COMPUTING SYSTEM RELIABILITY MODELING, ANALYSIS, AND OPTIMIZATION

LONG QUAN

(B.Eng., USTC)

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2008

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Prof. Xie Min for his great guidance, suggestions, patience and encouragement throughout my whole research work and life. I have learnt a lot from his knowledge as well as attitude of dealing with work. I would also give my thanks to my vice advisor Dr. Ng Szu Hui for her helpful suggestions on my research. This dissertation would not have been possible without their help.

I wish to thank the Department of Industrial & Systems Engineering for using its facilities. I would also like to thank other faculty members for the modules I have ever taken: Prof. Goh, Prof. Poh, Prof. Ong, Dr. Chai and Dr. Ng Kien Ming. Also I would like to thank Ms. Ow Lai Chun and the ISE Computing Lab technician Mr. Cheo for their kind assistance.

I would also like to express my thanks to my friends Hu Qingpei, Liu Xiao, Jiang Hong, Zhu Zhecheng, to name a few, for the joy they have brought to me. Specially, I would like to thank my colleagues in ISE departments from both seniors and juniors. They are Dai Yuanshun, Zhang Lifang, Liu Jiying, Sun Tingting, Cao Chaolan, Zhang Caiwen, Zhang Haiyun, Qu Huizhong, Muthu, Pan Jie, Wei Wei, and Yao Zhishuang for the support and help.

At last, I am grateful for the love and support from my family in China and Miss Yuan Le. Their understanding, patience and encouragement have been a great source of motivation for me to pursue my Ph. D.

TABLE OF CONTENTS

SUMMARY

This thesis investigates some important issues related to reliability modeling and analysis of various computing systems. Problems of optimization and resource allocation strategies are addressed as well for better utilizing the resources to improve computing system reliability.

In terms of configurations, executing manners and functionality, computing systems accomplish computing tasks in various forms, such as weighted voting systems, peer-to-peer network systems and etc. This makes quantitatively modeling system reliability difficult but even more necessary.

Traditional reliability models of weighted voting systems in literature assume binary or discrete state input. However, in practice, the phenomenon under test by weighted voting systems (WVS) is likely to be continuous, e.g. temperature, pressure, and etc. Research of reliability modeling and analysis on WVS are initially proposed by incorporating continuous state input. In this model, the concept of reliability is redefined to differentiate it from traditional models. Analytical as well as Monte Carlo Simulation methods are proposed to estimate the system reliability. As different types of voting units are assumed to have different accuracies and costs, the different allocations of these voting units make the reliability of the entire voting system different. A reliability optimization problem with cost constraints is then formulated and solved by genetic algorithm. The best solution improves the system reliability efficiently. Further analysis on the reliability model of WVS is also presented by considering system biased output and dependent accuracy of the units to the input.

Results show that the reliability of the biased voting system is lower than the unbiased voting system, given the same accuracy of the system.

Peer-to-peer media streaming system is widely used today. Its reliability is affected not only by software/hardware but also by unsteady network communication. This thesis constructs original general models for p2p media streaming system and introduces new analytical method to estimate service reliability it provides.

In order to apply the models to predict the reliability of the system, the parameters of the models need to be known or estimated. Parameter uncertainty arises when the input parameters are unknown. Moreover, the reliability computed from the models which are functions of these parameters is not sufficiently precise when the parameters are uncertain. This dissertation studies the uncertainty problems in reliability modeling first at component-level then further extends the uncertainty analysis to more complicated systems that contain numerous components, each with its own respective distributions and uncertain parameters. This method is also applied to weighted voting system to explore its uncertainty in reliability calculation and parameters estimation from scarce data.

For complex engineering systems, the components or subsystem are likely vulnerable to the mis-operations or intentional attacks. Preventive investment in the components is necessary to guarantee the safety critical systems to work properly and in high performance. Under resource budget, it is important but difficult to find out the resource allocation strategy to improve system reliability optimally. This dissertation presents a new preventive resource allocation strategy by introducing an important phenomenon of apical dominance in plant growth process.

List of Tables

List of Figures

CHAPTER 1 INTRODUCTION

This dissertation focuses on reliability modeling, analysis and optimization of some practical systems. The key issues include system reliability, software reliability, network reliability, weighted voting system, peer-to-peer system, uncertainty analysis, parameter estimation, optimization, and resource allocation strategy.

This chapter briefly introduces the background and some basic concepts of reliability theory, presents some important methodologies used in reliability modeling, analyzing, and optimization, and figures out the scope of this dissertation.

1.1 Background

Reliability is an important time-based measure of quality; which has received much attention in recent decades. Reliability is defined by Musa (1998) as the probability that a system will perform a required task during a period of time without any failure under the stated conditions.

Along with the explosive development of information technology in the recent decades, the concept of computing systems has been widely accepted to many practical areas. It is a kind of system of one or more computers/processors and associated software with common storage, which process data in a meaningful way. The size and complexity of the computing systems has increased exponentially in terms of the structure, number of components, computing tasks and etc, which makes assessment and modeling the performance of computing systems hard or costly. Under this background, reliability of computing system is a necessary metric to measure the system performance, which is generally defined as the probability that the output it produces is correct in given period of time under specified computing environment.

Most computing systems contain both software programs and hardware to achieve the various computing tasks and complete various services. The faults in software programs or hardware devices can result in the failure of the entire computing system in getting satisfactory services.

The computing tasks are executed on the support of hardware configurations, such as computers, processors, memories and so on. And these hardware devices generally work together in some meaningful organized structures. For example, in *k*-out-of-*N* voting configuration, the requisite to successfully accomplish computing tasks is that at least *k* hardware components are in operation out of total number of *N* components. A weighted voting system is a type of system in this configuration, of which each component (voting unit) is assigned with different weights to vote (Levitin, 2001). Network configuration is complex and hard to analyze, in which peer-to-peer systems and grid systems organize themselves to achieve their goals. Other fundamental and common configurations include series, parallel, bridge, and etc.

Besides the hardware, software is another important component in completing the computing tasks successfully. Software system has different properties from hardware, it does not wear-out and can be easily reproduced, software testing will be incomplete because of the complexity of software, and software requires different fault-tolerance techniques than hardware (Xie et al. 2004 and Pukite & Pukite, 1998). Software reliability can be improved over time accounting for faults detection and correction (Xie, 1991). So the way of modeling and analyzing software reliability is much different from hardware systems. Among all the software reliability models, Markov models are the most famous and fundamental, first proposed by Jelinski & Moranda (1972). Following that, many successful models are proposed, including Littlewood model (1979) and GO model (1979).

1.2 Methodologies

1.2.1 Markov Theory

Markov Modeling is a widely used technique in reliability analysis; it is flexible and effective to be implemented in reliability analysis for various computing systems. Xie *et al.* (2004) classify the Markov models into two major types: standard Markov models and non-standard Markov models, in which Markov property are not valid at all time.

According to their time space and state space, Markov model is classified into four categories: discrete time Markov chain, continuous time Markov chain, discrete time continuous state Markov model, and continuous time continuous state Markov model.

For the first type of Markov model, discrete time Markov chain, the mathematical definition is

$$\Pr\{X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0\} = \Pr\{X_{n+1} = j \mid X_n = i\} = P_{ij} \qquad (1.1)$$

where X_n=i denote the process in state i at time n, and P_{ij} is named one step transition probability from state i to state j.

Discrete time Markov chain is a widely used technique in system reliability analysis. Wang (2002) use Markov chain to calculate the reliability of distributed computing system by introducing two reliability measures, which are Markov chain distributed program reliability (MDPR) and Markov chain distributed system reliability (MDSR).

Continuous-time Markov chain (CTMC) $\{X(t)\}$, having values on the discrete state space Ω, is defined as the stochastic process satisfies following property:

$$\Pr\{X(t+s)=j \mid X(s)=i, X(u), 0 \le u \le s\} = \Pr\{X(t+s)=j \mid X(s)=i\} \qquad (1.2)$$

where $s \ge 0$, t>0 and each $i, j \in \Omega$. A *CTMC's* future state depends only on the present state and is independent of past, given the present state. For CTMC models, we have the Chapman-Kolmogorov equation (Ross, 2000) as:

$$p_{ij}(s,t) = \sum_k p_{ik}(s,u) p_{kj}(u,t), \qquad o \le s < u < t \qquad (1.3)$$

Many researchers apply continuous time Markov chain to formulate the hardware system, software system and distributed computing system to evaluate and analyze the system reliability (service reliability). Dai *et al.* (2003a) incorporate GO model into continuous time Markov chain model to evaluate the service availability. Gokhale *et al.* (2004) use a non-homogeneous continuous time Markov chain to analyze the effect of various kinds fault removal policies on the residual number of faults at the end of the testing process and extend the model to include imperfections in the fault removal process.

Markov models with continuous state are classified into two groups according to the time space: discrete time and continuous time. However, little research has been done on these two types of models, because the complexity and immense computation, so the continuous state Markov process will not be discussed in this proposal.

Non-standard Markov models include semi-Markov process and Markov regenerative process. The semi-Markov process was introduced in 1954 by Levy to provide a more general model for probabilistic systems. In a semi-Markov process, time between transitions is a random variable that depends on the transition. The discrete and the continuous-time Markov processes are special cases of the semi-Markov process. Becker (2000) uses a non-homogeneous semi-Markovian process to model reliability characteristics of components or small systems with complex test resp. maintenance strategies, in which the transition rates depend on two types of time in general: on process time and on sojourn time in one state.

1.2.1 Universal Generating Function

Universal Generating Function (UGF) is a well-known and effective technique for the reliability analysis and optimization of various multi-state systems. Much research has been done on incorporating UGF into reliability analysis of various series-parallel systems, bridge systems, weighted voting systems, acyclic transmission networks, linear multi-state sliding-window system, linear consecutively connected systems, and acyclic consecutively connected networks. Lisnianski & Levitin (2003) briefly describe the application of UGF in many systems; Levitin (2005) provides a generalized view of the method and its application to analysis and optimization of various types of binary and multi-state system.

Levitin *et al*. (1998) generalize a redundancy optimization problem to multi-state series-parallel systems, and use UGF to represent the availability of the multi-state system. Levitin & Lisnianski (1999a) formulates the joint redundancy and replacement schedule optimization problem, where the reliability is evaluated by UGF. Levitin & Lisnianski (1999b) provide an effective importance analysis tool for complex series–parallel multi-state systems based on UGF and extend this method to sensitivity analysis of important output performance measures. Levitin & Lisnianski (2001a) consider series-parallel systems with two failure modes; the reliability of the multi-state system is evaluated by UGF and optimized by Genetic Algorithm. Levitin

& Lisnianski (2001b) and Levitin (2002c) apply UGF as the evaluating method of the series-parallel multi-state systems.

UGF is also an effective evaluation tool to the multi-state system in bridge topologies. Levitin and Lisnianski (2000) evaluate the reliability of bridge system consisting of elements with different reliability and performance by UGF. Other application of UGF to the reliability analysis of bridge system can be found in Levitin (2003a), and Lisnianski *et al.* (2000).

Weighted voting system is another important multi-state system; UGF is widely applied to reliability analysis of weighted voting system. Levitin and Lisnianski (2001) provide a method to evaluate the reliability of weighted voting system based on UGF. Other similar method to evaluate reliability of weighted voting system can be found in Levitin (2002a) and Levitin (2002b).

Other applications of UGF to the reliability analysis of various multi-state systems are described in Levitin (2005) in detail.

1.2.2 Bayesian Theory

The Bayesian approach combines the prior knowledge/information of the unknown parameter with current data/observations to deduce the posterior probability distribution of the parameter. Moreover, this approach can also handle the correlation among those parameters by using the joint distributions.

To estimate the parameters $\vec{a} = \{a_1, a_2, ..., a_m\}$, observation data $\vec{s} = \{s_1, s_2, ..., s_n\}$ are collected by repeated experiments. Then, given the prior distribution $p(\vec{a})$ and observations $\vec{s} = \{s_1, s_2, ..., s_n\}$, the posterior distribution can be obtained by

$$p(\vec{a} \mid \vec{s}) \propto p(\vec{a}) \cdot p(\vec{s} \mid \vec{a}) \tag{1.4}$$

where

$$p(\vec{s} \mid \vec{a}) = \exp\{-m(s_n \mid \vec{a})\} \cdot \prod_{i=1}^{n} \lambda(s_i \mid \vec{a}) \tag{1.5}$$

The above standard Bayesian approach is well known and straightforward. However, applying this to software reliability modeling poses several challenges specific to software testing and reliability. It is an important characteristic that the number of failure data is usually scarce in a single test. The lack of failure data in a project has challenged the modeling of software reliability, which makes estimating proper posterior distributions more difficult.

1.2.3 NHPP

NHPP is a special class of counting process $\{N(t), t \geq 0\}$ to cumulate the number of events in a time interval $[0,t)$ with rate parameter $\lambda(t)$ such that the rate parameter of the process is a function of time. It can be classified as a very special case of the Non-Homogeneous Continuous Time Markov Chain models, see e.g. Gokhale et al. (1997). An classic example of an NHPP would be the arrival rate of faults or failures to a software system over the specified period. The faults would be detected in a higher rate at the beginning stage. The first application of NHPP in software reliability modeling can be found in classi G-O model (Goel and Okumoto, 1979).

1.3 Motivation

The computing system reliability models have been successfully applied in practice, and until now there are currently a number of practical papers summarizing their application experience (Xie et al., 2004). However, with the development of information technology and exponentially growing of complexity of the computing systems, the research on computing system reliability is necessary and everlasting. Therefore, research on some new developed computing systems, such as weighed voting systems, p2p computing system, grid computing systems, and etc, analysis on current reliability models, and strategies of optimal resource allocation have been underway. Based on this, research within the context of this thesis is conducted through the following specific topics.

1.3.1 Reliability of Weighed Voting Systems

Weighted Voting Systems (WVS) have attracted a lot of attention recently (see, e.g., Levitin, 2003, 2004, 2005a, Xie and Pham, 2005) as such systems are widely used in pattern recognition, human organization systems and technical decision making systems. They are a generalization of traditional k-out-of-n systems, with the following properties: each voting unit makes individual independent decision; each voting unit has its weight; and the decision of the system is based on the information from the individual voting units of the system. The entire weighted voting system reliability is defined as the probability that the system can successfully vote a correct output, which depends on the unit weights and the system threshold (Levitin and and Lisnianski 2001).

However, the limitation of the current models is that the inputs of the WVS have very small state spaces. Moreover, with increased input states, the number of different combinations of output increases significantly, increasing considerably computational complexity of the systems reliability. Furthermore, in many practical cases, the state of the input of the voting systems is continuous or approximately continuous and not discrete.

1.3.2 Reliability of Peer-to-peer Systems

Peer-to-peer (P2P) systems have recently received increasing attention from both research (see e.g. Leuf, 2002, Gong, 2002, Foster & Iamnitchi 2003, etc.) and industry. P2P system is a large-scale distributed system where there is no central server that stores all data. All data are distributed among nodes/peers which have the ability to self-organize. In P2P systems, peers cooperate to achieve a desired service, such as: distributed computing (Anderson et al., 2002), file sharing (Saroiu et al., 2001), distributed storage (Rowstron and Druschel, 2001), communication (see e.g. Jabber), and real time media streaming (Hefeeda et al., 2003).

From the perspective of the users of P2P media streaming systems, the most significant concern of the users is the performance of the software when downloading the huge volume of media data from a highly dynamic and unstable internet environment. The demanding users might have high requirement on the quality of media service provided by the P2P media streaming software. The P2P live media steaming software product with desirable features of running smoothly, recovering promptly from a sudden failure, high quality of the live media and etc will be attractive the users and outperform other similar competing P2P live media streaming products in the market. Hence, it would be very important to evaluate the service quality accurately and quickly to better develop the product further and compete to other products. However, to the best of our knowledge, no research has been done on measuring and modeling the performance of P2P media streaming network systems from the users' perspective.

1.3.3 Uncertainty Analysis of Reliability Models

Reliability modeling has gained considerable interest and acceptance by applying probabilistic methods to the real-world situation. A software usually contains one or more basic modules or components that are functioning together to achieve some tasks. These modules can be of various types resulting in a wide range of software and system reliability models proposed, e.g. Pham (2000), and Xie et al. (2004), Myrtveit et al. (2005).

