Mathematics Theory of Information Science

(A Course Report – Spring 2017)

Rab Nawaz
PhD Scholar (Reg#: BL16006002)
School of Information Science and Technology
University of Science and Technology of China, Hefei
Email: rabnawaz@mail.ustc.edu.cn

Submitted to

Prof. Yin Baoqun
Email: bqyin@ustc.edu.cn

Table of Contents

Referenced Book

Stochastic Learning and Optimization by Xi-Ren Cao

[A Report on First Three Chapters]

General Overview

Information is playing an increasing role in our industrialized society. A technical overview of the flourishing electronics industry stated in 1987: "On almost every technology front, the driving force behind new developments is the ever-rising demand for information. Huge amounts of data and information, larger than anyone ever dreamed of a generation ago, have to move faster and faster through processors and networks, then end up having to be stored.

The goal of learning and optimization is to make the "***best***" decisions to optimize, or to improve, the performance of a system based on the information obtained by observing and analyzing the system's behavior. A system's behavior is usually represented by a model, or by the sample paths (also called trajectories) of the system. A sample path is a record of the operation history of a system.

Performance optimization plays an important role in the design and operation of modern engineering systems in many areas, including communications (Internet and wireless networks), manufacturing, logistics, robotics, and bioinformatics. Most engineering systems are too complicated to be analyzed, or the parameters of the system models cannot be easily obtained. Therefore, learning techniques have to be applied.

To develop efficient algorithms for performance optimization, we need to explore the special features of a system. This process is called learning. For dynamic systems, learning may involve observing and analyzing a sample path of a system to obtain necessary information; this is in the normal sense of the word "learning", as it is used in research areas such as reinforcement learning. Simulation-based and on-line optimization approaches are based on learning from sample paths.

If we know something about the structure and the dynamics of the system, we may develop efficient optimization techniques (analytic, simulation, on-line, learning, etc.).

Stochastic processes have applications in many disciplines including sciences such as biology, chemistry, ecology, neuroscience, and physics as well as technology and engineering fields such as image processing, signal processing, information theory, computer science, cryptography and telecommunications. Furthermore, seemingly random changes in financial markets have motivated the extensive use of stochastic processes in finance.

Chapter 1

1.1 Stochastic Learning

The term stochastic occurs in a wide variety of professional or academic fields to describe events or systems that are unpredictable due to the influence of a random variable. The word "stochastic" comes from the Greek word στόχος (stokhos, "aim"). Researchers refer to physical systems in which they are uncertain about the values of parameters, measurements, expected input and disturbances as "stochastic systems". In probability theory, a purely stochastic system is one whose state is randomly determined, having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely. In this regard, it can be classified as non-deterministic (i.e., "random") so that the subsequent state of the system is determined probabilistically. Any system or process that must be analyzed using probability theory is stochastic at least in part. Stochastic systems and processes play a fundamental role in mathematical models of phenomena in many fields of science, engineering, finance and economics.

1.2 An Overview of Learning and Optimization

Performance optimization plays an important role in the design and operation of modern engineering systems in many areas, including communications (Internet and wireless networks), manufacturing, logistics, robotics, and bioinformatics. Most engineering systems are too complicated to be analyzed, or the parameters of the system models cannot be easily obtained.

1.3 Problem Description

The stochastic dynamic systems are dynamic system evolves as time passes. It is generally easier to explain the ideas with a discrete time model, in which time takes discrete values denoted as $l = 0,1,2,..$ Here, the word “state” is used in a strict sense that a sample path X is a Markov chain. This means that given the current state X_l, the system’s future behavior $\{X_{l+1},X_{l+2},...\}$ is independent of its past history $\{X_0,X_1, \ldots, X_{l-1}\}$.This is called the Markov property. Intuitively, a state completely captures the system’s current status in regard to its future evolution.

In optimization problems, at any time l, we can apply an action. The actions A_0, A_1... may affect the evolution of the system. With the Markov model, the actions control the transition probabilities of the state process.

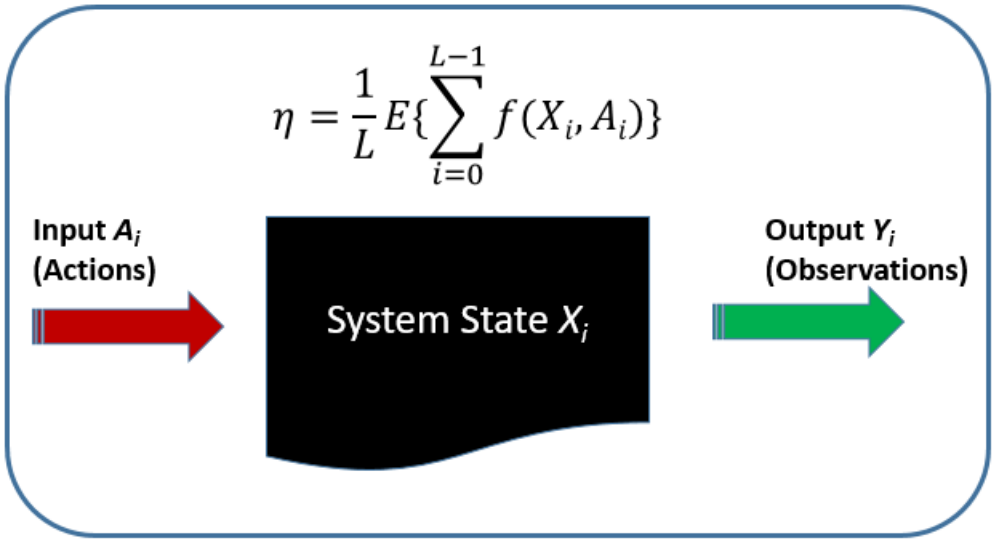


Figure 1. A Model of Learning and Optimization

A generic description of the learning and optimization is presented in figure 1 above. The shaded area represents a stochastic dynamic system. The system is essentially a black box and it can only interact with the outside through its inputs and outputs. The inputs provide a vehicle to intervene or to control the operation of the system, and/or to affect the reward of the operation. The outputs provide a window for observing the system. That is, the outputs are the observations. Associated with every system, there is a performance measure η. In the figure, as an example, η is taken to be the mean of the average reward.

The goal of an optimization problem is to answer the following question:

- ✓ *Based on the information we know about the system, i.e., the output history learned from observation.*
- ✓ *The input (action) history.*
- ✓ *What action should we take at a particular time so that we can obtain the best possible system performance?*

In engineering applications, at the design stage, sample paths can only be obtained by simulation following a system model; and while a system is operating, the paths can also be obtained by direct observation. If learning and optimization is implemented by simulation, then the approach is called a *simulation-based* approach.

For real systems, performance optimization (or improvement) decisions can be made through learning the system behavior by observing its sample paths recorded while the system is operating without interruption; we call such an approach an *on-line* app.

