \sum_{e=1}^{V} \pi^d(e) \sum_{i \in I(e)} \pi^d(i|e) \Bigg\{ \sum_{j \in O_i(e)} [p^{b(e)}(j|i,e) - p^{d(e)}(j|i,e)] g^d(j) + [f^b(i) - f^d(i)] \Bigg\}. \tag{7.50}$$

The above derivative equation does not require the condition (7.47). With (7.50), we can estimate the performance derivative along a direction in the

policy space (direction from policy $d$ to policy $b$, $b, d \in \mathcal{D}_e$) and a local optimum can be found by the followed gradient-based optimization.

In summary, the theory of event-based optimization provides an efficient way to optimize the system performance where the control decision is triggered by certain events. The event-based optimization inherits the main idea of the direct-comparison based optimization of Markov systems. By deriving the difference equation of event-based optimization, we can develop the policy iteration to find the optimal event-based policy. Similarly, we also derive the derivative equation and develop the gradient-based optimization. The performance potential plays a fundamental role in the above theory. The theory of event-based optimization may handle the problems which do not fit the traditional optimization framework, such as the dynamic programming. Therefore, this approach provides a new perspective of the optimization theory for stochastic systems beyond the dynamic programming.

### 7.4.3. *Financial engineering related*

The idea of the PA theory can be applied to many fields. In the financial engineering, some new results are derived recently.

Portfolio management is an important research topic in the financial engineering. A portfolio consists of a number of assets, for example, the bond assets and the stock assets, etc. The prices of the assets vary in the market environment according to some patterns of stochastic processes. The goal of portfolio management is to find an optimal policy determining the amount of all kinds of assets (buy or sell the assets) to make the total wealths maximal.

One of the formulations of the portfolio management problem is to use a partial observable Markov decision process (POMDP) [40]. Consider a discrete time model of this problem. At the beginning, the amount of the bond asset and the stock asset in the portfolio are $x_0^b$ and $x_0^s$, respectively. At time $l$, the price of the bond and the stock are denoted as $p_l^b$ and $p_l^s$, respectively, where $l = 0, 1, \ldots$. We have a dynamic equation to quantify the variation of the prices. For the price of the bond asset, we have

$$p_{l+1}^b = f^b(p_l^b, r), \tag{7.51}$$

where $f^b$ is a deterministic function and $r$ is the risk free interest rate of the market. For the price of the stock asset, we have

$$p_{l+1}^s = f^s(p_l^s, \mu, V_l, B_l), \tag{7.52}$$

where $f^s$ is also a deterministic function, $\mu$ is the appreciation rate, $V_l$ is called the volatility at time $l$, and $B_l$ is the value of a random walk at time $l$. We formulate that $V_l$ also follows a random walk. At each time $l$, we can observe $x_l^b, x_l^s, p_l^b, p_l^s$ and we cannot observe $V_l$. We define the system state as $(p_l^b, p_l^s, V_l)$ and we see that the system state is partial observable. The action is to determine the values of $x_l^b$ and $x_l^s$ at every time $l$. The goal is to choose an optimal policy which maximizes the long-run average reward rate, i.e.,

$$\lim_{l\to\infty} E\left\{\frac{\log(x_l^b p_l^b + x_l^s p_l^s)}{l}\right\}.$$

For this problem, we can use the approach of the direct comparison to conduct a complete analysis. The idea of the event-based optimization is applied to this problem [40]. By properly defining the event of this problem, we formulate this problem with the event-based optimization framework. Fortunately, the condition (7.47) for the optimality of event-based optimization is satisfied in this problem. The correctness of (7.47) in this problem is easy to understand based on the following fact, the change of an individual's portfolio management policy does not affect the entire market. That is, in a complete market, the buy or sale behavior of an individual investor has no notable effect on the price of the assets. Thus, the price distribution of assets is irrelevant to the policy of an individual and the condition (7.47) holds for this problem. Therefore, we can use similar approach to find the optimal policy of this portfolio management problem. Details can be referred to [40]. Other study of stock portfolio optimization as a Markov decision process can be referred to [20].