In order to apply the models to predict the reliability of the component, the parameters of the models need to be known or estimated. Parameter uncertainty arises when the input parameters are unknown. Moreover, the reliability computed from the models which are functions of these parameters is not sufficiently precise when the parameters are uncertain. Hence, it is necessary to determine the uncertainty in the parameters for the modeling work.

However, one special characteristic of software reliability modeling or testing is insufficient failure data, see e.g. Miller et al. (1992). Failure data are usually scarce and limited to a single test. Insufficient failure data makes software reliability modeling difficult, and makes its uncertainty analysis much more challenging. Though some previous research (e.g. Jewell, 1985, Yin & Trivedi, 1999) note this problem and suggest using the Bayesian approach to incorporate historical data into prior distributions, however they do not propose a systematic and practical approach on how to incorporate experts' suggestions with historical data for uncertainty analysis. For instance, Yin & Trivedi (1999) simply assumed the prior distribution is known, using for example a uniform distribution as a prior. They do not introduce how to comprehensively derive it from experts' suggestions and historical data.

1.3.4 Preventive Resource Allocation Strategy

For complex engineering systems, their components or subsystem are vulnerable to mis-operations or intentional attacks. Preventive investment in the components is necessary to guarantee the safety critical systems to work properly and in high performance. Under resource budget constraints, it is important to find the resource allocation strategies to improve system reliability optimally. However, it is very difficult to obtain such optimal strategies because the engineering systems are quite complex. For example, the systems may be in different configurations that some components are selectively important. The efficiency of resources in improving the different components might differ as well. Addtionally, the components and subsystems are potentially exposed to different levels of intentional attacks. Mixture of these all makes the whole problem difficult. Moreover, most existing research has just focused on engineering systems in comparatively simple configurations, such as series, and/or parallel configurations.

1.4 Research Objective and Scope

The purpose of this thesis is to develop more comprehensive and practical models to measure reliability of different distributed systems, to conduct detailed quantitative analysis on the reliability models to estimate the uncertainties and parameters, and to optimize the system reliability by finding resource allocation strategies in different ways.

The remainder of the thesis is organized as follows. Chapter 2 provides comprehensive review of the existing research on system reliability models and some related system analysis topics.

Chapter 3 to chapter 5 study two different distributed systems that are widely used in practice. Chapter 3 studies the reliability of Weighed Voting Systems with continuous state input. A new analytical model for the reliability of WVS system is formulated and the reliability optimization problem for WVS under cost constraints is analyzed. Chapter 4 considers the bias properties of the system output for WVS and looks into three cases where the system has different bias and accuracies. Chapter 5 formulates a new reliability model to estimate the service reliability of Peer-to-Peer media steaming network systems, with service quality considerations.

Chapter 6 and chapter 7 study the problems of uncertainty analysis and parameter estimation for software reliability models and WVS reliability models. Chapter 6 quantifies the uncertainties in the software reliability modeling of a single component with correlated parameters and in a large system with numerous components and solves the challenge of lacking failure data by Bayesian method. With the similar technique, chapter 7 studies the uncertainty problem in reliability modeling of distributed detection systems where the parameter is estimated from historical experiment data.

After studying reliability models for different systems, chapter 8 discusses possible preventive resource allocation strategies, which is enlightened from a famous phenomenon in botany-apical dominance to improve the system reliability efficiently.

Chapter 9 summarizes the thesis and suggests some possible further extensions related to the thesis.

CHAPTER 2 LITERATURE REVIEW

Computing systems contain both hardware systems and software systems. Hardware, such as computer, CPU, processor, storage, memory etc., provides the fundamental configurations to support software system accomplish computing tasks successfully. Reliability modeling and analysis of hardware systems and software systems are actually equivalently critical to the entire computing system. Much important research has been done on reliability analysis and modeling.

This chapter reviews and summarizes some important related work. The remainder of this chapter is organized as follows. Section 2.1 discusses the existing literatures on reliability models of weighted voting systems, and section 2.2 briefly introduce two recently developed network systems, grid systems and P2P systems, and reviews some related research on these two systems. Section 2.3 focuses on the literatures on reliability models of software systems and, lastly, section 2.4 summarizes the research work in the area of optimization technique.

2.1 Reliability Models of Weighted Voting Systems

Xie *et al*. (2004) and Pukite and Pukite (1998) classify hardware system into single component system, parallel configurations, load-sharing configurations, and standby configurations. Among the above configurations, parallel system is one of the most frequently used redundancy configurations in computing systems. In parallel system, the failure of the entire system can only occur in case that all the parallel components fail, this property ensures high reliability of the system.

Two kinds of parallel systems are studied abundantly and widely used in industry: *k*-out-of-*n* systems and voting systems. *k*-out-of-*n* system is well covered in Kuo & Zuo (2002), it is categorized into non-repairable *k*-out-of-*n* system, repairable *k*-out-of-*n* system and weighted *k*-out-of-*n* system. Optimal design and other topics are also provided in their book. Levitin (2005) introduces universal generating function in reliability evaluate of *k*-out-of-*n* systems.

Weighted voting systems (WVS) has attracted a lot of attention recently (see, e.g., Levitin, 2003b, 2004, Xie and Pham, 2005) as such systems are widely used in pattern recognition, human organization systems and technical decision making systems. They are a generation of traditional k-out-of-n systems, with the following properties:

1. Each voting unit makes individual independent decision
2. Each voting unit has its weight

3. The decision of the system is based on the information from the system units.

Such a model is a dynamic threshold weighted voting system subject to two failure modes. System units and their outputs are subject to different errors. For the weighted voting systems, the unit errors are defined into three types. The systems incorporate all the unit decisions into one unanimous system output *D*, with the following rules:

$$D(I)=\begin{cases}1, & if \sum_{d_j(I)\neq x} w_j d_j(P) \geq \tau \sum_{d_j(I)\neq x} w_j,\ \sum_{d_j(I)\neq x} w_j \neq 0 \\ 0, & if \sum_{d_j(I)\neq x} w_j d_j(P) < \tau \sum_{d_j(I)\neq x} w_j,\ \sum_{d_j(I)\neq x} w_j \neq 0 \\ x, & if \sum_{d_j(I)\neq x} w_j = 0\end{cases} \tag{2.1}$$

where $d_i(I)$ is output of unit *i*, *x* represents the state that units abstain from voting, τ is the preset threshold value and w_j is the weight assigned to j^{th} unit.

The system fails if *D*(*I*) is not equal to 1. The entire weighted voting system reliability is defined as the probability of D(I)=1. This depends on the unit weights and the system threshold. Nordmann and Pham (1998) first proposed the formula for calculating reliability of WVS as

$$R=\sum_{S_x\subseteq I}\left[\prod_{i\in S_x} q_x^{(i)}\right]\bullet\left[\prod_{i\in I\setminus S_x}\bar{q}_x^{(i)}\right]R(S_x)\cdot \tag{2.2}$$

where x is state of abstinence, S_x represents set of indices for units that are stuck-at-*x*, $\bar{q}_x^{(i)}=1-q_x^{(i)}$ in which $q_x^{(i)}$ represents the probability unit *i* fails stuck-at-*x* and $R(S_x)$ is pr ovided in that paper.

However, the computational complexity of eq. (2.2) increases exponentially in the number of units. This makes the method impractical in reality. To simplify this, Nordmann and Pham (1998) made the following two assumptions:

1. Weights are scaled to integers
2. The threshold is described by a rational number

These two assumptions simplify Eq. (2.2) through a recursive approach. Although the computational complexity of the reliability is reduced considerably, it remains a serious problem. Xie and Pham (2005) proposes a simpler method to calculate the WVS reliability similarly through a recursive approach as

$$R = Q_n(0) \cdot \Pr(P=1) + \tilde{Q}_n(0) \cdot \Pr(P=0) \tag{2.3}$$

where $Q_n(0)$ is the probability system output S =1 given the input P=1, and $\tilde{Q}_n(0)$ is the probability system output S =0 given the input P=0.

To improve this further, Xie and Pham (2005) applies saddle point approximation techniques to approximate this reliability function with an accurate large sample approximation formula.

In a series of papers Levitin (2001-2005) evaluates the reliability function based on the universal z-transform (or universal moment generating function, UMGF) technique, which is proven to be a very effective method for numerical implementation. In the application of UMGF in reliability evaluation, each voting unit state k can be characterized by two indices: state probability s_k and the scores this unit contributes to the whole WVS when it votes at state k, $G_k = H_k + \tau A_k$ (H_k is total weight of units supporting state k and A_k is total weight of abstaining units). After defining these two indices, the output of unit j is represented by a polynomial

$$U_I^{\{j\}} = \sum_{k=1}^{K} s_{jk} z^{G_{jk}} \ . \tag{2.4}$$

Weighted voting classifier (WVC) makes a classification decision to choose one ultimate winner among the multiple classes of input by tallying the weighted votes for each decision. The difference between WVC and WVS is that the inputs of WVC have multiple classes while the inputs of WVS have two states (0 and 1). This makes the output of these two systems different. For WVC, the input of each unit belongs to a set θ of K classes, $\theta=\{1\ldots,K\}$. Each unit identifies an object from class k to generate its individual classification decision $d_j(k)$. The unit can also abstain from voting by setting $d_j(k)=0$. The output of each voting unit is incorporated into the system classification decision by its weight in the entire system. This difference between WVS and WVC requires different methods to formulate the reliability problem and to calculate the systems reliability.

Levitin (2002a) suggests a method to calculate the WVC reliability for plurality voting with small object space. The entire WVC output $D(k)$ is calculated as follows:

$$D(k) = \begin{cases} X, & W_k^{\Lambda}(X) > W_k^{\Lambda}(Y) \\ 0, & W_k^{\Lambda}(X) = W_k^{\Lambda}(Y) \end{cases} \tag{2.5}$$

where $W_k^{\Lambda}(X)$ is total weight of units supporting state X.

The reliability of the WVC is defined as:

$$R = \sum_{k=1}^{K} p_k \cdot \Pr\{D(k) = k)\} \tag{2.6}$$

where p_k is the probability that input is in state k.

As each unit has $K+1$ outputs (K input and 1 state representing abstention), the entire WVC consisting of N independent voting units has at most $(K+1)^N$ different states corresponding to different combinations of unit outputs. As some different combinations of unit outputs can result in the same voting weight distribution (VWD), these different outputs are undistinguishable and can be treated as the same.

To take advantage of the above property, Levitin (2002a) develops an *H*-polynomial to calculate the WVS reliability based on the universal moment generating function technique. This *H*-polynomial relates the probabilities of subsystem λ states to VWDs corresponding to these states as

$$H_K^{\{j\}}(z)=\sum_{m=0}^{M} q_{km}^{\{j\}} z^{v_{km}^{\{j\}}} \tag{2.7}$$

where $v_{km}^{\{j\}}$ sum of weights of units belonging to subset λ that respond to input *k* with output *i* at state *m* and $q_{km}^{\{j\}}$ represents its probability.

By sequentially applying operator Ω under certain rules to incorporate all the individual *H*-polynomials, the *H*-polynomial of the entire WVC is obtained. It is shown in Parhami (1994) that threshold voting is considerably simpler than plurality voting. Levitin (2003b) takes threshold voting as its voting strategy, where the final system output is the one which has total support weight exceeding a certain threshold.

2.2Reliability Models of Grid/P2P Systems

2.2.1 Grid Systems

Grid computing is a recently developed technique for complex system with large-scale resource sharing, wide-area program communicating, and multi-institutional organization collaborating etc (Xie *et al.*, 2004). Foster & Kesselman (1998) present the concept of grid and propose a grid development tool addressing issues of security, information discovery, resource management etc. Foster *et al.* (2002) develop grid technologies toward an Open Grid Services Architecture (OGSA) which enables the integration of services and resources across distributed, heterogeneous, dynamic virtual organizations. Huang *et al.* (2004) present an approach to the deployment and re-deployment of grid services based on software architecture models.

The research on grid reliability is also an attractive topic recently. Xie *et al.* (2004) study grid reliability by classifying the research into two areas: reliability of the resource management systems and reliability of the network for communicating or processing, because of their different impacts to the entire grid system in different stage. Dai *et al.* (2002b) propose algorithms to evaluate grid computing reliability by calculating grid system reliability and grid program reliability separately. Limaye *et al.* (2005) propose a solution dealing with fault tolerance at the service level complementing the task-based solutions, and discuss various service availability issues related to the grid, and preliminary results obtained while implementing the smart failover and transparent job-queue replication mechanism and the automated grid installation package. Plank *et al.* (2003) provide a tool Logistical Runtime System (LoRS) that aggregates primitive storage allocations to optimize performance and reliability in grid computing systems. Taufer *et al.* (2005) show that it is possible to classify global computing hosts based on simple metrics such as availability and reliability, and it is efficient to assign tasks to such hosts accordingly. Li *et al.* (2005) apply the signaling game theory to the research on grid resource reliability, and propose a grid resource reliability model based on promise. Levitin & Dai (2005) study the service reliability and performance in grid system with star topology and present a method of universal generating function to evaluate the reliability. Dai and Wang (2005) maximize the grid service reliability by optimally allocating services on grid.

2.2.2 P2P Computing Systems

Peer-to-peer (P2P) systems have recently received immense attentions from both research and industry (see e.g. Leuf, 2002, Foster & Iamnitchi 2003, Steinmetz and Wehrle, 2005, Tian *et al*, 2006, and etc). P2P system is a large-scale distributed system where there is no central server that stores all data. The data are distributed among peers which have the ability to self-organize. P2P systems take advantage of the resources and storages located in the large-scale peers into a large shared-by-all pool of resources. In P2P system, peers cooperate to achieve a desired service, such as: distributed computing (Anderson *et al.*, 2002), file sharing (Saroiu *et al.*, 2002), distributed storage (Rowstron and Druschel, 2001), communication (see e.g. Jabber), and real time media streaming (Hefeeda *et al.*, 2003, Liu *et al.* 2006, and Tu *et al.* 2005). P2P systems are divided into two categories: structured and unstructured, based on the flexibility of placing files at peers. In structured P2P systems, a file is placed at a specific peer, while in unstructured P2P systems a file could be placed at any peer.

Structured P2P systems support hash table lookup/insert techniques, which are usually referred to as distributed hash tables (DHTs). DHTs make the services provided by P2P systems more efficiently: the file lookup/insert and peer join/leave operations take $O\log(N)$ steps, where N is the number of peers (Hefeeda, 2004). Chord, Pastry, Tapestry and CAN are the examples of DHTs.

By using the consistent hashing technique, Chord builds an efficient distributed hash table. The query services of Chord are conducted by locating keys onto nodes, which is a process running on a host. Dabek *et al.* (2001) outline the implementation of a peer-to-peer file sharing system based on Chord, in which, the peer-to-peer systems are able to decide where to compromise and offer better performance, reliability and authenticity. Rieche *et al.* (2004) present a new approach for replication of data in a structured peer-to-peer system to store data persistent using multiple numbers of nodes per interval in a DHT.

CAN (Content-Address Network) is a structured P2P systems which uses a large scale distributed hash table (Hefeeda 2004). Each peer in CAN is responsible for a zone that is dynamically partitioned by CAN from the entire space all the peers compose. Each peer stores the part a part of the distributed hash table that belongs to its region in the space. Ratnasamy *et al.* (2001) address two key problems in the design of CAN: scalable routing and indexing.

Unstructured P2P system locates its files and resources to any independent peer under loose control. The advantages of unstructured P2P systems are the system is more reliable and the queries are more flexible. However, expensive searching process for the desired files among the large-scale distributed peers is the mainly disadvantage of unstructured P2P systems. Yang and Garcia-Molina (2002) propose three efficient search algorithms for unstructured P2P systems. Lin *et al.* (2004) propose a hybrid search algorithm that decides the number of running walkers dynamically with respect to peers' topological information and search time state.