1.4 Optimal Policies

The main element in learning and optimization is the policy.

1.4.1 Open-Loop Policies

Figure 2. Shows the structure of an open loop policy system. With an open-loop policy, the output-and-action history are not used in determining the actions; i.e., the function $\boldsymbol{A_l = d_l(H_l)}$ does not depend on $\boldsymbol{H_l}$, l = 0,1,... , at all. The outputs may be random; the best policy corresponds to the best average performance.

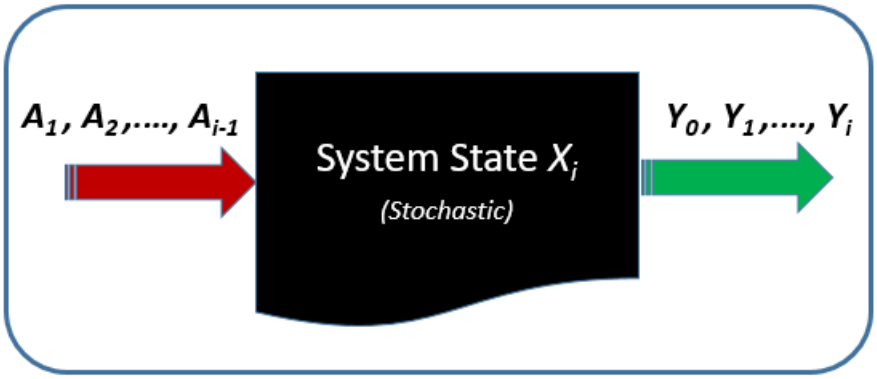


Figure 2. The Open Loop Policy System

1.4.2 Closed-Loop (Feedback) Policies

If the system is involved with randomness and there are observable outputs, a fixed sequence of actions may not lead to the best performance. In such cases, in addition to the past history of the actions, the action chosen at time l, Al, should also depend on the observations up to time l. This corresponds to the "closed-loop" or "feedback" control in control theory.

When a system involves randomness, how the observation $\boldsymbol{Y_l}$ ***e***volves after an action is taken is the system's intrinsic character, which is not controlled by us. In other words, if we choose action $\boldsymbol{A_l}$, then the nature will determine $\mathbf{Y}_{l+1}$, and so on. The best thing we can do is to use all the information that is

available to us up to time l to determine the action $A_l = d_l(H_l)$ such that on average the system performance is the best. Such a policy depending on the observation history is called a feedback policy or closed-loop policy in control theory. Unlike the open-loop policies, actions in a feedback policy cannot be pre-determined before the system operates.

Table 1. Gives an example of possible histories of actions and observations and their corresponding performance. To save space, in the table, we assume that for some reasons the observations of the system at time $l = 0$ and l=1are fixed as y_0 and y_1, respectively. The observation at l=2, however, may take either 0 or y1, depending on the randomness involved. We further assume that the probabilities of $Y_2 = y_0$ and $Y_2 = y_1$ are both 0.5, equally. As shown in the table, there are two possible histories corresponding to each action sequence; e.g., if we take action sequence $\{\alpha_0,\alpha_1,\alpha_0\}$,we may in fact have either $h_4:=\{y_0,\alpha_0,y_1,\alpha_1,y_0,\alpha_0\}$ or $h_5:=\{y_0,\alpha_0,y_1,\alpha_1,y_1,\alpha_0\}$ with an equal probability of 0.5, respectively. We find that the average performance for every action sequence is the same as the performance corresponding to the same action sequence in Table 1.

Table 1. The Action-Observation Histories and Their Rewards

	Action Sequences	Action-Observation Histories	Performance
h_0	$\{\alpha_0, \alpha_0, \alpha_0\}$	$\{y_0, \alpha_0, y_1, \alpha_0, y_0, \alpha_0\}$	2
h_1	$\{\alpha_0, \alpha_0, \alpha_0\}$	$\{y_0, \alpha_0, y_1, \alpha_0, y_1, \alpha_0\}$	4
h_2	$\{\alpha_0, \alpha_0, \alpha_1\}$	$\{y_0, \alpha_0, y_1, \alpha_0, y_0, \alpha_1\}$	8
h_3	$\{\alpha_0, \alpha_0, \alpha_1\}$	$\{y_0, \alpha_0, y_1, \alpha_0, y_1, \alpha_1\}$	6
h_4	$\{\alpha_0, \alpha_1, \alpha_0\}$	$\{y_0, \alpha_0, y_1, \alpha_1, y_0, \alpha_0\}$	12
h_5	$\{\alpha_0, \alpha_1, \alpha_0\}$	$\{y_0, \alpha_0, y_1, \alpha_1, y_1, \alpha_0\}$	4
h_6	$\{\alpha_0, \alpha_1, \alpha_1\}$	$\{y_0, \alpha_0, y_1, \alpha_1, y_0, \alpha_1\}$	6
h_7	$\{\alpha_0, \alpha_1, \alpha_1\}$	$\{y_0, \alpha_0, y_1, \alpha_1, y_1, \alpha_1\}$	2
h_8	$\{\alpha_1, \alpha_0, \alpha_0\}$	$\{y_0, \alpha_1, y_1, \alpha_0, y_0, \alpha_0\}$	0
h_9	$\{\alpha_1, \alpha_0, \alpha_0\}$	$\{y_0, \alpha_1, y_1, \alpha_0, y_1, \alpha_0\}$	10
h_{10}	$\{\alpha_1, \alpha_0, \alpha_1\}$	$\{y_0, \alpha_1, y_1, \alpha_0, y_0, \alpha_1\}$	12
h_{11}	$\{\alpha_1, \alpha_0, \alpha_1\}$	$\{y_0, \alpha_1, y_1, \alpha_0, y_1, \alpha_1\}$	8
h_{12}	$\{\alpha_1, \alpha_1, \alpha_0\}$	$\{y_0, \alpha_1, y_1, \alpha_1, y_0, \alpha_0\}$	10
h_{13}	$\{\alpha_1, \alpha_1, \alpha_0\}$	$\{y_0, \alpha_1, y_1, \alpha_1, y_1, \alpha_0\}$	8
h_{14}	$\{\alpha_1, \alpha_1, \alpha_1\}$	$\{y_0, \alpha_1, y_1, \alpha_1, y_0, \alpha_1\}$	3
h_{15}	$\{\alpha_1, \alpha_1, \alpha_1\}$	$\{y_0, \alpha_1, y_1, \alpha_1, y_1, \alpha_1\}$	9

The structure of a feedback or a closed-loop policy is shown in Figure 3. In both examples, we do not know how the actions control the system's operation.