Another recent result about the portfolio management is the optimization with a distorted performance criterion, which is also based on the PA theory [14]. In the portfolio management, we use an utility function to measure the investor's "satisfaction" to the return of the portfolio. Caused by the human's psychological characteristics, the behavior of the investor's "satisfaction" to different returns of the portfolio is nonlinear. Such nonlinearity can be explained as the people's preference to different risks. Such nonlinear behavior makes the utility function complicated and we call it distorted performance function, since it can be modeled by distorting the probability of events. With the distorted performance criterion, the standard optimization approach, such as the dynamic programming, fails to work for this problem. This is because, if a policy is optimal for a distorted performance starting from time $t$, this policy may be no longer optimal for

the system starting from $t'$, where $t' < t$. This is caused by the nonlinearity of the distorted performance function. Thus, the recursion for the optimal policy in dynamic programming is not valid.

Fortunately, [14] find the mono-linearity of the distorted performance function. That is, the distorted performance function becomes locally linear if we properly change the underlying probability measure. Thus, the derivative of the distorted performance can be estimated by the expectation of the sample-path based derivatives of the distorted performance. Then, we can apply the PA theory to efficiently estimate the performance derivative from the sample path. A gradient-based optimization approach is developed to solve this portfolio management problem. For more details, readers can refer to [14].

In summary, the PA theory provides a new and efficient perspective to study the portfolio management problem in the financial engineering, especially when the standard optimization approaches fail to work. This also demonstrates the broad applicability of the PA theory to various research fields.

## Acknowledgments

The chapter briefly records the research development starting from PA that the authors have been involved. Prof. Ho's deep insight has been motivating all my research activities and influencing my students (by Cao) We wish Mrs. and Prof. Ho, a long, happy, and healthy life.

## References

[1] Bryson, A. E. and Ho, Y. C. (1969). *Applied Optimal Control: Optimization, Estimation, and Control*, Blaisdell, Waltham, Massachusetts.

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

[3] Cao, X. R. (1987a). First-order perturbation analysis of a single multi-class finite source queue. *Performance Evaluation* 7, 31–41.

[4] Cao, X. R. (1987b). Realization probability in closed Jackson queueing networks and its application. *Advances in Applied Probability* 19, 708–738.

[5] Cao, X. R. (1988). A sample performance function of Jackson queueing networks. *Operations Research* 36, 128–136.

[6] Cao, X. R. (1994). *Realization Probabilities — The Dynamics of Queueing Systems*. New York: Springer Verlag.

[7] Cao, X. R. (2000). A unified approach to Markov decision problems and performance sensitivity analysis. *Automatica* 36, 771–774.

[8] Cao, X. R. (2003a). From perturbation analysis to Markov decision processes and reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications* 13, 9–39.

[9] Cao, X. R. (2003b). Semi-Markov decision problems and performance sensitivity analysis. *IEEE Transactions on Automatic Control* 48, 758–769.

[10] Cao, X. R. (2004). The potential structure of sample paths and performance sensitivities of Markov systems. *IEEE Transactions on Automatic Control* 49, 2129–2142.

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

[12] Cao, X. R. (2007). *Stochastic Learning and Optimization — A Sensitivity-based Approach*. New York: Springer.

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

[14] Cao, X. R. and Wan, X. (2012). Analysis of non-linear behavior — a sensitivity-based approach. *Proceedings of the 2012 IEEE Conference of Decision and Control*, Maui, Hawaii, US.

[15] Cao, X. R. and Zhang, J. (2008a). Event-based optimization of Markov systems. *IEEE Transactions on Automatic Control* 53, 1076–1082.

[16] Cao, X. R. and Zhang, J. (2008b). The nth-order bias optimality for multichain Markov decision processes. *IEEE Transactions on Automatic Control* 53, 496–508.

[17] Cao, X. R. and Chen, H. F. (1997). Potentials, perturbation realization, and sensitivity analysis of Markov processes. *IEEE Transactions on Automatic Control* 42, 1382–1393.