Among many applications of P2P systems, the research on media streaming by P2P has been receiving increasing attention. This system provides the services that users/peers can download simultaneously distributing media sources, which may be truly living or a playback of a recording, and users playback the media while it is being downloading. Xu *et al*. (2002) study two problems in P2P media streaming systems: the assignment of media data to multiple supplying peers and fast capacity amplification of the enter P2P system. Hefeeda & Bhargava (2003) propose a P2P media streaming model that can serve many clients in a cost effective manner and present a P2P streaming protocol used by a participating peer to request a media file from the system. Hefeeda *et al*. (2003) a novel P2P media streaming system PROMISE, encompassing the key functions of peer lookup, peer-based aggregated streaming, and dynamic adaptations to network and peer conditions. In PROMISE, one receiver collecting media stream data from multiple senders. The above literature mainly focuses on the structure of peer-to-peer media streaming systems. Some of the research studies how performance of the entire peer-to-peer system is influenced.

2.3 Software Reliability Models

Software is an important element in computing systems. And more than half of all system failures attribute to faulty software design (Xie et al., 2004); software is not as reliable as hardware, so it is important to evaluate the software reliability in the entire system.

Pukite & Pukite (1998) define software reliability as the probability of failure free operation of a computer program for a specified time in a specified environment.

The definition of software reliability is similar to the definition of hardware reliability, they are both associated with the distribution of time to failure. The research on hardware reliability has strongly influenced software reliability modeling. However, there are many differences between software and hardware systems. Software systems do not wear out with respect to time factor. Another difference is that software will never fail because of the faults that have been removed from the software systems.

Many models for software reliability have been proposed in recent decades. Among them, Markov models and NHPP models are widely used in software reliability analysis. Xie (1991) summarize many well known models of software reliability published from the sixties to 1991.

The following subsection will describe some famous software reliability models in history and introduce some recent research on this topic.

2.3.1 Markov Models

JM model, which is developed by Jelinski and Moranda (1972), is the most known software reliability model which is a Markov process model. This model is based on such following assumptions: at the initial stage, the software is with an unknown but fixed constant number of faults, which is removed immediately without introducing new faults after it being detected, there are no different effects to the failure from the remaining faults in the software, and the intervals between failures are independent, exponentially distributed random variables. The failure rate in JM model is the product of a constant ϕ and the number of remaining faults in the software. This implies that the failure rate is constant between the detection of two consecutive failures, that is, JM model assumes a discrete change of the failure rate at the time of the removal of a fault. This model assumes that all the software faults are of the same size, Xie (1987) presents a general shock model for software failures by assuming that large faults are likely detected early. In literature, many other generalizations of the JM model are proposed (Xie, 1991).

Recently, many other software reliability models based on Markov models have been proposed with different assumptions. Goseva-Popstojanova & Trivedi (2000) consider the phenomena of failure correlation to study its effects on the software reliability measures, and by extending their results, Dai *et al*. (2005) develop a software reliability model based on Markov renewal process for the modeling of the dependence among successive software runs, where more than one type of failure is allowed in general formulation. Rajgopal & Mazumdar (2002) present a method to assess the reliability of a software system by decomposing it into a finite number of modules. From the above literature, Markov models in software reliability modeling have been developed to more and more complex. Software system is more and more considered as a complex system composed of multiple components, each of which has corresponding parameters to estimate and influence the entire software system in different ways.

2.3.2 NHPP Models

Non-homogeneous Poisson Process (NHPP) model, which strongly influences the development of software reliability modeling, is originally presented by Goel and Okumoto (1979), this is a simple model assuming that the cumulative failure process is NHPP with a simple mean value function $m(t) = a(1 - e^{-bt})$ (a>0, b>0). a can be explained as the expected number of faults which are eventually detected. b is interpreted as the failure occurrence rate per fault. The failure rate function can be obtained by $\lambda(t) = \frac{d}{dt} m(t) = abe^{-bt}$. Xie *et al.* (2004) summarize various kind NHPP models for software reliability analysis by extending G-O models with different assumptions. These models include S-shaped NHPP models by Yamada *et al.* (1984), Duane model, Log-power model by Xie & Zhao (1993), and Musa-Okumoto model by Musa and Okumoto (1984).

Recently, many other NHPP models have been studied. Kuo *et al.* (2001) propose a new scheme, which provides an efficient parametric decomposition method for software reliability modeling by considering both testing efforts and fault detection rates, for constructing software reliability growth models. Zhang & Pham (2002) provide methods to predict software failure rates from a user perspective, based on NHPP models. Pham & Zhang (2003) present a model incorporating testing coverage information and compare this model with other existing models. Huang *et al.* (2003) compare several existing software reliability growth models based on NHPP, and propose a more general NHPP model from the quasi arithmetic viewpoint. From the literature above, we can see that the research focus has been moved to reliability prediction and reliability improvement given that NHPP is proposed for estimating software system reliability. The active research problems are parameter estimation, reliability prediction and test coverage.

2.4 Optimization Techniques

Abundant research work has been done on solving system reliability optimization problems since 1970. Kuo and Prasad (2000) categorize the solution methods for the nonlinear programming problems involving integer variables into 3 classes:

1). Exact methods based on dynamic programming, implicit enumeration, and branch-and-bound,

2). Approximate methods based on linear and nonlinear programming techniques,

3). Heuristic methods which yield reasonably good solutions with little computation.

Table 2.1Reference Classification by Optimization Methods

Exact Algorithm	Charles *et al*. (2003), Li *et al*. (2005),Isada *et al*. (2005), Ramirez-Marquez *et al*. (2004), Yalaoui *et al*. (2004), Romera *et al*. (2004), Prasad and Kuo (2000) Cui *et al*. (2004),Agarwal and Renaud (2004), Azaiez and Bier (2007), Bier *et al*. (2005)
Genetic Algorithm	Levitin (2001), Levitin and Lisnianski (2001), Zhao and Liu (2003), Ramirez-Marquez and Coit (2004), Yun and Kim (2004), You and Chen (2005), Marseguerra *et al*. (2004), Levitin (2000), Hsieh (2003), Marseguerra and Zio (2005), Hsieh and Hsieh (2003), Long *et al*. (2007)
Ant Algorithm	Liang and Smith (2004), Zhao *et al*. (2007), Nahas and Nourelfath (2005), Dorigo (2001), Maniezzo and Carbonaro (2001), Liang and Smith (1999)
SA and TS	Wattanapongsakorn and Levitan (2004), Ryoo (2005)
Others	Nourelfath and Dutuit (2004), Ravi *et al*. (2000).

The literatures are categorized in Table 2.1. The review in following part only focuses on genetic algorithm developed after year 2000.

Genetic Algorithm was successfully used in 1990s to solve reliability optimization problems, especially effective for complex combinatorial problems. Gen and Cheng (2000) present detailed applications of GA in system reliability optimization.

Yun and Kim (2004) propose a genetic algorithm to optimize the system reliability of a series system in which redundancy is available at all levels subject to cost, volume and weight constraints. Modular redundancy with identical spare parts is more effective than component redundancy after comparison. Marseguerra and Zio (2005) illustrate the basic concept of genetic algorithm as an optimization tool in RAMS application, highlighting the strength of the approach as well as its limitation.

Much recent research work combines Genetic Algorithm with other methods, such as universal generating function, local search, steepest decent method, and neural network, to improve the solution. Levitin (2000, 2001) and Levitin and Lisnianski (2001) consider redundancy optimization problems for multi-state system, which the total investment-cost is minimized under the required reliability level constraints. To solve the problem, genetic algorithms are used as optimization tools; the universal generating function is used for evaluating the availability of multi-state series-parallel systems. Ramirez-Marquez and Coit (2004) are the first to analyze the MSPS problem with using genetic algorithm. The methodology in that paper is flexible in sense that the practitioner is not limited to a single solution. Zhao and Liu (2003) formulate a stochastic model for redundancy optimization problems of both parallel redundant systems and standby redundant system whose components are connected with each other in a logic configuration. Stochastic simulation, neural network and genetic algorithm are integrated for solving this model.

Genetic Algorithms are also used, combined with local search and steepest decent method, to solve the optimal task allocation and hardware redundancy policies problem for distributed computing system. Hsieh and Hsieh (2003) consider cycle-free distributed computing systems with hardware redundancy. It first obtain the relationship between system cost and the hardware redundancy level for a given task assignment, a hybrid heuristic combining genetic algorithms and the steepest decent method is developed to minimize the system cost. Hsieh (2003) also presents a hybrid genetic algorithm integrated with a local search procedure to solve the optimal task allocation and hardware redundancy policies problems. The local search procedure searches for the locally optimal hardware redundancy levels for a given task allocation, and the Hybrid Genetic Algorithm performs genetic search over the subspace of task allocations.

Compared to other heuristic algorithms, genetic algorithm is more general and easier to handle. By choosing appropriate parameters, genetic algorithm can converge into a good solution in a limited number of simulations. And genetic algorithm can be used by incorporating other methods, such as stochastic simulation, neural networks and etc. The various combinations make genetic algorithm even powerful and attractive. Genetic algorithm is also the most robust heuristic algorithm so far.

CHAPTER 3 WEIGHTED VOTING SYSTEM RELIABILITY

Reliability estimation of the weighted voting systems is a complex problem, which has attracted the attention of many researchers. Nordmann and Pham (1998) first proposed the formula for calculating the reliability of a WVS which is simplified by two given restrictions. However, the computational complexity increases exponentially in the number of units. Xie and Pham (2005) propose a simpler method to calculate the WVS reliability and saddle point approximation techniques are applied to simplify the calculation. In a series of papers Levitin (2001-2005a) evaluates the reliability function based on the universal z-transform (or universal moment generating function, UMGF) technique, which is proven to be a very effective method for numerical implementation of obtaining the reliability of the multi-state weighted voting system (Levitin, 2005b and Levitin *et al*, 2007).

The limitation of the above models is that the inputs of the WVS have very small state spaces. Although in weighted voting classifier (WVC) systems, the input state space is enlarged from {0, 1} in WVS to {1, …, K}, the K states still cannot represent all the states in a real situation. Moreover, with increased input states, the number of different combinations of output increases significantly, increasing considerably computational complexity of the systems reliability. Furthermore, in many practical cases, the state of the input of the voting systems is continuous or approximately continuous and not discrete. For example, to maintain the engines of a plane at a safe level, we need to update the status continuously by measuring the heat and vibration of the working engines to decide whether we need to maintain the engines or not to prevent the possible occurrence of accidents. A monitoring system is installed consisting of a group of parallel sensors to measure the heat and vibration of the engines, which is not directly observable. In this case, a discrete WVS model built for this monitoring system is inappropriate because the heat and vibration states of the engines cannot be accurately simulated by using discrete states.

The rest of this chapter is organized as follows. In section 3.1, we formulate a new model of weighted voting systems that considers continuous inputs. To illustrate this model, some numerical examples are shown. Based on the model formulated, the reliability of the entire voting system is calculated using an analytical method and a Monte Carlo Simulation method. Section 3.2 presents a reliability optimization model under cost constraints and proposes a genetic algorithm to solve this reliability optimization problem. Section 3.3 presents a numerical example to illustrate the reliability optimization problem. Finally, the last section summarizes the work in this chapter.

3.1 Prosposed New Model for Continuous Inputs

The notations for developing the new model of WVS are introduced as follows.

Acronyms

WVS	weighted voting system
WVC	weighted voting classifier
GA	genetic algorithms
MC	Monte Carlo
AM	analytical method
UMGF	universal moment generating function

Notations

N	number of units belonging to WVS
w_i	weight of unit *i*
X	continuous input of the entire system
Y_i	output of individual voting unit *i*
Y	output of entire voting system
$g_x(y)$	probability density function of system output *Y* given the input *X*=*x*
$g_x^i(y_i)$	probability density function of output *Yi* given the input *X*=*x*
$p(x)$	probability that the decision of entire system is correct given *X*=*x*

R	reliability of entire system given any input
$f(x)$	probability density function of input X
a	threshold of the entire system for judging if the output is correct
M	number of different types of voting units
V_{im}	structure of the voting system, V_{im}=1 if unit of type m allocated in position i of the voting system
J_m	number of voting units of type m
C_m	cost of voting units of type m
c_i	cost of voting unit at position i in the entire system
C	cost limit for entire system
σ_m^R	standard deviation of the distribution of the output of voting units of type m in resource R
σ_i^S	standard deviation of the distribution of the output of voting unit at position i in the entire voting system

3.1.1 General Case

In the following, we discuss the reliability analysis of the weighted voting system with N independent weighted voting units w_i and a unanimous input X. The structure of WVS is shown in Figure 3.1. The input of this voting system has continuous states, so it can be assumed as a continuous random variable with probability density function *f(x)* in the range $[x_L, x_H]$. Based on the definition of probability density function, the integration of *f(x)* from x_L to x_H is equal to 1, i.e., $\int_{x_L}^{x_H} f(x)dx = 1$.

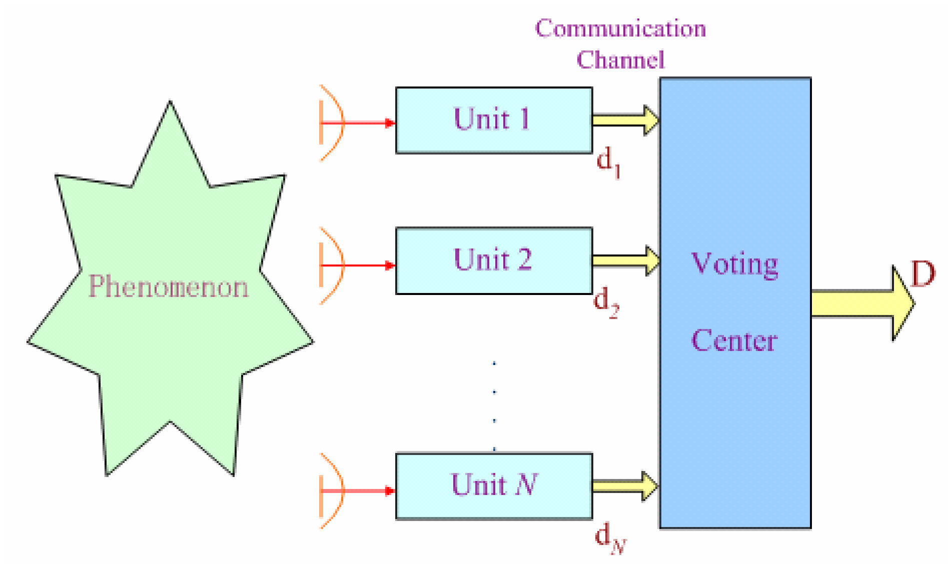

Figure 3.1 Structure of WVS

Given the input X=x, the output of the independent voting unit i is Y_i, whose probability density function is denoted as $g_x^i(y_i)$. The following table describes the configuration of a weighted voting system consisting of N independent voting units with continuous states input.

Table 3.1 Configuration of WVS with *N* Voting Units

	Unit 1	Unit 2	…	Unit N
Weight	w_1	w_2	…	w_N
Output	Y_1	Y_2	…	Y_N
Density	$g_x^1(y_1)$	$g_x^2(y_2)$	…	$g_x^N(y_N)$

The system output comprises of the outputs of individual voting units, weighted by their individual weights. As the output of each voting unit has continuous states, the majority voting algorithm to generate the system output cannot be applied. In order to obtain the output of the entire voting unit system, the weighted average of all the N outputs is calculated by

$$Y = \frac{w_1 \cdot Y_1 + w_2 \cdot Y_2 + ... + w_N \cdot Y_N}{w_1 + w_2 + ... + w_N} \tag{3.1}$$

where Y satisfies the distribution of

$$g_x(y) = f\left(g_x^1(y_1),...,g_x^i(y_i),...,g_x^N(y_N), w_1,...,w_i,...,w_N\right) \tag{3.2}$$

The reliability of the entire system is a function of the weight and accuracy of each individual voting unit. As the standard deviation of the output distribution is a good measure of the accuracy of the individual voting units (a unit with high accuracy will have a small standard deviation), in this chapter, we adopt the standard deviation of the output of each voting unit as a measure of accuracy of that unit. The reliability of the entire system is then determined by the weights and standard deviations of the individual voting units.