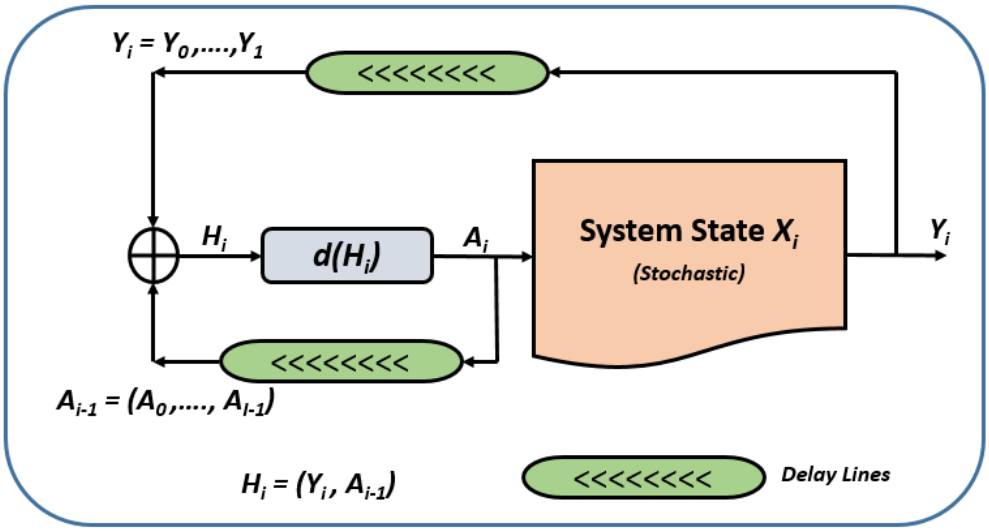


Figure 3. The Optimal Action Depends on the Action-Observation History

1.5 Fundamental Limitations of Learning and Optimization

1.5.1 Exhaustive Search is not feasible

A set of policies constitutes a policy space. To find an optimal policy in a given policy space is a typical search problem. However, even for a small problem, the policy space is too large for the exhaustive search approach. The number of policies increases exponentially with respect to the number of states. Therefore, exhaustive search, which requires computing or comparing the performance of every policy, is not computationally feasible for most practical problems.

1.5.2 Learning and Optimization

To develop efficient algorithms for performance optimization, we need to explore the special features of a system. This process is called ***learning***. For dynamic systems, learning may involve observing and analyzing a sample path of a system to obtain necessary information; this is in the normal sense of the word "learning", as it is used in research areas such as ***reinforcement learning***.

1.6 The Fundamental Limitations of Learning and Optimization

1. A system can be run and/or studied under only one policy at a time.

2. By learning from the behavior of a system under one policy, we cannot obtain the performance of other policies, if no structural information of the system is available.
3. We can only compare two policies at a time.

These simple rules describe the boundaries in developing learning and optimization approaches. First of all, if there is no structural information for the system, from the fundamental limitations A and B, we need to observe/analyze every policy to get or to estimate its performance, and from the fundamental limitation ***C***, for ***M*** policies we need to make ***M−1*** comparisons. This is the exhaustive search method.

1.7 Performance Gradients

As indicated by the fundamental limitations 1 and 2, if we analyze a system's behavior under one policy, we can hardly know its behavior under other policies. It is natural to believe that if two policies are "close" to each other, then the system under these two policies may behave similarly. If a policy space can be characterized by a continuous parameter θ, then two policies are "close" if their corresponding values for θ are close. Such a policy space is called a ***continuous policy space***. With some knowledge about the system structure under different policies, by studying the behavior of a system under one policy, we can determine the performance of the system under the policies in a small neighborhood of this policy; i.e., determine the performance gradient.

The gradient method does not apply to discrete policy spaces. For discrete policy spaces, we need to compare the performance of different policies that may not be close to each other. With some assumptions on the system structure, by studying the behavior of a system under one policy, we can find a policy that performs better, if such a policy exists. In summary, if no information about the system structure is available, we can only do exhaustive, or blind, searches in the policy space. There are different types of policies: those depending on the histories of both input actions and output observations, and those depending only on the current state. If we know something about the structure and the dynamics of the system, we may develop efficient optimization techniques (analytic, simulation, on-line, learning, etc.).

Different disciplines in learning and optimization, such as ***Markov Decision Processes (MDPs)*** in operations research, ***Perturbation Analysis (PA)*** in discrete event dynamic systems (DEDSs), ***reinforcement learning (RL)*** in computer science, and ***identification and adaptive control (I&AC)*** in control systems, have different formulations of the system structures. These different disciplines also differ in the way they utilize the structural information.

Roughly speaking, both MDPs and RL assume a Markov structure for the systems; PA started with a queuing network-type of structure and was extended to the Markov structure later; in I&AC, the system evolution is described by dynamic (differential or difference) equations. By and large, the MDP literature focuses on analytical solutions, and the parameters are usually assumed to be known to us. RL emphasizes the learning aspect, and algorithms are developed based on sample paths obtained from simulation. PA extracts information from a sample path to answer the what-if type of questions:

"What is the effect on a system performance if the system parameter or structure changes?"

This is done by predicting the system behavior after the parameter or structural changes.

A sample path of the Markov chain (see Figure 4) is denoted as $\boldsymbol{X=\{X_0, X_1,...\}}$, with $X_l \in S$ being the state at time i =0,1,....

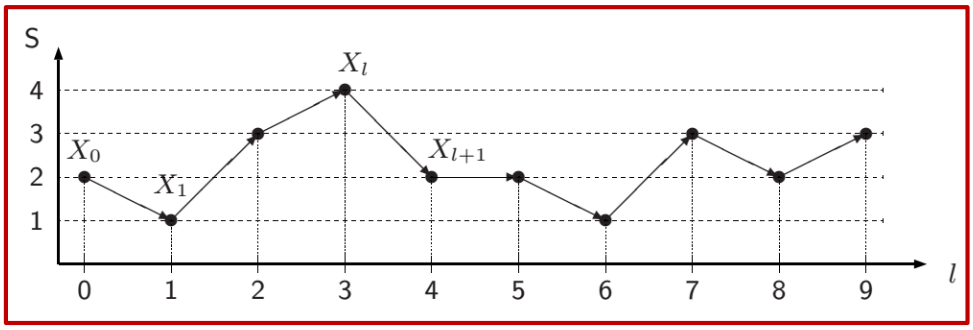


Figure 4. A sample path of a Markov Chain

We study the long-run average performance measure η of a ***Markov Chain*** defined as,

$$\eta(i) = \lim_{L\to\infty} \frac{1}{L} \sum_{i=0}^{L-1} E[f(X_i) I\, X_0 = i]$$