[18] Cassandras, C. G. and Lafortun, S. (2008). *Introduction to Discrete Event Systems, 2nd Edition*, Springer-Verlag, New York.

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

[20] Ding, C. and Xi, Y. (2012). Study on stock portfolio optimization problem based on Markov chain. *Computer Simulation* 29, 366–369.

[21] Fu, M. C. and Hu, J. Q. (1994). Smoothed perturbation analysis derivative estimation for Markov chains. *Operations Research Letters* 15, 241–251.

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

[23] Glasserman, P. (1990). The limiting value of derivative estimators based on perturbation analysis. *Communications in Statistics: Stochastic Models* 6, 229–257.

[24] Glasserman, P. (1991). *Gradient Estimation via Perturbation Analysis*. Boston, MA: Kluwer Academic Publishers.

[25] Glasserman, P. and Gong, W. B. (1990). Smoothed perturbation analysis for a class of discrete event system. *IEEE Transactions on Automatic Control* 35, 1218–1230.

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

[27] Gordon, W. J. and Newell, G. F. (1967). Closed queueing systems with exponential servers. *Operations Research* 15, 252–265.

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

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

[30] Ho, Y. C. and Cao, X. R. (1991). *Perturbation Analysis of Discrete Event Systems*. Norwell: Kluwer Academic Publishers.

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

[32] Ho, Y. C., Eyler, A., and Chien, T. T. (1979). A gradient technique for general buffer-storage design in a serial production line. *International Journal of Production Research* 17, 557–580.

[33] Huang, Y. and Chen, X. (2012). A sensitivity-based construction approach to sample-path variance minimization of Markov decision process. *Proceedings of 2012 Australian Control Conference*, Sydney, Australia, 215–220.

[34] Jia, Q. S. (2011). On solving event-based optimization with average reward over infinite stages. *IEEE Transactions on Automatic Control* 56, 2912–2917.

[35] Leahu, H., Heidergott, B., and Hordijk, A. (2012). Perturbation analysis of waiting times in the G/G/1 queue. *Discrete Event Dynamic Systems: Theory and Applications* 22, DOI 10.1007/s10626-012-0144-0.

[36] Marbach, P. and Tsitsiklis, J. N. (2001). Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control* 46, 191–209.

[37] Meyn, S. P., Tweedie, R. L., and Glynn, P. W. (2009). *Markov Chains adn Stochastic Stability*. Cambridge: Cambridge University Press.

[38] Parr, R. E. (1998). *Hierarchical Control and Learning for Markov Decision Processes*. Ph.D. Dissertation, University of California at Berkeley.

[39] Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: John Wiley & Sons.

[40] Wang, D. X. and Cao, X. R. (2011). Event-based optimization for POMDP and its application in portfolio management. *Proceedings of the 18th IFAC World Congress*, Milano, Italy, 3228–3233.

[41] Veinott, A. F. (1969). Discrete dynamic programming with sensitive discount optimality criteria. *The Annals of Mathematical Statistics* 40, 1635–1660.

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

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

[44] Xia, L., Chen, X., and Cao, X. R. (2009). Policy iteration for customer-average performance optimization of closed queueing systems. *Automatica* 45, 1639–1648.

[45] Xia, L. and Cao, X. R. (2012). Performance optimization of queueing systems with perturbation realization. *European Journal of Operational Research* 218, 293–304.

[46] Xia, L. (2013a). Event-based optimization of admission control in open queueing networks. *Discrete Event Dynamic Systems: Theory and Applications*, under review.

[47] Xia, L., Jia, Q. S., and Cao, X. R. (2013b). Event-based optimization, — A new optimization framework. *Discrete Event Dynamic Systems: Theory and Applications*, under review.

[48] Xia, L. and Shihada, B. (2013c). Max-Min optimality of service rate control in closed queueing networks. *IEEE Transactions on Automatic Control*, to appear.

[49] Zhang, J. and Cao, X. R. (2009). Continuous-time Markov decision processes with $n$th-bias optimality criteria. *Automatica* 45, 1628–1638.