The reliability of a discrete state input voting system is defined as the probability that the output is exactly equal to the input, i.e. D(*I*)=*I*. However, this definition is not suitable for the continuous state input case as the probability that the output calculated by incorporating outputs and weights of individual voting units is exactly the same with the unanimous input *X*, is always 0. Hence the reliability for a continuous model needs to be defined separately. Firstly, the definition of 'correct output' is modified to the following:

If the system output Y satisfies $Y \in (x - a, x + a)$*, we say that this output is correct given the input X=x (where a is a constant threshold).*

With this definition of 'correct output', the probability that the output of the entire voting system is correct given a certain input *X*=*x* is

$$p(x) = \Pr\{Y \in (x - a, x + a) \mid X = x\} = \int_{x-a}^{x+a} g_x(y)dy \tag{3.3}$$

The reliability of the voting system can then be defined as the probability that the system makes a correct output for any given input:

$$R = \int_{X_L}^{X_H} p(x) \cdot f(x) dx = \int_{X_L}^{X_H} \int_{x-a}^{x+a} f(x) g_x(y) dy dx \tag{3.4}$$

3.1.2 Solution Algorithm

When the distribution of each individual voting unit is complex, it is difficult to obtain the joint probability function of the output of the system with *N* independent voting units analytically. One feasible method is to obtain the distribution of the entire system output by Monte Carlo simulation. The following algorithm provides a generic Monte Carlo Simulation method to calculate the reliability of the voting system with multiple voting units.

```
Begin
  For j=1 to J do            //J is the total number of the iterations
    For m=1 to M
      Simulate $X \sim F(x)$     // Generate a sample X from the distribution of F(x)
      For n=1 to N do        //Generate the output of all N voting units
        Generate $Y_i(x)$
      End
      $Y \leftarrow f(Y_1, Y_2,,, Y_N)$  //Obtain the output of the entire system from the
                             outputs of all the N independent voting units, given
                             input X;
    Check correctness of the output
```

```
                // if $Y \in (x-a, x+a)$, the output is correct
        End         // a simulation on $X$ following the distribution of $F(x)$
      End       //Now, $J$ sample points of the system reliability are saved in $R$.
                $Summarize(R)$    // the reliability is the number of
                'correct outputs' out of total trial number.
    End
```

With the above algorithm, we can compute the reliability of the entire voting system accurately and efficiently.

3.1.3 Special Cases

In this section, we describe a special case where the probability distribution of entire system output can be derived analytically. We first assume that given the system input $X=x$, the outputs of each individual voting unit i are independent identically normally distributed normal random variables with mean x and standard deviation σ_i, i.e., $Y_i \sim N(x, \sigma_i^2)$. This assumption is reasonable in practice, as firstly the output of each voting unit i is usually symmetrically distributed around the system input x and secondly the voting unit is typically accurate enough that the output is close to the real input with higher probability. This assumption can reduce the computing complexity in obtaining the distribution of the output of entire voting system. The physical meaning of variance $\sigma^2(x)$ of the normal distribution is the overall distance of all data to its input value. This can be used to represent the accuracy of a voting unit. For remainder of this chapter, we use standard deviation of the distribution as a measure of its accuracy.

To simplify the computation further, the random input X can be assumed to have a uniform distribution or some simple continuous distribution. These two assumptions reduce the computational complexity in evaluating the reliability of the entire voting system.

Since the output of the entire system is $Y=\sum_i w_i Y_i / \sum_i w_i$, Y is normally distributed with mean x and variance σ^2, where $\sigma^2 = \sum_i w_i^2 \cdot \sigma_i^2 /(\sum_i w)^2$. Given the input X=x, the probability that output is correct is $p(x) = \Pr\{Y \in (x-a, x+a) \mid X = x\} = \int_{x-a}^{x+a} g_x(y)dy$.

The reliability of entire voting system is then given as $R = \int_{X_L}^{X_H} p(x) \cdot f(x)dx$.

In this special case, the distribution of output of the entire system given the input X=x is normal with mean value x and standard deviation σ . We can transform this distribution into the standard normal distribution $N(0,1)$ by substituting $z=(y-x)/\sigma$ for y. Then

$$p(x) = \int_{-a/\sigma}^{a/\sigma} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz = 2\Phi\left(\frac{a}{\sigma}\right) - 1 \tag{3.5}$$

With this transformation, the probability of correct decision is constant given the threshold a and the accuracy of the voting system, making this voting system insensitive to the input.

3.1.4 Illustrative Example

Here we present a simple example to illustrate the analysis on the reliability of a weighted voting system with continuous state input.

3.1.4.1 Model Description

A temperature detecting system, which is used to measure the temperature of an object, comprises of six independent temperature detecting sensors with different accuracies and weights. The input temperature for all the sensors is the same. It is uniformly distributed between 1 and 2 degrees centigrade. After collecting the input data, the sensors generate their own outputs independently and send them to the processing component in the detecting system to calculate the weighted average of the outputs. For simplicity, we assume that all data transmissions in this system are perfect.

The structure of the temperature detecting system is given in Figure 3.1.

This example involves a voting system consisting of six independent voting units with different weights: w_1=1, w_2=1, w_3=2, w_4=2, w_5=3 and w_6=4. The weighted voting units have different accuracies in measuring the input.

Table 3.2 Weights and Standard Deviation of Individual Voting Units

	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6
Weight (w_i)	1	1	2	2	3	4
Accuracy(σ_i)	$\sigma_1 = 0.05$	$\sigma_2 = 0.02$	$\sigma_3 = 0.02$	$\sigma_4 = 0.01$	$\sigma_5 = 0.04$	$\sigma_6 = 0.01$

We assume that the input temperature *X* satisfies the uniform distribution in the range [1, 2],

$$f(x) = \begin{cases} 1, & if \quad 1 \le x \le 2 \\ 0, & otherwise \end{cases}$$

3.1.4.2 Reliability Analysis of One Voting Unit

First we calculate the probability that the first voting unit generates a correct output. Here we set the threshold a=0.02 degree centigrade, which implies that the output is considered 'correct' if the output Y_1 is between (x-0.02, x+0.02).

Given the input X=x, the output Y_1 is normal with mean x and standard deviation σ=0.05 degree centigrade,

$$g_x^1(y_1) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_1-x)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\cdot 0.05} e^{-\frac{(y_1-x)^2}{2\cdot 0.05^2}} .$$

Then the probability of the voting unit 1 generating a correct output given the input X=x is

$$\begin{aligned} p(x) &= \Pr\{Y_1 \in (x-a, x+a) \mid X = x\} \\ &= \int_{x-a}^{x+a} g_x^1(y_1)dy_1 = \int_{x-0.02}^{x+0.02} \frac{1}{\sqrt{2\pi}\cdot 0.05} \exp\left\{-\frac{(y-x)^2}{2\cdot 0.05^2}\right\} dy_1 \end{aligned} .$$

In this model, input X follows a uniform distribution, so from eq. (3.4), the probability that the output of the first voting unit is correct given any input X is

$$R = \int_{X_L}^{X_H} p(x)\cdot f(x)dx = \int_1^2 \int_{x-0.02}^{x+.02} \frac{1}{\sqrt{2\pi}\cdot 0.05} \exp\left\{-\frac{(y_1-x)^2}{2\cdot 0.05^2}\right\}\cdot f(x)dy_1 dx = 0.3108$$

3.1.4.3 Reliability Analysis of Entire Voting System

The output of unit i, Y_i, given the system input X=x is assumed to follow a normal distribution with mean x and standard deviation σ_i, i.e. $Y_i \sim N(x, \sigma_i)$. Since the output of the entire system is given as

$$Y = \frac{\sum_i w_i \cdot Y_i}{\sum_i w_i} = \frac{Y_1 + Y_2 + 2Y_3 + 2Y_4 + 3Y_5 + 4Y_6}{13},$$

so $Y \sim N(x, \sigma^2)$, where

$$\sigma^2 = \frac{\sum_i w_i^2 \cdot \sigma_i^2}{\left(\sum_i w_i\right)^2} = 1.24 \times 10^{-4}$$

Given the input X=x, the probability that output is correct is given as

$$p(x) = \Pr\{Y \in (x-a, x+a) \mid X = x\}$$
$$= \int_{x-a}^{x+a} g_x(y)dy = \int_{x-0.02}^{x+0.02} \frac{1}{\sqrt{2\pi} \cdot 0.0111} \exp\left\{-\frac{(y-x)^2}{2 \cdot 0.0111^2}\right\} dy \cdot$$

Hence the reliability of the entire voting system is

$$R = \int_{X_L}^{X_H} p(x) \cdot f(x)dx = \int_1^2 \int_{x-0.02}^{x+.02} \frac{1}{\sqrt{2\pi} \cdot 0.0111} e - \frac{(y_1 - x)^2}{2 \cdot 0.0111^2} \cdot f(x)dy_1 dx = 0.9281$$

3.1.4.4 General Monte Carlo Simulation method

For the special normal case, the probability function of the output of the entire system can be obtained analytically. However, for general distributions and more complex voting systems, closed analytical forms may not be obtained. In these cases, the Monte Carlo Simulation method is an efficient and effective alternative to evaluate the reliability of the complex system.

To evaluate the effectiveness and accuracy of this method, we compare it with the analytical method proposed in the previous section. Considering the system in Figure 3.1, all the parameters are kept unchanged, and if the output of the entire system is between (x-a, x+a) (a=0.02), the output is considered to be correct.

As we know, the accuracy of Monte Carlo Simulation method is greatly influenced by the sizes of samples we use to simulate. To analyze the accuracy of Monte Carlo Simulation method with the analytical method, the simulations based on five samples of different sizes are taken to obtain the reliability of the WVS presented in Figure 3.1. Reliability here is calculated as the proportion of correct outputs out of the total number of outputs. The program runtime (in seconds) is recorded and the error of the Monte Carlo Simulation method is compared to the result from analytical method in the following table. The error is defined as:

$$error = \frac{|\,reliability\ estimated\ by\ MC - reliability\ estimated\ by\ AM\,|}{reliability\ estimated\ by\ AM} \times 100\% \quad (3.6)$$

Table 3.3 Comparison of reliability estimates for different sample sizes

Sample size	10	100	**1000**	10000	100000
Reliability	0.9000	0.9100	**0.9210**	0.9303	0.9283
Error	3.03%	1.95%	**0.77%**	0.24%	0.0216%
Runtime (*s*)	0.0160	0.0470	**0.2660**	2.3590	23.3280

From this table, the simulation with 1000 random samples is sufficient to estimate accurately (*error*<1%) the reliability of the WVS by the Monte Carlo Simulation method. Correspondingly, the program runtime is 0.2660s, indicating high efficiency of the Monte Carlo Simulation method.

3.1.4.5 Voting System with Different Numbers of Voting Units

To further study the effect of the number of voting units on the reliability of the entire system, the number of independent voting units is varied and the reliability is recalculated. Table 3.4 shows some numerical results of the reliability which increase in the number of voting units.

Table 3.4 Voting systems with different numbers of voting units

	1 unit	2 units	6 units
Threshold	0.02	0.02	0.02
Reliability	0.3108	0.7372	0.9281

3.2 Reliability Optimization with Cost Constraints

3.2.1 Optimization Model Formulation

Different strategies of allocating independent voting units to the voting system can result in different reliabilities of the entire voting system. To increase the reliability of the entire system, we have two alternatives. One is to increase the system reliability by choosing the voting components with high accuracy. The second is to enhance the reliability of entire system by optimizing the structure of the system.

However, when given the limitation of redundancy and accuracy of individual voting units available, the system reliability can only be enhanced by optimizing the structure of the system. Consider a weighted voting system consisting of *N* independent voting units chosen from *M* types of voting units with distinctive accuracies and costs. For each type *m* the maximum number of the units that can be used is J_m. These J_m units have the unique costs and accuracies, denoted as C_m and σ_m^R respectively. Thus, the total number of voting units that can be chosen is $\sum_m^M J_m$. The reliability function of the entire system is only determined by the weights w_i and standard deviation σ_m^S of the individual voting unit *i*:

$$R_{sys} = f\left(w_1, w_2, ..., w_N; \sigma_1^S, \sigma_2^S, ..., \sigma_N^S\right) \tag{3.7}$$

To reduce the computational complexity of estimating the reliability of the WVS in the optimization problem in this section, the normality assumption on the output distribution of voting units is applied, that is, we assume that the output distribution of voting unit *i* follows normal distribution $Y_i \sim N(x, \sigma_i)$, where *x* is given as the input value and σ_i is the accuracy of the voting unit *i*.

We define a variable V_{im} to represent the structure of the entire system:

$$V_{im} = \begin{cases} 1, & if \ voting \ unit \ i \ is \ of \ type \ m \\ 0 & if \ voting \ unit \ i \ is \ not \ of \ type \ m \end{cases}.$$

After obtaining V_{im} of the entire system, the total cost can be calculated by

$$C_{tot} = \sum_{i=1}^{N} \sum_{m=1}^{M} V_{im} \cdot C_m \tag{3.8}$$

The optimal voting unit allocation problem can then be formulated as below:

$$\begin{aligned} &Max: && R_{sys} = f\left(w_1, w_2, \ldots, w_N; \sigma_1^S, \sigma_2^S, \ldots, \sigma_N^S\right) \\ &Subject\ to: && C_{tot} = \sum_{i=1}^{N} \sum_{m=1}^{M} V_{im} \cdot C_m \le C \\ & && \sum_{i=1}^{N} V_{im} \le J_m \quad for\ \forall m \end{aligned} \tag{3.9}$$

Transforming the original problem into a cost minimizing problem under a given system reliability requirement, the problem can be reformulated as

$$\begin{aligned} &Min: && C_{tot} = \sum_{i=1}^{N} \sum_{m=1}^{M} V_{im} \cdot C_m \\ &Subject\ to: && R_{sys} = f\left(w_1, w_2, \ldots, w_N; \sigma_1^S, \sigma_2^S, \ldots, \sigma_N^S\right) \ge R_{rqm} \\ & && \sum_{i=1}^{N} V_{im} \le J_m \quad for\ \forall m \end{aligned} \tag{3.10}$$

Following figure depict the optimization problem for weighted voting system.

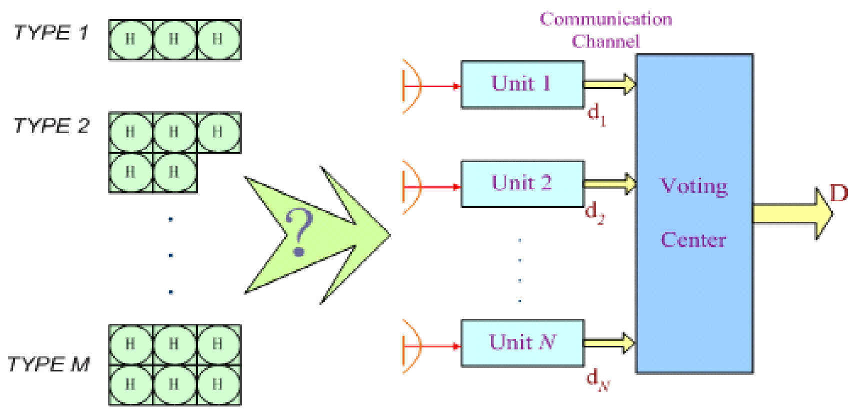

Figure 3.2 Resource allocation problem for weighted voting system

3.2.2 Optimization Technique

To solve the above optimization problem for the optimal structure of the voting system, we propose to use a Genetic Algorithm (GA). The Genetic Algorithm was first published by John Holland (1975), and has since been widely used to solve optimizations problems in industrial engineering. Gen and Cheng (2000) summarize the applications of the GA in engineering optimization. Painton and Campbell (1995) use genetic algorithm as the optimization method to maximize the performance of personal computer system subject to cost constraints. Coit and Smith (1996) present a penalty guided genetic algorithm to identify a feasible best solution by searching over feasible and infeasible region effectively and efficiently. Lisnianski and Levitin (2003) optimize the reliability of the multi-state system (MSS) by the GA.

In general, a GA has four basic components, as summarized by Michalewicz (1992). We describe in detail the application of the GA to solve the above voting unit allocation problem.