Three state Markov chain transition is described in figure 5,

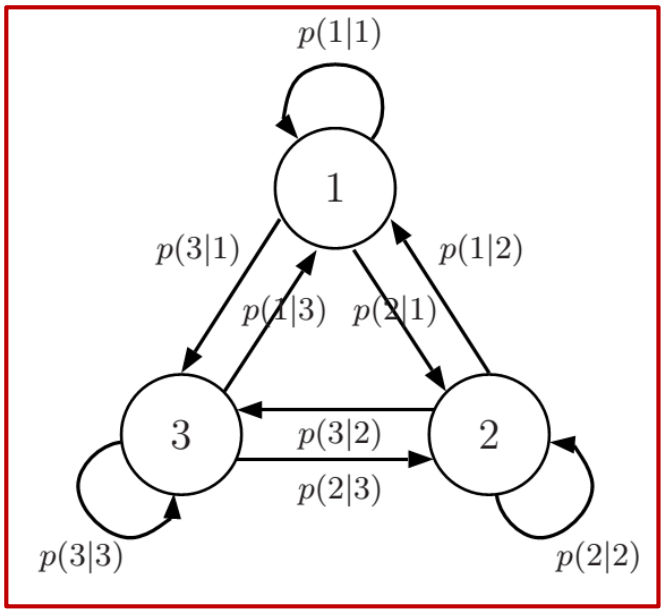


Figure 5. The State-Transition Diagram of a Three-State Markov Chain

Chapter 02

2.1 Perturbation Analysis

Perturbation analysis (PA) is the core of the gradient-based (or policy gradient) learning and optimization approaches. The basic principle of PA is that the derivative of a system's performance with respect to a parameter of the system can be decomposed into the sum of many small building blocks, each of which measures the effect of a single perturbation on the system's performance, and this effect can be estimated on a sample path of the system.

PA estimates the performance derivatives with respect to system parameters by analyzing a single sample path of a stochastic dynamic system. The most significant contribution of PA is that it testifies to the fact that a sample path of a dynamic system may contain information not only for the performance of the system under observation, but also for the derivatives of the performance with respect to system parameters. Perturbation analysis was first developed for queuing systems and was later extended to Markov systems.

2.2 Perturbation Analysis of Markov Chains

Consider an ergodic (irreducible and a periodic) Markov chain $\boldsymbol{X = \{X_l : l \geq 0\}}$ on a finite state space $\boldsymbol{S = \{1,2,\ldots ,S\}}$ with a transition probability matrix $\mathbf{P = [p(j|i)]S\ i,j=1}$. Its steady-state probabilities are denoted as a row vector $\boldsymbol{\pi = (\pi(1),\ldots ,\pi(S))}$ and the reward function is denoted as a (column) vector $\boldsymbol{f = (f(1),f(2),\ldots ,f(S))T}$. We have $Pe = e$, where $e = (1,1,\ldots ,1)T$, and the probability flow balance equation $\pi = \pi P$. We first consider the *long-run average reward* (or, simply, the average reward) as the performance measure, which is defined as follows:

$$\boldsymbol{\eta = E\pi(f) = \sum \pi(i)f(i) = \pi f}$$

where $E\pi$ denotes the expectation corresponding to the steady-state probability π on S.

$$\boldsymbol{P\delta = P + \delta\Delta P = \delta P + (1 - \delta)P}$$

with $0 \leq \delta \leq 1$ and $\boldsymbol{\Delta P = P - P := [\Delta p(j|i)]}$. Since $Pe = P\ e = e$, we have $\boldsymbol{(\Delta P)e = 0}$ **and** $\boldsymbol{P\delta e = e}$.

For simplicity, we first assume that the Markov chain with transition probability matrix $P\delta$ in (2.1) for all $0 \leq \delta \leq 1$ has the same reward function f,
and we denote it as $(P\delta,f)$. The steady-state probability of transition matrix $P\delta$ is denoted as $\pi\delta$ and

the average reward of the Markov chain (***Pδ,f***) is denoted as $\eta\delta = \pi\delta f$. Then $\eta 0 = \eta = \pi f$ **and** $\eta 1 = \eta = \pi f$. Set $\Delta\eta\delta = \eta\delta - \eta$. The derivative of $\eta\delta$ with respect to δ at $\delta = 0$ is

$$\left.\frac{d\eta_\delta}{d\delta}\right|_{\delta=0} = \lim_{\delta\to 0}\frac{\Delta\eta_\delta}{\delta},$$

which can be viewed as the directional derivative in the policy space along the direction from policy P to policy P. The goal of perturbation analysis is to determine the performance derivative $d\eta\delta\ d\delta$ by observing and/or analyzing the behavior of the Markov chain with transition probability matrix P. In particular, we wish to estimate this derivative by observing and analyzing a single sample path of the Markov chain with transition probability matrix P.

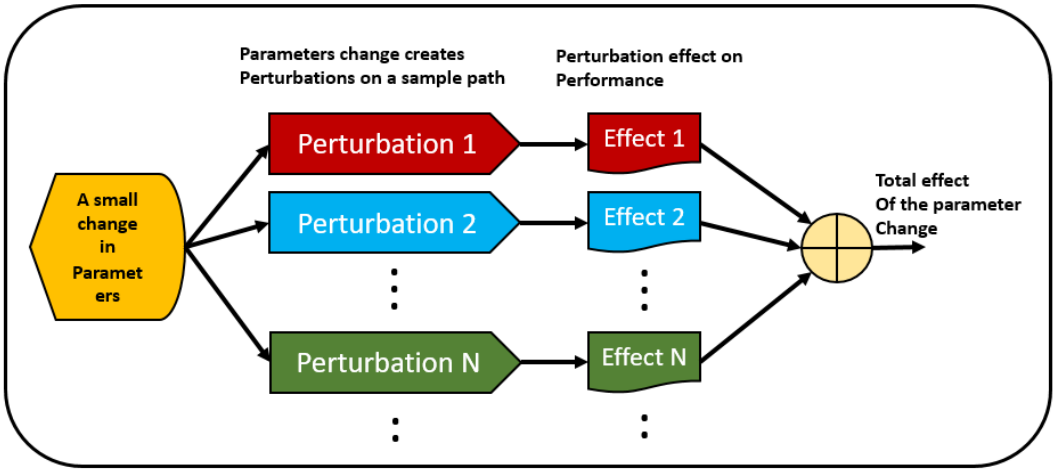


Figure 6: The basic principle of perturbation analysis

These basic principles are illustrated in figure 6. The important step in this approach is to determine the average effect of a single perturbation, i.e., the realization factor. The main idea of PA comes from the fact that given a sample path of the Markov chain with transition probability matrix P, we can construct a sample path of the Markov chain with transition probability matrix $P\delta$, when δ is small. Following the PA terminology, we call the Markov chain with transition probability matrix P the *original Markov chain* and that with $P\delta$ the *perturbed Markov chain*. Their sample paths are called the *original sample paths* and the *perturbed sample paths*, respectively.

We first review how to simulate a sample path for a Markov chain with transition probability matrix P. We generate a uniformly distributed random variable $\xi l \in [0,1)$. If

$$\sum_{k'=1}^{u_{l+1}-1} p(k'|k) \le \xi_l < \sum_{k'=1}^{u_{l+1}} p(k'|k), \qquad u_{l+1} \in \mathcal{S}$$

The performance measure η can be estimated from the sample path $\boldsymbol{X}$. In fact, if the Markov chain is ergodic, we have

$$\eta = \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} f(X_l), \qquad \text{w.p.1},$$

w.p.1 stands for with probability 1,

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

Then we have,

$$\eta = \lim_{L\to\infty} \frac{F_L}{L}.$$

To save computation, we may try to use the same sequence $\{\xi_0, \xi_1, \ldots, \xi_l, \ldots\}$ to generate the perturbed path (figure 7). However, we need to use

$$\sum_{k'=1}^{u_{\delta,l+1}-1} [p(k'|k) + \delta\Delta p(k'|k)] \le \xi_l < \sum_{k'=1}^{u_{\delta,l+1}} [p(k'|k) + \delta\Delta p(k'|k)]$$

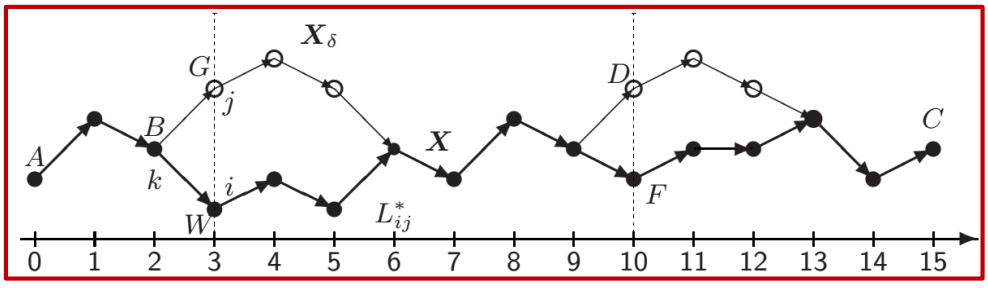


Figure 7: Constructing a Perturbed sample path

2.3 Perturbation Realization Factors and Performance Potentials

The sample path is perturbed from state i to state j. This perturbation will certainly affect the system's behavior and the system's performance. As shown in Figure 2.5, after $l = 3$, the perturbed Markov chain evolves differently from the original chain, until, at $l = L_{ij}$, the perturbed path merges with the original one. The effect of the perturbation takes place in the period from $l = 3$ to L_{ij}. In PA terminology, we say that the perturbation generated at $l = 3$ is realized by the system at $l = L^*_{ij} = 6$.

In summary, there are a number of advantages of PA: It can estimate performance derivatives along all directions based on a single sample path of a Markov chain. It can estimate derivatives along any direction on line as a whole, and the ‘‘curse of dimensionality’’ issue does not appear; furthermore, the approach applies to any policy space or subspace with constraints.

However, PA-based approaches may reach a local optimal point. In addition to PA of Markov systems, efficient algorithms were developed by PA principles for queuing systems; these algorithms utilize the special ‘‘coupling’’ feature among servers to determine the effect of a single perturbation. Recently, fluid model of queuing systems was introduced into PA, which provides good approximations.

2.4 Performance Derivation Formulas

To derive the performance derivative $\mathbf{d\eta\delta / d\delta}$ at policy (P,f) along any direction ΔP, we consider a sample path X with a transition probability matrix P consisting ofL, L>>1, transitions. Among these transitions, on average, there are $L\pi(k)$ transitions at which the system is in state k. Each time when X visits state is after visiting state k, because of the change from Pto $P\delta = P+\delta\Delta P$, the perturbed path Xδ may have a jump, denoted as from state i to state j(i.e., after visitingk, X moves to I and Xδ moves to j), as shown in Figure 2.5. For convenience, we allowi=jas a special case. A “real jump” (with i!=j) happens rarely. Denote the probability of a jump from it after visiting state k as p(i, j|k). We have, (see the scanned handwritten calculations on the next page)

$$P(i,j|k) = P(i|k)\underbrace{P_\delta(k,j|k,i)}_{*}$$

(*) is a ~~functional~~ conditional Probability. that X_δ moves from state k to state j given that X moves from k to i, ∴

$$\sum_{j=1}^{s} P(i,j|k) = P(i|k) \quad \text{— (A)}$$

$$\sum_{i=1}^{s} P(i,j|k) = P_\delta(j|k) \rightarrow \text{(B)}$$

As in (A) $i,j = 1$, the whole $A = 1$

∴ $\delta \rightarrow 0$

the probability that the markove chain jumps at $\ell = 3$ and X_δ from $\ell = 4$ to L_{ij}^*

then the Average effect due to change in P to $P_\delta = P + \delta\Delta P$ is:

$$E(F_{\delta,L} - F_L)$$

$$\simeq \sum_{k=1}^{s}\left[\sum_{i,j=1}^{s} L\pi(k)P(i,j|k)\gamma(i,j)\right]$$

$$= \sum_{k=1}^{s}\left\{\sum_{i,j=1}^{s} L\pi(k)P(i,j)|k)[g(j)-g(i)]\right\}$$

$$= \sum_{k=1}^{s} L\pi(k)\left\{\sum_{j=1}^{s}\left[g(i)\sum_{i=1}^{s} P(i,j|k)\right] - \right.$$

$$\left.\sum_{i=1}^{s}\left[g(i)\sum_{j=1}^{s} P(i,j|k)\right]\right\} \rightarrow \text{(C)}.$$

From (A) & (B), (C) becomes

$$E(F_{g,L} - F_L) \simeq \sum_{k=1}^{S} L\pi(k) \left\{ \left[\sum_{j=1}^{S} P_g(j|k)\, g(i) \right] - \left[\sum_{i=1}^{S} P(i|k)\, g(i) \right] \right\}$$

$$= \sum_{k=1}^{S} L\pi(k) \left\{ \sum_{j=1}^{S} [P_g(j|k) - P(j|k)] g(i) \right\}$$

$$= L\pi (P_g - P) g = L\pi(\Delta P) \delta g \quad \text{(D)}$$

Thus $\eta_g - \eta = \lim_{L \to \infty} \frac{1}{L} E(F_{g,L} - F_L) \simeq \pi(\Delta P) \delta g$ → (C).

Finally letting $\delta \to 0$, we obtain the performance derivative formula

$$\left. \frac{d\eta_g}{d\delta} \right|_{\delta = 0} = \pi(\Delta P) g \quad \rightarrow (d) \quad (*)$$

2.5 Performance sensitivity of Markov Processes

Markov processes. Consider an irreducible and aperiodic (ergodic) Markov process $X=\{X_t, t \geq 0\}$ with a finite state space $S=\{1,2,...,S\}$ and an infinitesimal generator $B=[b(i, j)]$, where $b(i, j) \geq 0$, $I \mathrel{!}= j$, $b(i, i) < 0$. Let π be the steady-state probability (row) vector.