3.2.2.1 Chromosome Representation

The representation of the entire system structure is by a *2N*-length integer string where the value b_i in *i*th position corresponds to the type of voting unit allocated to unit *i*, and the value w_i in position ($N+i$) of the integer string is the weight assigned to unit *i*. For example, the integer string [b_1 b_2 ...b_i... $b_{N,}$, w_1, w_2,..., w_i,..., $w_{N,}$] corresponds to type b_i voting unit allocated to position *i* with weight w_i. b_i ranges between (1,*M*).

3.2.2.2 Initial Population

The *2N*-length integer string composes of two parts; the first part represents the type of voting units allocated to the corresponding position and the second part indicates the weights assigned to each unit. The two parts have different representations and different ranges, and the way of generating the initial values are different as well. The first part is obtained by the following algorithm:

Step 1: Start from i=1. Let L_m be the number of left-over voting units of type m. At the initial stage, the value of L_m is J_m.

Step 2: Determine if L_m is 0 or positive. If L_m is 0, remove mth type of voting units from the selection list. For the ith position at the chromosome, randomly select one value from the selection list, say m. Put m into b_i. Go to Step 3.

Step 3: L_m is then set as L_m=J_m-1, and i=i+1.

Step 4: Repeat step 2 until i=N;

Step 5: Stop the generation program if all the initial population have been generated, otherwise repeat step 1.

The initial weight allocations of the N voting units are generated with random integers in the range (1,100). As multiplying all the unit weights by a constant does not change the weight allocation in the entire system, the unit weights can be normalized so that the total weights of all units is set at a constant W'. After normalization, the weight of unit i is given as

$$w_i' = \frac{w_i \cdot W'}{\sum_{i=1}^{N} w_i} \tag{3.11}$$

With this normalized form, it is easier to compare the weights of the voting units.

3.2.2.3 Fitness of a Chromosome

After the solution is decoded, the fitness values are estimated. They are the values of the objective function which measures the quality of the solutions obtained. These fitness values can be used to compare the different solutions. The fitness values of the chromosomes are determined by two types of variables, the weight w_i allocated at the ith position of the chromosome, and the standard deviation σ_m^S.

$$f=f(w_1, w_2, \dots, w_N;\ \sigma_1^S, \sigma_2^S, \dots \sigma_N^S) \tag{3.12}$$

Calculating fitness values of the chromosomes is the most time-consuming part in the GA even when the exact allocation strategy of the voting units in resource for each position in the WVS is given. In section 3.1, two methods (analytical method and Monte Carlo Simulation method) are proposed to estimate accurately the reliability of WVS. However, these two methods are inefficient to estimate fitness of the chromosome in a genetic algorithm because of the large number of chromosomes to be estimated.

With the normality assumption, the fitness value of a chromosome is only determined by the standard deviation of the distribution of the output of the entire system, which is a function of the standard deviation and weight of individual voting units. Based on this, in our genetic algorithm, we develop an efficient method based on a fitness-standard deviation index table to evaluate fitness of the chromosomes according to eq. (3.5). After obtaining the standard deviation of the entire system, we check this index table to calculate fitness of this chromosome.

To sum up, the procedure of estimating fitness of a chromosome $f=f(w_1, w_2, \dots, w_N;\ \sigma_1^S, \sigma_2^S, \dots \sigma_N^S)$ based on index table is described below:

Step 1: according to eq. (3.4) and (3.5), calculate an index table where the entry at position *ind* is assigned value $2\Phi(ind)-1$. The interval between two continuous indices is defined to be Δ;

Step 2: for each chromosome, given weight and accuracy allocated to each voting unit, find the accuracy σ of the entire WVS, which is the standard deviation of the output of the entire system as well;

Step 3: with the preset threshold a and σ, calculate the fitness index of current chromosome: $ind = \frac{a}{\sigma \cdot \Delta}$ and find the entry value at this index according to Step 1;

Step 4: according to eq. (3.4), calculate the system reliability by incorporating the given input distribution and the entry value obtained in Step 3.

In the example in Section 3.1.4, the threshold for the voting system is a=0.02 and the standard deviation of the output of that entire system is $\sigma = 0.0111$, which is easily calculated from the corresponding chromosome. The index of search for the fitness value of this chromosome is 360 (the integer part of $\frac{a}{\sigma \cdot \Delta} = \frac{0.02}{0.0111 \cdot \Delta}$, where Δ is the interval of the index table and Δ is assumed to be 0.005 in this chapter). Checking the 360th item in this index table, the fitness value of 0.9281 is obtained according to eq. (3.4). We find that the result 0.9281 is exactly the same with the reliability estimated by analytical method. This is because the interval Δ is sufficiently small; the estimation error caused by this index table method is negligible. The computational complexity of this method is $O(\frac{1}{\Delta})$, where Δ is a preset constant. We conclude that the reliability estimate method based index table is more accurate and efficient than Monte Carlo Simulation method.

To account for the voting unit costs constraints, the penalty function is incorporated into the fitness function to transform the constrained problem into an unconstrained one. This fitness function eq. (3.12) is then given as

$$f = f\left(w_1, w_2, \ldots, w_N; \sigma_1^S, \sigma_2^S, \ldots, \sigma_N^S\right) - \sum_{m=1}^{M} \pi_m \eta_m - \pi_c \eta_c \tag{3.13}$$

where π_m is the penalty coefficient of constraint *m*, and η_m is the penalty related to the constraint *m*. The penalty coefficient π_m is chosen in such a way that the smallest value of the fitness function that meets all the constraints is greater than the solution with the largest value of fitness function but violating at least one constraint. The penalty η_m is proportional to the extent of the constraint violation. π_c is the penalty coefficient of the total cost constraint and η_c is the penalty to the fitness function when the cost constraint is violated.

3.2.2.4 Selection

The selection procedure provides the evolutionary force in the GA. Two important issues determine the force of selection: population diversity and selective pressure. With strong selective pressure, the genetic search will terminate prematurely and converge to a local optimum. With too little selection force, the evolutionary progress will be slow. A good selection method is crucial to the quality of solution and the speed of the evolutionary process. In the past few years, many selection methods have been applied into the optimization problems: Roulette wheel selection, $(\mu + \lambda)$ - selection, Tournament selection, Steady-state reproduction, Ranking and scaling and Sharing.

For the voting unit allocation problem above, we use Roulette wheel selection proposed by Holland (1975), which is the best known selection method. The basic idea of Roulette wheel selection method is to determine selection probability for each chromosome proportional to the fitness value. The chromosome with greater fitness value will be selected with higher probability.

3.2.2.5 Crossover

The crossover procedures generate the offspring solution by swapping parts of genes from two selected parent chromosomes. The offspring will inherit some useful properties from the parent chromosomes. Lisnianski and Levitin (2003) consider three crossover procedures for the assignment problems: single point crossover, two-point or fragment crossover and uniform crossover. In our problem, we generate the offspring by uniform crossover, in which each element is copied from either parent with equal probability.

3.2.2.6 Mutation

To avoid the premature convergence into a local optimum, the mutation operator is introduced to modify slightly some of the string elements of the offspring solution. The commonly used mutation procedure changes the value of a randomly selected string element by 1.

3.2.2.7 Parameters in GA

The GA parameters such as population size, maximum generation, crossover ratio and mutation play an important a role, hence choosing good parameters is crucial. As GA is a dynamic and adaptive process, some parameters are modified during the run of the algorithm. Thierens (2002) shows that varying the mutation ratio is preferable to fixed mutation rate. Gen and Cheng (2000) provide three kinds of rules to modify the GA parameters: deterministic, adaptive and self-adaptive. In the genetic algorithm developed in this chapter, we choose the deterministic rules to adapt some of the strategy parameters during the execution time due to its simplicity and effectiveness. For example, the mutation ratio is decreased gradually over the generations: $p_m = 0.1 - 0.03\frac{t}{G}$, where t is the current generation number and G is the maximum generation. However, no significant improvement on the final optimization solution has been found in our experiments due to the deterministic adaptation of mutation rate.

3.3Numerical example

3.3.1 Optimization Problem

In this section we present a numerical example to illustrate the solution of the reliability optimization problem for the temperature detection system discussed in Section 3.1.4. Four different types of sensors are available to assemble the detecting system and the output of each sensor follows a normal distribution with the mean value as the input temperature of the system. The configuration of each voting unit resource is presented in Table 3.5

Table 3.5 Parameters of the voting units

	Type 1	Type 2	Type 3	Type 4
Total number	4	3	3	2
Standard deviation	$\sigma_1^R = 0.05$	$\sigma_2^R = 0.04$	$\sigma_3^R = 0.02$	$\sigma_4^R = 0.01$
Cost	2	4	8	12

Considering a total cost constraint of $C \leq 30$, problem (3.9) is solved to determine the allocation strategy of the sensors to optimize the reliability of entire system. In this example, the threshold for determining the correct decision is set at *a*=0.02. Other parameters of the GA are: no. of generations is 1000, population size is

50, penalty coefficient for cost constraint, that is π_c, is 0.05, penalty coefficient for unit type constraint π_m=0.1 for m=1,…, M, and the probability of crossover is 0.3. These parameters are fixed during the run of our program. The deterministic adaptation is only applied to alter the probability of mutation over generations: $p_m = 0.1 - 0.03\frac{t}{G}$. The runtime of the GA coded in Matlab is about 30 seconds on a Pentium IV computer.

3.3.1 The Best Solutions from GA

The best feasible solution obtained by the GA is presented in Table 3.6. The reliability of the entire voting system allocated in this strategy is R=0.9602. The overall performance of the GA is measured by conducting 30 independent experiments where the mean value of the system reliability from these experiments is $\overline{R} = 0.9435$, and the standard deviation of the results is 0.0543. This result indicates that the Genetic Algorithm in this chapter converges to a very good solution based on a sufficient number of trials and the result is statistically sound.

Table 3.6 The best voting system configuration obtained by the GA

	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6
Type	Type 1	Type 3	Type 4	Type 2	Type 1	Type 1
Weight	0.5512	2.1029	4.8353	1.3176	0.6762	0.5168

To validate the results obtained by genetic algorithm is a near optimal solution, the experiment by changing the weights and type of voting unit in any location of the weighted voting system could be applied. In the validation experiment, reliability will be measured and compared to the ‘near optimal solution’ suggested in table 3.6. We will find in most cases (in this example, the probability is 99% as 0.9602 is at 99% confidence level for the 30 independent experiments if the assumption of normality is valid), system performance will be deteriorated that the reliability is less than 0.9602. In this way, the reliability model and genetic algorithm can be simultaneously validated.

3.3.2 Sensitivity Analysis on the Total Cost Limit

As the allocation of voting units is constrained by the total cost in this example, different total cost constraints will result in different allocation of the voting units. This in turn affects the reliability of entire system. Figure 3.3 illustrates how the total cost constraints affect the reliability of the best allocation strategy obtained from the GA. It can be seen from Figure 3.3 that the reliability increases with increase of the allowed total cost. This is expected as the more voting units allocated to the voting system, the higher its reliability.

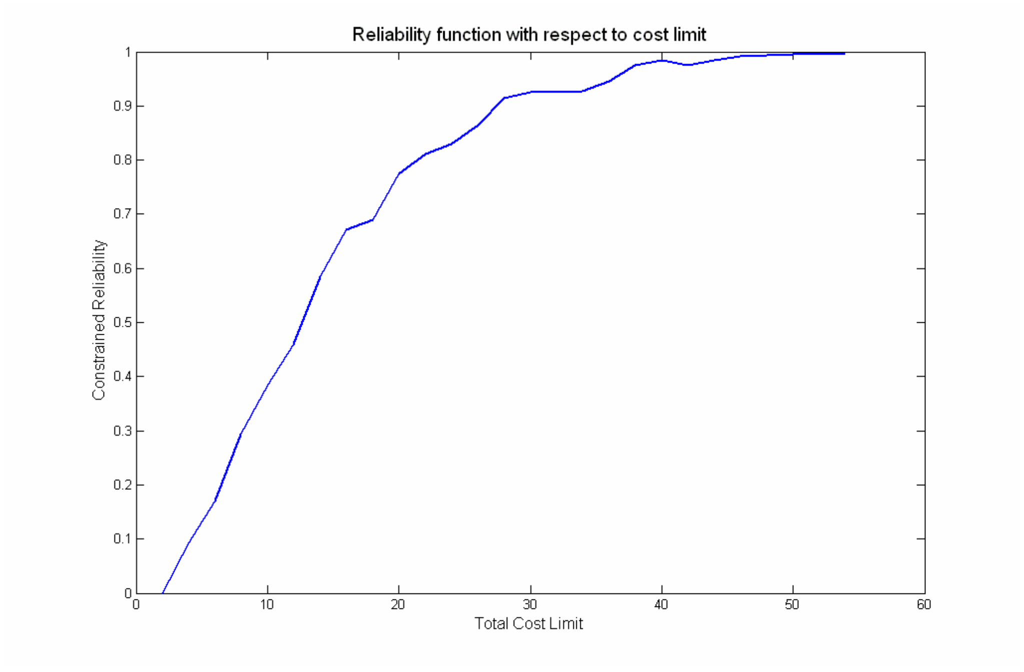

Figure 3.3 Reliability function with different cost limit

3.4Summary

Weighted voting systems and weighted voting classifiers are widely used in human organization systems, pattern recognition and other technical fields. One drawback of the existing models in the literature is that the inputs in these models are all assumed to be discrete. In practice, the input can be continuous. The model proposed in this chapter is formulated by taking into account the possibility of continuous inputs. The definitions of 'correct decision' and 'reliability of entire system' are redefined correspondingly for the case of continuous inputs. The model is evaluated analytically by making some simplifying assumptions. The distribution of the output of each voting unit is assumed normal with the mean value coinciding with the input. The output of entire system is then a weighted sum of the outputs of the units composing the system.

Alternatively, a Monte Carlo Simulation method can be applied if an analytical solution to the model is not available.

A reliability optimization problem with cost constraints is then formulated. As different types of voting units can have different accuracies and costs, the different allocations of these voting units make the reliability of the entire voting system different. In addition, we also provided a detailed description of the GA adapted to solve the optimization problem and illustrated its application with a numerical example.

CHAPTER 4 FURTHER ANALYSIS ON WVS RELIABILITY

Weighted voting systems and weighted voting classifiers are widely used in human organization systems, pattern recognition and other technical fields. One drawback of the existing models in the literature is that the inputs in these models are all assumed to be discrete. In practice, the input can be continuous.

In last chapter, we discussed a new reliability model of WVS by taking into account continuous inputs, such as measures of temperature and pressure. In that model, system output is assumed to follow normal distribution with mean value being input and standard deviation as accuracy of the unit. This is a perfect situation that system output is unbiased and reflects exactly how system input distributes. However, in practice, this underlying assumption is fraught with problems because practical voting systems can never perform in such a perfect and idealistica manner. Practically, the mean value of the output of the voting units are biased to the input value, and the accuracy (represented by standard deviation of the distribution of the unit output) may also depend on the inputs. Hence the reliability of the entire voting system depends not only on the accuracy but also on the bias of each unit.

This chapter extends the models built in chapter 3 by considering the continuous state input weighted voting systems with biased output. In this chapter, we will discuss three cases by relaxing the assumptions of the weighted voting systems. Each of the assumptions represents one specified application background. In the first case, the output of each voting unit is unbiased to the unanimous input but the accuracy of this voting unit is assumed to be dependent on the input. For the same voting unit, its voting accuracy varies considerably due to the difference of input object. In the second case, we consider a common used biased voting system, of which the mean value of the output is biased to the input due to the irremovable defects in designing the process and calibration process. The last case discusses a weighted voting system with time dependent accuracy. For better understanding, three corresponding numerical examples are presented to illustrate how to calculate the reliability of the weighted voting systems under the given assumptions. Both the Monte Carlo Simulation method and analytical method are applied to these examples, and a comparison between the two methods is made at the end of each example.

The remainder of this chapter is organized as follows. Section 4.1 discusses an unbiased voting system. In section 4.2, a biased voting system is considered and in section 4.3 we study the effect of time dependent accuracy. Section 4.4 compares

Monte Carlo Simulation method and analytical method in calculating system reliability. Finally, section 4.5 summarizes this chapter.

4.1 Unbiased Voting System

4.1.1 The Model

In this case, the output of each voting unit is unbiased to the unanimous input and the accuracy of this voting unit is assumed to be dependent on the input. For the same voting unit, its voting accuracy varies considerably due to the difference of input object.

The output of each voting unit follows certain distributions, given the unanimous input. Another important assumption is the unbiased property of the assumed distribution: the mean value of the output distribution is unbiased to the given input. This assumption is mathematically represented by $\mu_i(x)=x$, given the system input x. Under these two assumptions, the output Y of the entire voting systems is normally distributed with mean value x, and variance $\sigma^2(x)=\sum_i w_i^2 \cdot \sigma_i^2(x)/(\sum_i w)^2$, where $\sigma_i(x)$ can be polynomial function which will be discussed in details in later subsections. So $g_x^i(y_i)$ ~$N(x,\ \sigma_i^2(x)$) and the distribution of the system output $g_x(y)$~$N(x, \sigma^2(x)$). In terms of the definition of reliability of weighted voting systems, given the unanimous input X=x, the probability that the output of the entire voting system is considered as correct is

$$p(x) = \Pr\{Y \in (x-a, x+a) \mid X = x\} = \int_{x-a}^{x+a} g_x(y)dy \tag{4.1}$$

and the reliability of the system is

$$R = \int_{X_L}^{X_H} p(x) \cdot f(x)dx = \int_{X_L}^{X_H} \int_{x-a}^{x+a} f(x) g_x(y) dxdy \tag{4.2}$$

4.1.2 Numerical Example

Suppose we use a temperature detecting system to measure the temperature of an object comprising of 4 independent temperature detecting sensors with different accuracies and weights. The input temperature for all the sensors is unanimous, and is assumed uniformly distributed between 100 and 200 degrees centigrade. The threshold *a* is preset at 2, so the output which is out of the range (*x*-2, *x*+2) is considered as wrong output given the input *X*=*x*. After collecting the input data, the sensors generate their own outputs independently and send them to the processing component in the detecting system to calculate the weighted average of the outputs. For simplicity, we assume that all data transmissions in this system are perfect. The following table shows the parameters assigned in the weighted voting systems.

Table 4.1 Parameters in the weighed voting systems

	Unit 1	Unit 2	Unit 3	Unit 4
Weights	*1*	*3*	*2*	*4*
Bias	*0*	*0*	*0*	*0*
$\sigma(x)$	*0.01x*	*0.01x+1*	$10^{-4}x^2$	*5-0.02x*

$$\sigma^2(x)=\frac{1\times(0.01x)^2+9\times(0.01x+1)^2+4\times10^{-4}x^4+16\times(5-0.02x)^2}{(1+3+2+4)^2}$$
$$=\frac{4\times(0.01x)^4+74\times(0.01x)^2-302\times(0.01x)+409}{100}$$

So, given the input *X*=*x*, the probability that the systems generates correct output is

$$p(x)=\Pr\{Y\in(x-a,x+a)\mid X=x\}$$
$$=\int_{x-a}^{x+a}g_x(y)dy=\int_{x-2}^{x+2}\frac{1}{\sqrt{2\pi}\sigma(x)}e^{-\frac{(y-x)^2}{2\sigma(x)^2}}dy=2\Phi\left(\frac{2}{\sigma(x)}\right)-1$$

As the input is assume as uniformly distributed in [100,200], the reliability of the entire voting system can be obtained given any input:

$$R=\int_{X_L}^{X_H}p(x)\cdot f(x)dx=\int_{100}^{200}\frac{1}{100}\times\left[2\Phi\left(\frac{2}{\sigma(x)}\right)-1\right]dx$$

Since the above expression of reliability is hard to calculate, Matlab was used to numerically solve it with the threshold *a*=2, producing a result *R*=0.8834.

At the same time, we apply Monte Carlo Simulation method to simulate this weighted voting system, and the reliability is calculated as *R*=0.8933. The difference between these two methods is 0.00992, that is 1.11% error in percentage.

4.2 Biased Voting Systems

4.2.1 The Model

This subsection considers a biased voting system which is common in practice. For some voting systems, the mean value of the output is biased to the input due to the irremovable defects in designing the process and calibration process. The bias denoted by $\Delta_i(x)$ can be a constant Δ_i for voting unit *i*. This means that the bias property is an inherent property and is independent of system input. The mean value of the output distribution of voting unit *i* is then $\mu_i(x)=x+\Delta_i(x)$, with variance $\sigma_i^2(x)$. The variance could also be described as polynomial function which is illustrated in the following example. With the output of the entire voting system given as $Y=\dfrac{w_1\cdot Y_1+w_2\cdot Y_2+...+w_N\cdot Y_N}{w_1+w_2+...+w_N}$,

Y then follows a normal distribution:

$$g_x(y)\sim N\left(x+\frac{\sum_i w_i\cdot\Delta_i(x)}{(\sum_i w_i)},\frac{\sum_i w_i^2\cdot\sigma_i^2(x)}{(\sum_i w)^2}\right) \tag{4.3}$$

4.2.2 Numerical Example

In this example, we consider a weighted voting system with biased output. The parameters of this system are shown in Table 4.2. We assume the same input distribution as the first example, and the threshold is preset at 2.

So,
$$\mu(x)=\frac{1\times(2+0.01x)+3(-3+0.02x)+2\times5-4\times0.01x}{10}+x$$
$$=1.003\,x+0.3$$

Table 4.2 Parameters in the weighed voting system

	Unit 1	Unit 2	Unit 3	Unit 4
Weights	*1*	*3*	*2*	*4*
Bias	*2+0.01x*	*-3+0.02x*	*5*	*-0.01x*
$\sigma(x)$	*0.01x*	*0.01x+1*	$10^{-4}x^2$	*5-0.02x*

$$\sigma^2(x)=\frac{4\times(0.01x)^4+74\times(0.01x)^2-302\times(0.01x)+409}{100}$$

$$p(x)=\int_{x-2}^{x+2}\frac{1}{\sqrt{2\pi}\sigma(x)}e^{-\frac{(y-\mu(x))^2}{2\sigma(x)^2}}dy$$

$$=\Phi\left(\frac{-0.003x+1.4}{\sigma(x)}\right)+\Phi\left(\frac{2.6+0.003x}{\sigma(x)}\right)-1$$

The reliability of the entire system is *R*=0.8195 by analytical method of incorporating equation (3.4) and (4.3) and R=0.8290 by the Monte Carlo Simulation method. The difference is 1.14%.

Comparing these results to the reliability in the first example, given the same accuracy, we infer that the reliability of this voting system is worse than that in the first case. The bias factor influences the performance of the voting systems and should be accounted for.

4.3 Time Dependent Accuracy

4.3.1 The Model

The above two cases do not consider the time factor in the reliability analysis of weighted voting systems. The accuracy of each voting unit is assumed independent of time and is considered a constant. However, weighed voting systems which are similar to other hardware systems, can wear out and their performance and accuracy will decrease over time. This can happen in some extreme situations where maintenance service becomes cost-prohibitive or in many cases is simply not feasible. For example, in outer space missions, many exploration robots and devices are launched into outer space where humans cannot be sent. These robots and devices are often hit by asteroids and are exposed to various cosmic rays which in the long run can cause the voting systems in the exploration robots to deteriorate and become inaccurate; maintenance of these devices which is mainly controlled from the ground cannot solve this kind of problems all the time. The accuracy of these voting systems deteriorates as the necessary maintenance cannot be conducted in time, making it a function of time. The distribution of system output which follows a normal distribution is then dependent on time *t*,

$$g_{(t,x)}(y) \sim N\left(\mu(t,x), \sigma^2(t,x)\right) \tag{4.4}$$

The reliability of this voting system is then also a function of time *t*.

4.3.2 Numerical Example

In this example, the accuracy and bias of the weighted voting system depend not only on the input but also on the time elapsed. Due to the time factor, the performance and reliability of the weighted voting system deteriorates. We assume the same input distribution as the first example, and the threshold is preset at 2. Table 4.3 gives the information of this system.

Given the information of the weighted voting system, the reliability function is obtained by both Monte Carlo Simulation method and analytical method of incorporating equation (3.4) and (4.4), which is shown in the following graph. Figure 4.2 and Figure 4.3 show the differences of reliability estimation between the two methods. From the graph, we observe that the differences are all less than 0.016 for all the time points.

Table 4.3 Parameters in the weighed voting system

Weights		Bias	$\sigma(x)$
Unit 1	1	$\left(1.2-0.4e^{-\frac{t}{100}}\right)(2+0.01x)$	$0.01x\left(1.3-0.3e^{-\frac{t}{100}}\right)$
Unit 2	3	$\left(1.1-0.2e^{-\frac{t}{200}}\right)(-3+0.02x)$	$\left(1.1-0.2e^{-\frac{t}{150}}\right)(0.01x+1)$
Unit 3	2	$5\times\left(1.2-0.3e^{-\frac{t}{100}}\right)$	$10^{-4}x^2\left(1.2-0.4e^{-\frac{t}{200}}\right)$
Unit 4	4	$-0.01x\left(1.3-0.3e^{-\frac{t}{200}}\right)$	$\left(1.1-0.3e^{-\frac{t}{200}}\right)(5-0.02x)$

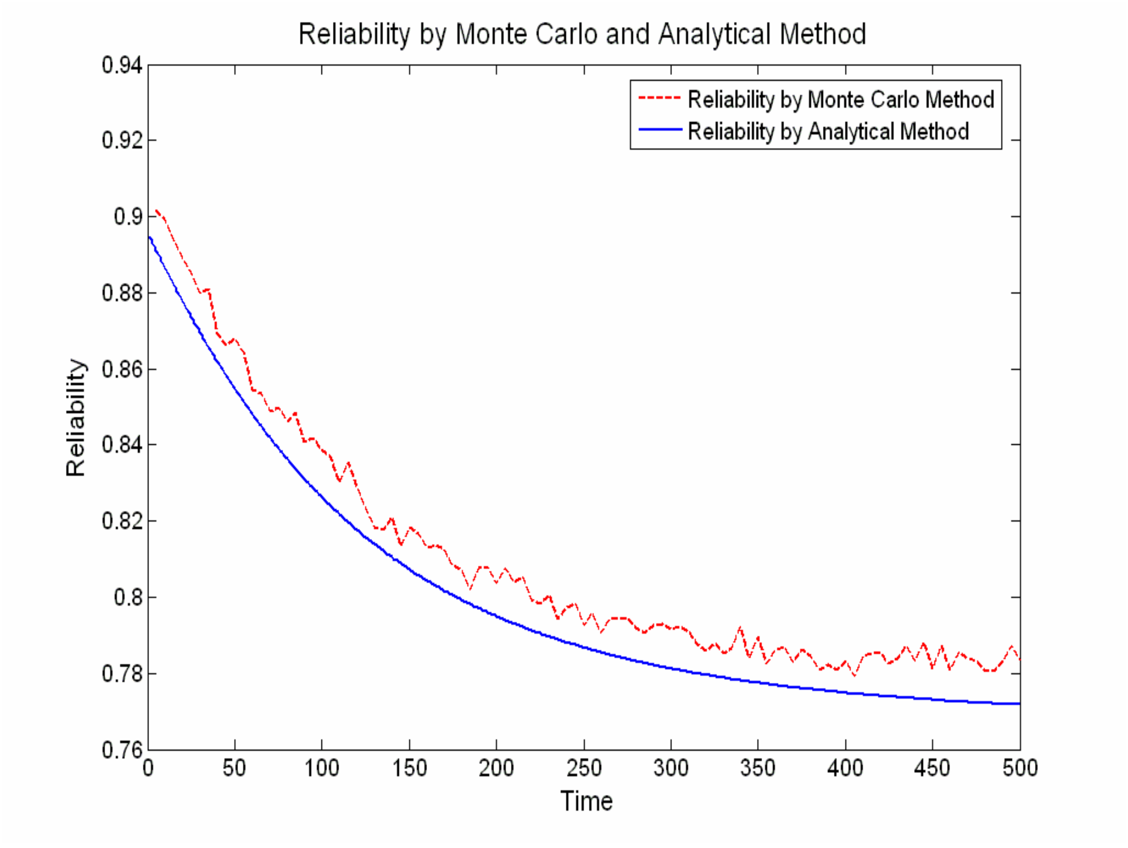

Figure 4.1 Reliability by Monte Carlo and analytical method

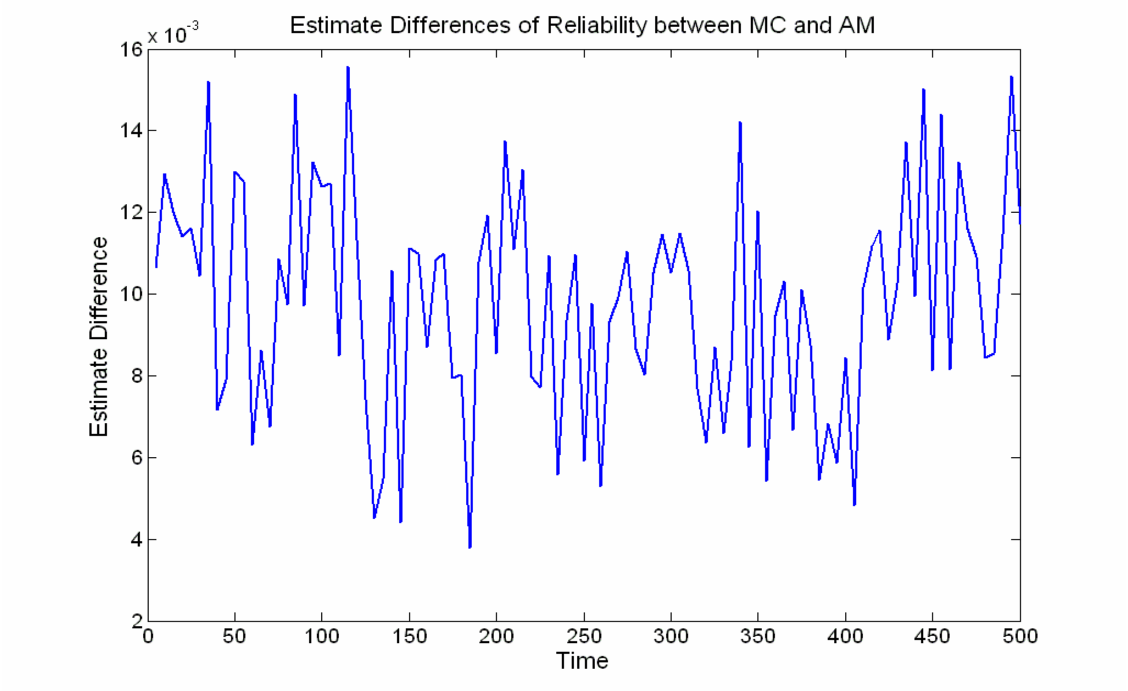

Figure 4.2 Differences of reliability estimation of the two methods

4.4 Comparison between Monte Carlo and Analytical Method

Both Monte Carlo Simulation method and analytical method are applied to calculate the reliability of the weighted voting systems in the three cases above. It shows that the reliability estimations by Monte Carlo Simulation method are very close to the results obtained by analytical method. This justifies that both methods are applicable for the reliability analysis of the weighted voting system with similar quality of approximation.

However, the two methods have their own advantages, so selecting one of them should depend on the corresponding conditions and specific requirements. In general, the Monte Carlo simulation is broadly suitable for most tools (such as Markov model, Fault tree, Petri Net, Block diagram, Network diagram etc) in evaluating system reliability; while the analytic method is only applicable for the model we formulate above. And analytic method only applies for certain distributions. On the other hand, the analytic method is more effective and less time-consuming than the Monte Carlo simulation. It is because usually the simulation method is repeatedly run for many times, which is computationally expensive, especially for some complicated systems; while the analytic method can directly obtain the results from the formula by one-step substitution and computation.