We have $\pi e=1$ and

$$Be = 0, \qquad \pi B = 0.$$

We can construct an embedded Markov chain (discrete-time) that has the same steady-state probability as the Markov process X. This is called uniformization. Thus, the sensitivity analysis of a Markov process can be converted to that of a Markov chain, and then it can be translated to Markov processes.

2.6 Performance Sensitivities of Semi-Markov Processes

A semi-Markov process $X=\{X_t, t \geq 0\}$ defined on a finite state space $S=\{1,2,...,S\}$. Let $T_0, T_1, ..., T_l, ...,$ with $T_0 = 0$, be the transition epoches. The process is right continuous so the state at each transition epoch is the state after the transition. Let $X_l = X_{Tl}$, $l=0,1,2,....$ Then, $\{X_0, X_1, ...\}$ is the embedded Markov chain. The interval $[T_l, T_{l+1})$ is called a period and its length is called the sojourn time in state X_l.

2.7 The Embedded Chain and the Sojourn (Break) Time

The semi markov is defined as,

$$p(j;t|i) := \mathcal{P}\left(X_{l+1} = j, T_{l+1} - T_l \leq t | X_l = i\right),$$

Which we assume does't not depend on time homogeneous. Set,

$$p(t|i) := \sum_{j \in \mathcal{S}} p(j;t|i) = \mathcal{P}(T_{l+1} - T_l \leq t | X_l = i),$$

$$h(t|i) := 1 - p(t|i),$$

$$p(j|i) := \lim_{t \to \infty} p(j;t|i) = \mathcal{P}(X_{l+1} = j | X_l = i),$$

$$p(t|i,j) := \frac{p(j;t|i)}{p(j|i)} = \mathcal{P}(T_{l+1} - T_l \leq t | X_l = i, X_{l+1} = j).$$

Normally, $p(i|i) = 0$, for all $I \in S$. But, in general, we may allow the process to move from a state to itself at the transition epoches; in such a case, $p(i|i)$ may be nonzero and our results still hold. However,

a transition from a state to the same state cannot be determined by observing only the system states of a semi-Markov process.

The matrix [p(j|i)] is the transition probability matrix of the embedded Markov chain. We assume that this matrix is irreducible and aperiodic [20].

Let

$$m(i) = \int_0^\infty sp(ds|i) = E[T_{l+1} - T_l | X_l = i]$$

be the mean of the sojourn time in state i. We also assume that $m(i) < \infty$ for all $i \in S$. Under these assumptions, the semi-Markov process is irreducible and aperiodic and hence ergodic. Define the hazard rates as,

$$r(t|i) = \frac{\frac{d}{dt}p(t|i)}{h(t|i)},$$

and

$$r(j;t|i) = \frac{\frac{d}{dt}p(j;t|i)}{h(t|i)}.$$

The latter is the rate at which the process moves from I to j in [t, t+dt) given that the process does not move out from state i in [0,t).

2.8 Performance Sensitivity Formulas

Consider a semi-Markov process $X = \{(X_t, Y_t), t \geq 0\}$ with $T_0 = 0$ and an initial state $X_0 = j$. We define the reward function as f (i, j), i, j $\in$S, where f: S×S →R. The long-run average reward is,

$$\eta = \lim_{T\to\infty} \frac{1}{T} E\left[\int_0^T f(X_t, Y_t)dt \middle| X_0 = j\right],$$

Which does not depend on j because X is ergodic.

2.9 Perturbation Analysis of Queuing System

The early works on perturbation analysis (PA) focused on queueing systems. The idea of PA was first proposed for the buffer allocation problem in a serial production line and was first studied for queueing networks. The special structure of queueing systems, especially the interactions among different customers or different servers, makes PA a very efficient tool for estimating the performance derivatives with respect to the mean service times based on a single sample path. This section contains an overview of the main results of PA of queueing systems.

The main difference between PA of Markov chains and PA of queueing systems is that in the former, a perturbation is a "jump" on a sample path from one state to another due to parameter changes, while, in the latter, it is a small (infinitesimal) delay in a customer's transition time. Some queueing (such as the Jackson-type) networks can be modelled by Markov processes and therefore the theory and algorithms developed for Markov processes can be applied. However, because of the special features of a queueing system, the performance derivatives with respect to service time changes can be obtained by a much more efficient and more intuitive approach, which applies to non Markov queueing systems as well.

The dynamic nature of a system's behavior is explored more clearly in PA of queueing systems. Its basic principle can be described as follows: a small increase in the mean service time of a server generates a series of small delays, called perturbations, in the service completion times of the customers served by that server. Each such perturbation of a customer's service completion time will cause delays in the service completion times of other customers (at the same server or at other servers). In other words, a perturbation will be propagated through the system due to the interactions among customers and servers. Thus, a perturbation will affect the system performance through propagation. The average effect of a perturbation on the system performance can be measured by a quantity called the perturbation realization factor (PRF).

Finally, the effect of a change in the mean service time of a server equals the sum of the effects of all the perturbations generated on the service completion times of the server due to this change in its mean service time.

2.10 Three Fundamental rules of Perturbation Analysis

1. Perturbation generation
2. Perturbation propagation
3. Perturbation realization.

Figure 8, illustrates a sample path of a three-server five-customer closed queueing network. The vertical dashed arrows signal the customer transitions among servers, and each of the three staircase-like curves indicates the evolution of a server in the network.

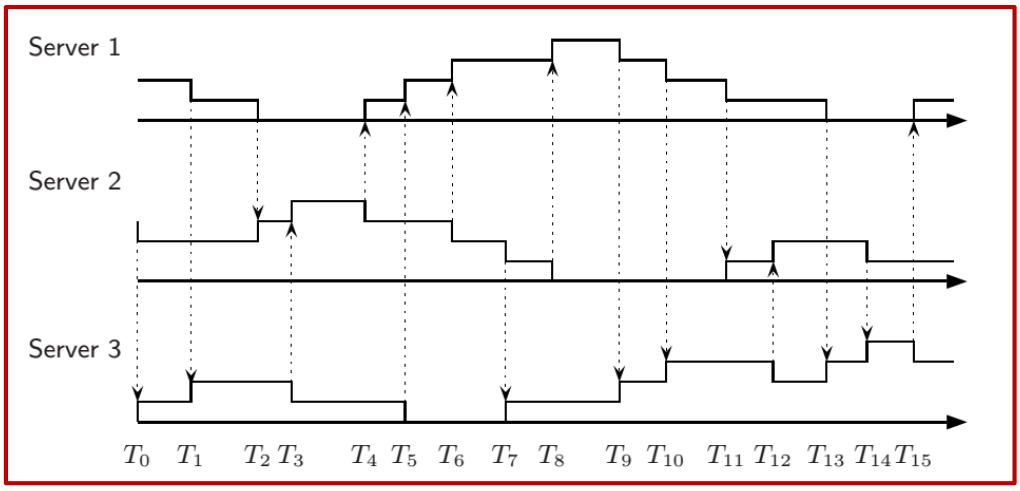


Figure 8: A Sample Path of a Closed Queueing Network with M= 3 and N=5

2.11 The Goal of Perturbation Analysis

The goal of PA is to obtain an estimate for the performance derivatives $\frac{d\eta^{(f)}}{d\bar{s}_v}$,v=1,2,...,M, by observing and analyzing an original sample path. This is shown in Figure 9, in which we use θ to denote a generic parameter.

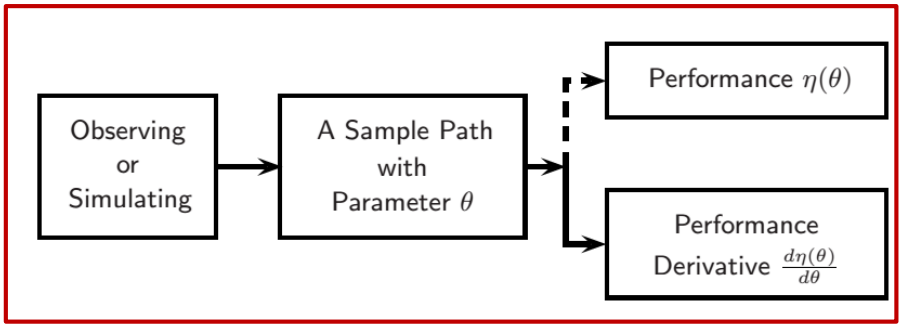


Figure 9: Goal of PA

Chapter 03

3.1 Learning and Optimization with Perturbation Analysis

In this chapter, the author first discussed the numerical methods and sample-path-based algorithms for estimating performance potentials, and then derived the sample-path-based algorithms for estimating performance derivatives. In performance optimization, the process of estimating the potentials and performance derivatives from a sample path is called ***learning***. This chapter is closely related to ***reinforcement learning***.

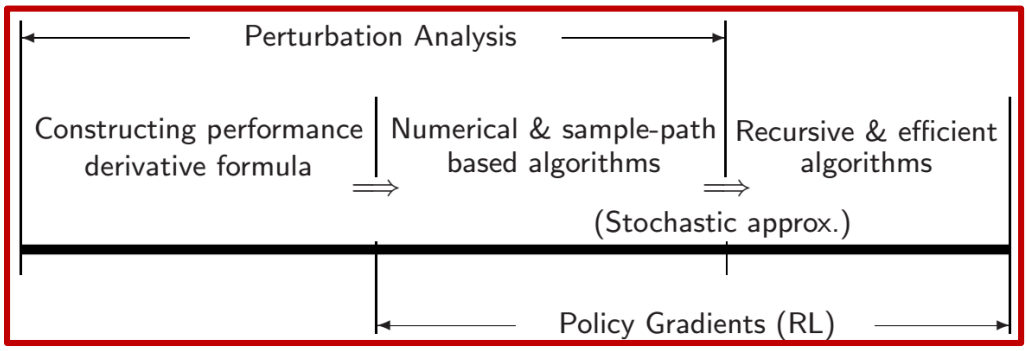


Figure 8: Perturbation Analysis vs. Policy Gradients

The authors first study the potentials for ergodic Markov chains (discrete time), and the results can be extended to ergodic Markov processes (continuous time) naturally. The first numerical method depends on the equation for performance potentials. PA and PG are modeled in figure 8 as a qualitative comparison.

3.2 The Fundamental Ergodic Theorem

This theorem is very useful in proving the convergence results related to sample-path-based algorithms, we will refer to it as the *Fundamental Ergodicity Theorem. It can be represented as*

Let $\boldsymbol{X} = \{X_n, n \geq 0\}$ be an ergodic Markov chain on state space S; $\varphi(x_1,x_2,...)$, $x_i \in S$, $i = 1,2...$, be a function on S^{∞}. Then the process $\boldsymbol{Z} = \{Z_n, n \geq 0\}$ with $Z_n = \varphi(X_n,X_{n+1},...)$ is an ergodic Markov chain.

In particular, we have

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{L-1} \phi(X_n, Xn+1, \dots\dots) = E[\phi(Xn, Xn+1, \dots\dots)]$$

where "*E*" denotes the steady-state expectation of the Markov chain Z, and the right-hand side of this equation does not depend on *n*. On a sample path of $\boldsymbol{X} = \{X_l, l \geq 0\}$ with $X_0 = i$, for each pair of states *j* and *i*, we define two sequences of epochs $\{l_k(j)\}$ and $\{l_k(i)\}$ as follows: diagrammatic representation is in figure 9,

$l0(i) = 0$,

$lk(j)$ = the epoch that $\{Xl\}$ first visits state *j* after $lk-1(i)$, $k \geq 1$,

$lk(i)$ = the epoch that $\{Xl\}$ first visits state *i* after $lk(j)$, $k \geq 1$. (3.18)

Note that $\{lk(j)\}$ and $\{lk(i)\}$ are well defined on a sample path;