4.5 Summary

In this chapter, we have proposed reliability models for weighted voting systems with continuous state input, in which the output of voting units are considered to be biased to the input and the accuracy of the units are assumed to depend on the input. Three different cases of the weighted voting systems, accounting for different assumptions and application backgrounds, were discussed. To illustrate the three cases, three numerical examples were conducted respectively. Reliability of the weighted voting system was calculated both by Monte Carlo and by analytical method for each example. Comparing the first two cases, we find that the reliability of the biased voting system is lower than the unbiased voting system, given the same accuracy of the system. A brief comparison of the two methods was conducted and we find that both methods have their own advantages and disadvantages.

CHAPTER 5 PEER-TO-PEER SYSTEM RELIABILITY

5.1 Introduction

Peer-to-peer (P2P) systems have recently received increasing attention from both research (see e.g. Leuf, 2002, Foster & Iamnitchi 2003, Steinmetz and Wehrle, 2005, Tian *et al*, 2006, and etc) and industry. P2P system is a large-scale distributed system where there is no central server that stores all data. The data is distributed among peers which have the ability to self-organize. In a P2P system, peers cooperate to achieve a desired service, such as: distributed computing (Anderson *et al*., 2002), file sharing (Saroiu *et al*., 2002), distributed storage (Rowstron and Druschel, 2001), communication (see e.g. Jabber), and real time media streaming (Hefeeda *et al.*, 2003, Liu *et al*. 2006, and Tu *et al*. 2005).

Currently, there is extensive research and literature in the areas of P2P technology and development. However, little research has been done on the reliability analysis of P2P network system. Most of the researchers think reliability is not a critical issue in P2P network systems as P2P network systems are always considered perfectly reliable. This is because even with extensive damages in the P2P network system causing many peers to fail, the whole P2P network system can still function in excellent condition.

Among the many applications of P2P systems, the research on media streaming by P2P has received increasing attention. This system provides users/peers the services to download simultaneously distributing media sources, which may be living or a playback of a recording, and enabling the users to playback the media while it is being downloaded. Xu *et al*. (2002) study two problems in P2P media streaming systems: the assignment of media data to multiple supplying peers and fast capacity amplification of the enter P2P system. Hefeeda & Bhargava (2003) propose a P2P media streaming model that can serve many clients in a cost effective manner and present a P2P streaming protocol used by a participating peer to request a media file from the system. Hefeeda *et al*. (2003) propose a novel P2P media streaming system PROMISE, encompassing the key functions of peer lookup, peer-based aggregated streaming, and dynamic adaptations to network and peer conditions. Zhang *et al*. (2005) present a Data-driven Overlay Network (DONet) for live media streaming, of which each node periodically exchange data availability information with a set of partners which the node shares media streaming data with. An implementation based on DONet, called

Coolstreaming is also introduced this chapter.This has rapidly attracted a large number of users enjoying the live media streaming from internet all around the world.

The performance of the service becomes a critical problem in the development of P2P media streaming network systems. It is mainly determined by streaming data distribution algorithm the network systems apply, the bandwidth of the internet within the P2P network systems and the unpredictable departure/failure of peers (Tu *et al*, 2005, and Piotrowski, *et al.*, 2006). Saroiu *et al.* (2002) report that half of the peers connecting to the network will be replaced by new participants within one hour in both Napster and Gnutella. Hence, the performance of the systems in the highly unsteady environment becomes important. Tu *et al.* (2005) present a simple model to study the effect of peer failures, which are defined as peers leaving the media streaming systems permanently, on the capacity growth of the media streaming systems. The distribution of the lifespan of the peers is considered to be arbitrary in the model. Zhang *et al.* (2004) present a dynamic passive replication scheme to improve the reliability of multicasting systems which comprises of unreliable peers. A reliability analysis is also conducted given the replication scheme.

From the perspective of the users of P2P media streaming systems, the most significant concern of the users is the performance of the software when downloading the huge volume of media data from a highly dynamic and unstable internet environment. The demanding users might have high requirement on the quality of media service provided by the P2P media streaming software such as PROMISE and *Coolstreaming* introduced previously. The P2P live media steaming software product with desirable features of running smoothly, recovering promptly from a sudden failure, high quality of the live media etc. will be attractive to the users and outperform other similar competing P2P live media streaming products in the market. Hence, it would be very important to evaluate the service quality accurately and better develop the product further to compete with other products. However, to the best of our knowledge, no research has been done on measuring and modeling the performance of P2P media streaming network systems from the users' perspective. In this chapter, we measure the system performance by calculating the reliability from the users' perspective under data transmission rate requirements of receiving data from other peers. A reliability model of P2P media streaming network systems is proposed under some necessary assumptions. Further analysis on this simple model is conducted by taking into account the time influence on the usage of the internet. The reliability of the P2P media streaming network system is then estimated by applying universal generating function, a powerful mathematical tool for solving the problems with multi-state systems (Levitin, 2005).

The rest of the chapter is organized as follows. Section 5.2 presents the reliability model formulated for the media streaming service of peer-to-peer systems. Section 5.3 introduces the algorithm of universal moment generating function to compute the service reliability of the P2P media streaming systems. In section 5.4, an illustrative example is introduced to explain how to compute the reliability of the P2P service. Section 5.5 conducts the further analysis of the P2P service reliability by taking into account the time influence on the usage of internet and in section 5.6 we consider an improvement strategy on P2P system reliability, that is, buffer technique and quantify the effect. Finally in section 5.7, a summary is given.

5.2 Reliability Model of P2P Systems

The notations used in this chapter are introduced as follows.

Acronyms

P2P	peer-to-peer
UGF	universal generating function

Nomenclature

N	number of all the peers composing the whole P2P network
M	state space of index m
B	total data transmission rate to the center peer from all the peers in the systems
b_i	data transmission rate to the center peer from peer i
p_i	probability that peer i is in connecting state
b_{ik}	the k^{th} state of data transmission rate of peer i
p_{bik}	probability that the data transmission rate of link i is b_{ik} given it is connecting state
K_i	state space of the data transmission rate of link i
B_m	m^{th} state of total data transmission rate to the center peer
Q_m	probability of m^{th} state of total data transmission rate
v	the threshold value to determine the service is successful or not
$R(v)$	reliability of the P2P media streaming service with respect to v
$B(t)$	total data transmission rate to the center peer at time t
$B_i(t)$	data transmission rate from peer i at time t
$p_i(t)$	probability that peer i is in connecting state at time t
$B_m(t)$	m^{th} state of total data transmission rate to the center peer at time t

$Q_m(t)$	probability of m^{th} state of total data transmission rate at time t
$R(v,t)$	reliability of the P2P media streaming service with respect to v at time t
$R_{service}(v)$	reliability of a specific service given threshold value v
$U(z)$	u-function representing the probability distribution of data transmission rate
$U(z,t)$	u-function representing the probability distribution of data transmission rate at time t

In this section, we formulate a new reliability model of P2P media streaming network service under certain performance requirement. To reduce the computation complexity, some necessary assumptions and simplifications are made.

Consider a Peer-to-peer media streaming network whose architecture is depicted in Figure 5.1 (Hefeeda, 2003). Here, the peers are interconnected through a P2P substrate. A peer in this network system can be a computer, PDA or other device. A service starts when a peer requests to download data, such as media streaming data, from the P2P network, and ends when all the requested data have been transferred to this requesting peer. The performance of this P2P network varies all the time during its data requesting period as the network performance is unstable and the number of online peers varies.

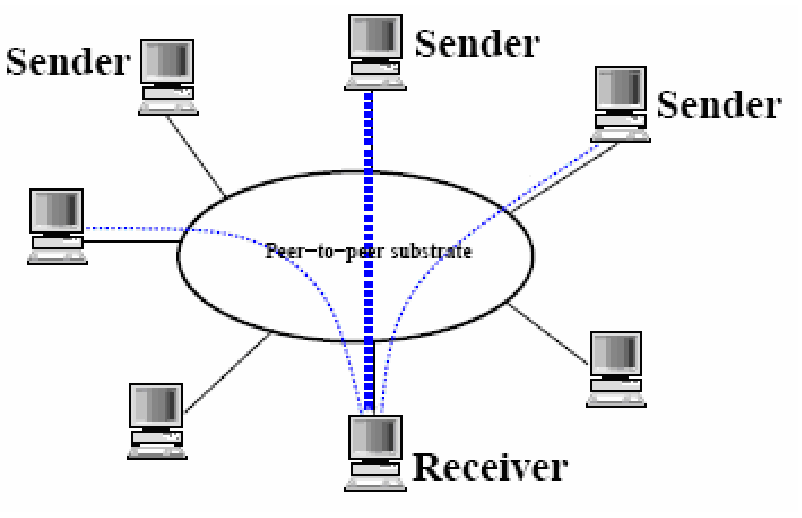

Figure 5.1 Architecture of P2Pmedia streaming network systems

From the perspective of the peer which is requesting data service, the architecture of the whole P2P network is simplified as star topology in Figure 5.2. In this topology, the center node represents the peer which is requesting media streaming service, and the periphery peers represent the peers which can provide the media streaming data to the center peer on the internet. Data are transmitted through the links between the periphery peers and the center peer at their own data transmission rate in units of kilobyte per second (kb/s).

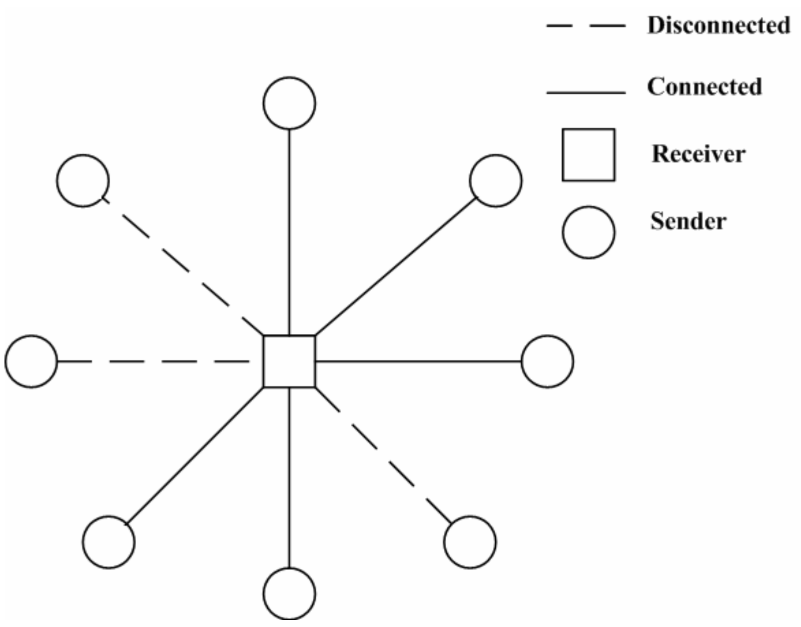

Figure 5.2 Topology of P2P network

We first formulate a simple reliability model of the above star topology assuming each peer *i* has a constant probability, p_i, of connecting directly to the center peer.

This model considers the connecting states of all the peers aretime independent. Here, we first provide the assumptions made to analyze this model:

1. all the peers and the links works independently;
2. peer *i* connects to the center peer successfully at probability p_i;
3. the center peer always works in perfect status;
4. the data transmission rate of transmitting data from peer *i* to center peer has multi states (with state space K_i for peer *i*), each state is assigned to constant probability;
5. the total data transmission rate to center peer is the sum of the data transmission rates of all the connecting links;
6. time to build the communication between peer *i* and center peer is negligible compared with the data transmission time;
7. the media data is transmitted continuously and not in package;
8. no replication scheme is applied in the system.

In the assumptions above, the periphery peer *i* in Figure 5.2 has only two states: 'connecting' and 'disconnecting'. We use state S=1 to represent the 'connecting state', and state S=0 to denote the 'disconnecting state'. For peer *i*, we have: $P_i(S=1)=p_i$, $P_i(S=0)=1-p_i$, i=1, 2,…, N, where N represents the number of all the peers composing the whole network.

The data transmission rate b_i of the link between periphery peer *i* and center peer has multi values b_{ik} (k=1, 2,…, K_i), the probability that the data transmission rate of link *i* is b_{ik} given peer *i* is in connecting state is p_{bik} .

When peer *i* is in 'disconnecting state', which is represented by the dash lines in Figure 5.2, link *i* does not really exist in the network system. At this time no data can be transmitted through link *i* between peer *i* and the center peer, which is requsting data, the data transmission rate b_i is 0 kb/s. Data can only be transmitted through link *i* when peer *i* is in 'connecting state'.

The above description of the P2P network can be mathematically represented in the Bayesian framework by a set of equations:

$$\begin{cases} P(b_i = b_{ik} \mid S_i = 1) = p_{bik} \ ; \\ P(b_i = b_{ik} \mid S_i = 0) = 0 \ ; \\ P(b_i = b_{ik}) = p_{bik} \cdot p_i \ . \end{cases} \tag{5.1}$$

Based on the assumption, the total data transmission rate to center peer is sum of data transmission rates of all the links connecting to the center peer:

$$B = \sum_{i \in \{S_i = 1\}}^{N} b_i \tag{5.2}$$

where the set $\{S_i = 1\}$ represents the set of the link *i* which is in the state of connecting to the center peer.

The number of states of links and peers is finite, so the total data transmission rate *B* has finite state space, say *M*. The probability that the total data transmission rate B=B_m is P(B=B_m)= Q_m.

The media streaming services provided by P2P network systems have a special requirement on the data transmission rate. The service is only available in a stable internet environment. Once the total data transmission rate of the links from other online peer computers is below a certain value *v*, the online media service will delay or stop. This is considered as a failure of the live service. The threshold value *v* is mainly determined by the type of services required/requested and the data encoding techniques in this computer which is requesting media service from the internet. Different threshold values *v* set different requirements on the performance of data transmission rate, which subsequently determine the reliability of this live media streaming service.

Hence we use the concept of service reliability to model and quantify users' satisfaction level in this chapter. The service reliability is defined as the probability that the data transmission rate is greater than *v* during the whole service time; the definition is represented by the following function of the threshold value *v*:

$$R(v) = \sum_{m \in M} Q_m \cdot 1(B_m > v) \tag{5.3}$$

where $1(\cdot)$ is an indicator function which equals to 1 if the condition is satisfied, 0 otherwise.

5.3 Algorithm for Computing the Service Reliability

5.3.1 Background of Universal Generating Function

The universal moment generating function technique (u-function) is used in this chapter to evaluate the reliability of P2P media streaming systems. The U-function has shown to be effective for reliability evaluation of multi state systems and an introduction can be found in Lisnianski and Levitin (2003).

Universal Generating Function (UGF) is a well-known and effective technique for the reliability analysis and optimization of various multi-state systems. Much research has been done on incorporating UGF into reliability analysis of various series-parallel systems, bridge systems, weighted voting systems, acyclic transmission networks, linear multi-state sliding-window system, linear consecutively connected systems, and acyclic consecutively connected networks (see Levitin, 2005 and Levitin & Dai, 2006). Lisnianski & Levitin (2003) briefly describe the application of UGF in many of these systems; Levitin (2005) provides a generalized view of the method and its application to the analysis and optimization of various types of binary and multi-state systems.