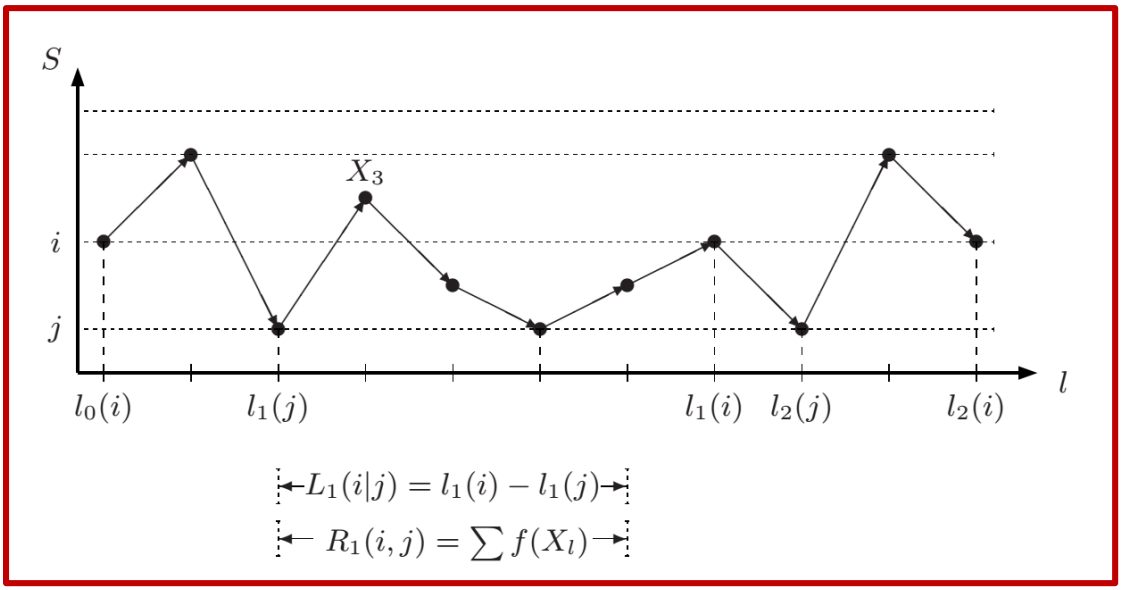


Figure 9: Estimating $\gamma(i,j)$

Introducing co-relation between the two sample paths $\boldsymbol{X}$ and $\boldsymbol{X}$ is called the ***coupling approach*** in simulation. In fact, it is well known that introducing co-relation between the random samples of two random variables may reduce the variance in estimating the difference of their mean values.

The coupling method is generally used in simulation to reduce the variance of the estimates for the difference of the mean of two different random variables. Applying this approach to estimate $\gamma(i,j) = g(j) - g(i)$ with two coupled sample paths still requires further research and we will not discuss the details in this book. Problems 3.9 and 3.10 provide a brief introduction to this variance-reduction simulation approach.

The performance potentials obtained by numerical methods or by learning from sample paths can be used to calculate the performance derivatives using the performance derivative formula

$$\frac{d\eta_\delta}{d\delta} = \pi(\Delta P)g.$$

One disadvantage of this approach is that it requires us to estimate the potentials for all the states. This is sometimes difficult for a number of reasons: The number of states may be too large; some states may be visited very rarely; and for systems with special structures (e.g. queueing networks), it may not be convenient even to list out all the states. Therefore the performance derivatives can be estimated directly from sample paths without estimating each individual potential.

It is interesting to note the difference in the process of developing the PA theory for both queuing systems and Markov systems. For queuing systems, the performance derivative estimation algorithms were developed first, and the concept of the perturbation realization factor and the performance derivative formula were developed later to provide a theoretical background for the algorithms. For Markov systems, the concept of performance potentials and performance derivatives were developed first, and the sample-path-based algorithms, both for potentials and for derivatives directly, were proposed later, by using the formulas. The algorithms for estimating $c(f)(\boldsymbol{n},v)$ in queuing systems should be easy to develop; however, there has not been much effort in this direction, perhaps because there have not been many applications with $c(f)(\boldsymbol{n},v)$ alone so far.

3.3 Optimization:

The PA gradient estimates can be used to implement sample-path-based performance optimization. When the sample path is long enough, the estimates are very accurate and we can simply use them in any gradient-based optimization procedure for deterministic systems. If the sample path is short, then the gradient estimates contain stochastic errors, and stochastic approximation techniques have to be used in developing optimization algorithms.

In sample-path-based implementation, the gradient estimation error depends on the length of the sample path. Therefore, the convergence of the optimization algorithm relies on the coordination among the lengths of sample paths in every iteration and the step sizes.

There are two ways to implement optimization algorithms with PA.

- First, we can run a Markov, or a queuing, system under one set of parameters for a relatively long period to obtain an accurate gradient estimate and then update the parameters accordingly. When the estimation error is small, we hope that this standard gradient-based method for performance optimization of deterministic systems works well.
- Second, when the sample paths are short, we need to use the stochastic approximation based algorithm. It is well known that the standard step size sequence (e.g., $\kappa k = k$ 1) makes the algorithm very slow, so some ad hoc methods are usually used in practice to speed up the convergence.

3.4 Optimization with Perturbation Analysis (PA)

3.4.1 Gradiant Methods

The PA gradient estimates can be used to implement sample-path-based performance optimization. When the sample path is long enough, the estimates are very accurate and we can simply use them in any gradient-based optimization procedure for deterministic systems. If the sample path is short, then the gradient estimates contain stochastic errors, and stochastic approximation techniques have to be used in developing optimization algorithms.

3.4.2 Sample path based implementation

In sample-path-based implementation, the gradient estimation error depends on the length of the sample path. Therefore, the convergence of the optimization algorithm relies on the coordination among the lengths of sample paths in every iteration and the step sizes.

As discussed, ***there are two ways to implement optimization algorithms with PA.***

First, we can run a Markov, or a queueing, system under one set of parameters for a relatively long period to obtain an accurate gradient estimate and then update the parameters according to (3.51). When the estimation error is small, we hope that this standard gradient-based method for performance optimization of deterministic systems works well.

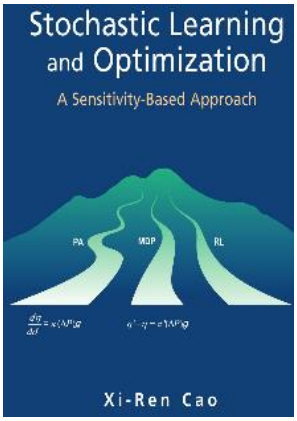
Second, when the sample paths are short, we need to use the stochastic approximation based algorithm.

3.5 Applications of Perturbation Analysis

There have been hundreds of papers in the area of PA and its applications in the literature, few well knows application areas are as follow,

1. Capacity planning
2. Inventory problems
3. Resource allocation
4. Flow control
5. Bandwidth provisioning
6. Traffic shaping
7. Pricing
8. Stability and reliability analysis
9. Network Communications
10. Manufacturing and logistics.

4. Referenced book



This book is written by Xi-Ren Cao, from Hong Kong University of Science and Technology (HKUST). This book is primarily written on performance optimization of modern engineering systems. As Performance optimization is very important in the design and operation of modern engineering systems in many areas, including communications (Internet and wireless), manufacturing, robotics, and logistics. Most engineering systems are too complicated to be modelled, or the system parameters cannot be easily identified. Therefore, learning techniques have to be utilized. The book is available online on the following link,

https://link.springer.com/book/10.1007%2F978-0-387-69082-7

Acknowledgment

I am really thanks to ***Prof. Yin BaoQun*** for his kind support during this course. He is very kind, humble, supportive and hardworking professor. He is very sound in theoretical aspects of mathematics and information science. He taught us the Mathematical Theory in Information Science course by selecting some good topics from a well know reference book "***Stochastic Learning and Optimization – A Sensitivity based Approach by Xi-Ren Cao***" and discussed all the nitty gritty of the basic concepts of learning using Markov Chain theory and Perturbation Analysis in a very detail manner in the class. I have learnt a lot in this course by myself that I never learnt before. Thanks to Prof. Yin Baoqun.

My detailed webpage (blog) is http://jadoon956.wordpress.com