5.3.2 Universal Generating Function

The U-function of a discrete random variable *Y* is defined as following polynomial:

$$u(z)=\sum_{l=1}^{L}\alpha_l\cdot z^{y_l} \qquad (5.4)$$

where the variable *Y* has *L* possible values and α_l is the probability that $Y=y_l$. It is very easy to use the u-function to represent the probability mass function of two independent random variables $\varphi(Y_i,Y_j)$ by introducing composition operators. Simple algebraic operations on the individual u-functions are described in Lisnianski and Levitin (2003). All the composition operators take the form:

$$U(z)=u_i(z)\underset{\varphi}{\otimes}u_j(z)=\sum_{l=1}^{L_i}\alpha_{il}z^{y_{il}}\underset{\varphi}{\otimes}\sum_{h=1}^{L_j}\alpha_{ih}z^{y_{ih}}=\sum_{l=1}^{L_i}\sum_{h=1}^{L_j}\alpha_{il}\alpha_{jh}z^{\varphi(y_{il},y_{jh})} \tag{5.5}$$

In the case of P2P systems, the u-function can represent the distribution of data transmission rate of links. From Equation (5.2), the link i has data transmission rate b_{ik} with probability $p_{bik}\cdot p_i$ (the probability that data transmission rate is equal to 0 is 1-p_i). Therefore, the u-function takes the form:

$$u_i(z)=\sum_{k=1}^{K}\left(p_{bik}\cdot p_i z^{b_{ik}}+(1-p_i)z^0\right). \tag{5.6}$$

The total data transmission rate to center peer is the sum of the data transmission rates of all periphery peers. Using a composition operator with $\varphi(Y_i,Y_j)=Y_i+Y_j$, we get:

$$\begin{aligned}U_{ij}(z)&=u_i(z)\underset{+}{\otimes}u_j(z)\\&=\left[\sum_{k=1}^{K_i}\left(p_{bik}\cdot p_i\cdot z^{b_{ik}}+(1-p_i)z^0\right)\right]\bullet\left[\sum_{k=1}^{K_j}\left(p_{bjk}\cdot p_j\cdot z^{b_{jk}}+(1-p_j)z^0\right)\right]\end{aligned} \tag{5.7}$$

Hence, the u-function for the distribution of total data transmission rates to center peer can be represented as:

$$U(z)=\prod_{i=1}^{N}u_i(z)=\prod_{i=1}^{N}\left[\sum_{k=1}^{K_i}\left(p_{bik}\cdot p_i\cdot z^{b_{ik}}+(1-p_i)z^0\right)\right] \tag{5.8}$$

This can be recursively obtained by:

$$U_i(z) = U_{i-1}(z) \cdot u_i(z) = U_{i-1}(z)\left[\sum_{k=1}^{K_i}\left(p_{bik} \cdot p_i \cdot z^{b_{ik}} + (1-p_i)z^0\right)\right] \tag{5.9}$$

The final u-function calculated recursively by collecting like terms represents the distribution of total data transmission rate *B*, which takes the form of $U_N(z) = \sum_{m \in M} Q_m \cdot z^{B_m}$. With this distribution, we can obtain the reliability with respect to different performance requirement value *v*:

$$R(v) = \sum_{m \in M} Q_m \cdot 1(B_m > v) \tag{5.10}$$

5.3.3 Algorithm for Computing Service Reliability

With the universal generation function introduced above, a simple and efficient algorithm is proposed to evaluate the reliability of P2P media streaming network service systems:

```
Step 1
1. for all $i \in [1,N]$ do
2. collect the values of $p_{bik}$ , $p_i$, $b_{ik}$ for each peer $i$,
3. define $u_i$(z) by equation (5.6)
Step 2
4. for all $i \in [1,N]$ do
5. calculate $Ui$(z) by using equation (5.9) recursively.
Step 3
6. remove the terms whose total data transmission rate is below threshold v
   alue $v$.
```

Step 4

7. evaluate the service reliability of P2P media streaming service system by equation (5.10)

5.4 Illustrative Example

In this section, we illustrate how to formulate the reliability model of media streaming service of P2P system and how to estimate the reliability function given the internet condition and the information of service users. Consider the P2P media streaming system depicted in Figure 5.2. Table 5.1 lists the probability that peer *i* is in the 'connecting state'. The data transmission rate (DTR, in unit of kb/s) and the corresponding probability of each peer *i* are listed in Table 5.2.

Table 5.1 The probability that peer i is in 'connecting state'

Peer *i*	1	2	3	4	5	6
Probability	0.8	0.9	0.7	0.8	0.6	0.9

The u-function representing the distribution of total data transmission rate to center peer is obtained as follows:

$$\begin{aligned} U(z) &= \prod_{i=1}^{N} u_i(z) = \prod_{i=1}^{N}\left[\sum_{k=1}^{K_i} pb_{ik}\cdot p_i \cdot z^{b_{ik}} + (1-p_i)z^0\right] = \\ & 0.000048 + 0.000365z^{10} + 0.000475z^{15} + 0.00131z^{20} + 0.00262z^{25} \\ & + 0.00542z^{30} + 0.00959z^{35} + 0.0134z^{40} + 0.0246z^{45} + 0.037z^{50} + 0.0474z^{55} \\ & + 0.0559z^{60} + 0.0847z^{65} + 0.100z^{70} + 0.0889z^{75} + 0.0959z^{80} + 0.120z^{85} \\ & + 0.103z^{90} + 0.0595z^{95} + 0.0569z^{100} + 0.0594z^{105} + 0.0290z^{110} + 0.00427z^{115} \end{aligned} \qquad (5.11)$$

Table 5.2 Probability distribution of data transmission rate of link i

Link *i*				
1	DTR(kb/s)	10	15	20
	Probability	0.1	0.2	0.7
2	DTR(kb/s)	10	20	
	Probability	0.3	0.7	
3	DTR(kb/s)	15	20	
	Probability	0.6	0.4	
4	DTR(kb/s)	15	20	
	Probability	0.8	0.2	
5	DTR(kb/s)	20		
	Probability	1		
6	DTR(kb/s)	10	20	
	Probability	0.5	0.5	

From this u-function, we can calculate the reliability of this P2P system under the requirement of total data transmission rate to center peer. If the service requires the data transmission rate to be greater than 100, the reliability can be estimated as follows:

R(v=100)=0.0569+0.0594+0.0290+0.00427=0.150

If the requirement is set as 50, the reliability is calculated:

R(v=50)=0.0370+0.0474+0.0559+0.0847+0.100+0.0889+0.0959+0.120+0.103+ 0.0595+0.0569+0.0594+0.0290+0.00427=0.942

The service reliability of the P2P system is a function of the system performance requirement, which is the total data transmission rate of the links from other computers to the center computer in this P2P network system. Higher requirement means a lower service reliability of the system. To improve the service reliability of the P2P system, we need to improve both the system performance as well as lower the performance requirement of the system by applying new encoding techniques. The service reliability function with respect to the requirement of the system performance (*v*) is plotted in Figure 5.3:

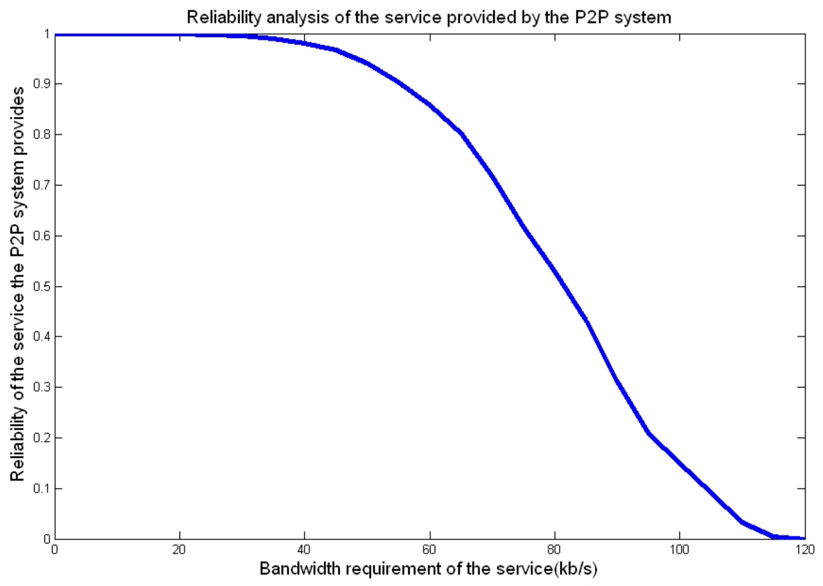

Figure 5.3 Service reliability of the P2P system under performance requirement

5.5 Time-dependent Model of the P2P Network System

5.5.1 The Modified Model

In the model formulated above, the assumption that the periphery peer i is in 'connecting state' with constant probability p_i was made. In this case, all the peers work independently on time which makes the computation much simpler. However, what happens more often in the real world is that P2P users (peers) surf the internet taking into account the time factor. Some users may prefer to surf the internet in the morning, and some other users may prefer night. Different service types also determine the different usage of the internet, and hence, the usage time is also influenced by the service. In these situations, the probability function of 'connecting state' is time dependent, and we denote it as $p_i(t)$.

In practice, the data transmission rate of internet connection between the computer users is also strongly influenced by the time factor. This means the data transmission rate b_i for ith connection to the center peer needs to be modified to account for time, $b_i(t)$. To simplify the computing complexity, here we also assume that the data transmission rate $b_i(t)$ has only finite number of states at each time point.

To simplify this reliability model, we make some necessary assumptions here.

1. all the peers and the links work independently;
2. peer i connects to the center peer successfully with probability $p_i(t)$;
3. the center peer always work in perfect status;
4. the data transmission rate of transmitting data from peer i to center peer is $b_i(t)$, which is a multi-state value at each time point;
5. the total data transmission rate to center peer is the sum of the data transmission rates of all connecting links;

6. time to establish communication between peer *i* and the center peer is negligible compared with the data transmission time;
7. the media streaming data is transmitted continuously and not in packages;
8. no replication scheme is applied in the system.

Comparing the assumptions above with those in section 2, items 2 and 4 are different as some properties in this system are time dependent. In this model, we assume the data transmission rate from peer *i* to the center peer at time *t* is $b_i(t)$ in this section. The total data transmission rate to the center peer $B(t)$ is the sum of data transmission rate of all the links that connect to the center peer, which is represented by equation (5.12):

$$B(t) = \sum_{i \in \{S_i = 1\}}^{N} b_i(t) \tag{5.12}$$

As the number of states of links and peers is finite, the total data transmission rate $B(t)$ has finite state space M, and so, the probability that the total data transmission rate $B(t)=B_m(t)$ is $P(B(t)=B_m(t))=Q_m(t)$.

This service can only be carried out successfully in a good internet environment. Once the total data transmission rate of the links from other online computers is below a certain value v, the media system will stop the media service, and this is considered as failure of the media system. The service reliability is therefore a time dependent function as the data transmission rate varies according to time:

$$R(v,t) = \sum_{m \in M} Q_m(t) \cdot 1(B_m(t) > v) \tag{5.13}$$

From Equation (5.13), the system reliability function can be plotted with respect to time. For a specific service the user starts, the service reliability is defined as the continuous average over the service time. This is the definite integral of the reliability function in equation (5.13) from the time service begins t_a, to the end of the service t_b, over the duration of the service (t_b-t_a):

$$R_{service}(v) = \frac{\int_{t_a}^{t_b} R(v,t)dt}{t_b - t_a} \tag{5.14}$$

We can extend the reliability analysis and u-function technique in the model in section 2 and 3 by replacing p_i and b_i in above model with $p_i(t)$ and $b_i(t)$ respectively. The universal generating function of peer i is subsequently extended to time area:

$$u_i(z,t) = \sum_{l=1}^{L} \alpha_{il}(t) \cdot z^{y_{il}(t)} = p_i(t) z^{b_i(t)} + (1 - p_i(t)) z^0 \tag{5.15}$$

$$\begin{aligned} U_{ij}(z,t) &= u_i(z,t) \underset{+}{\otimes} u_j(z,t) \\ &= \left[p_i(t) z^{b_i(t)} + (1 - p_i(t)) z^0\right] \cdot \left[p_j(t) z^{b_j(t)} + (1 - p_j(t)) z^0\right] \end{aligned} \tag{5.16}$$

To estimate the service reliability of P2P network system, we need to collect necessary data, including the users' profile and the data of internet condition over time. The users' profile includes users' habits, what time the users prefer to use the internet, what kind of services the user requests etc. These can be obtained from a questionnaire. The data of internet condition can be obtained directly from the public reports released from the government or other public organizations.

After the analysis, we can get the reliability function which satisfies the requirements on the data transmission rate. Since the reliability is a function of time, we can plot the reliability-time curve to determine when the reliability function reaches its peaks in the given period of time. This can help determine when the P2P network provides the greatest reliability, and in turn, provide information to the users on the best times to join the network to experience the most stability and reliability.

5.5.2 Numerical Example of the Modified Model

A simple numerical example is presented here to illustrate the extended model with time factor considerations. Suppose we obtain the data of the probability of connecting to internet of the users of P2P media streaming systems and the data transmission rate of network links from a survey or other sources. The probability that each individual peer connects to the center peer is given in Table 5.3 for different time periods of the day. For the sake of simplicity, we use the probability of connecting to network in simple forms. The data transmission rate of the links is shown in Table 5.4. To further simplify computations, we assume that the data transmission rates in a short period are constant.

Table 5.3 Time-dependent connecting probability of each peer

Peers	12:00-18:00
1	0.5+0.02*t
2	0.85
3	0.4+0.03*t
4	0.8-0.01*t
5	0.75
6	0.7+0.01*t

From the perspective of users of this P2P media steaming software, the requirement level *v* is set at 60 kb/s which means the service will be considered as failure when the total bandwidth rate is lower than the requirement. Given this definition, the service reliability is calculated as the probability that the total bandwidth rate is greater than 60 kb/s.

Table 5.4 The data transmission rate of each network link

Time / rate	12:00-13:00	13:00-14:00	14:00-15:00	15:00-16:00	16:00-17:00	17:00-18:00
$b_1(t)$	18	15	17	20	19	20
$b_2(t)$	20	22	23	25	23	20
$b_3(t)$	18	20	25	28	30	32
$b_4(t)$	25	23	20	20	20	22
$b_5(t)$	15	15	18	20	20	25
$b_6(t)$	20	23	25	23	20	23

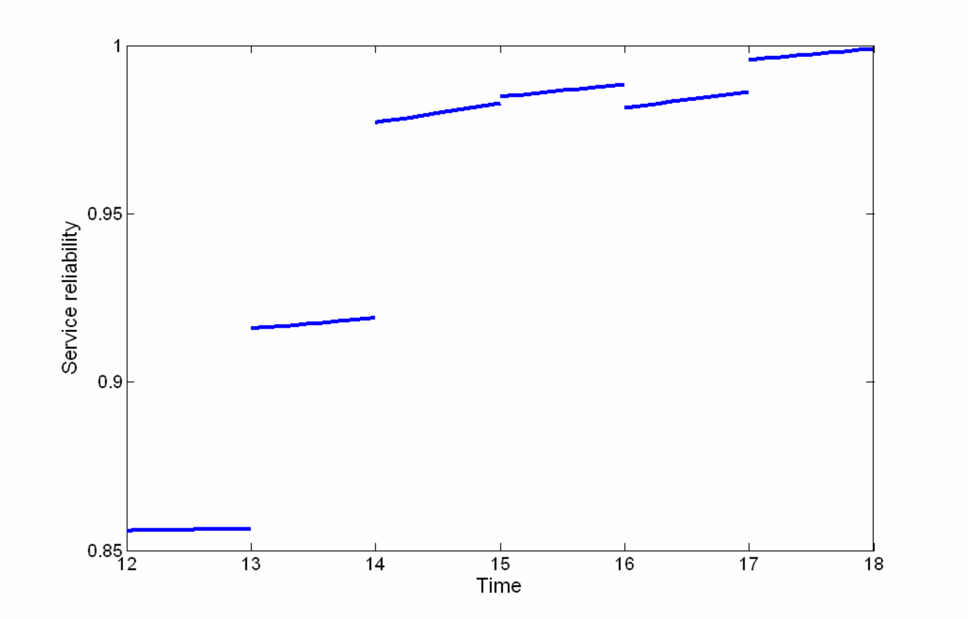

Figure 5.4 Time-dependent service reliability of P2P media streaming systems

Given this performance requirement, the service reliability of the P2P network system is calculated and plotted in Figures 5.4, in which time is in unit of hour. As the probability of connecting the central peer is time dependent for each periphery peer and its data transmission rate varies with time, the service reliability of the P2P system varies with time as well, as seen from the figures.

From Figure 5.4, we find that the service reliability is significantly related to both connecting probability and the data transmission rate of each peer. Within each period when the data transmission rate from periphery peers are assumed to be constant, the service reliability increases monotonically over time because the average probability of periphery peers connecting to the network increases. Obviously, average service reliability in each period is also highly related to the average data transmission rate of the corresponding period. The conclusions above actually describe one inherent and famous feature of P2P network: the more participants, the better system performs.

It is easy to calculate service reliability of the network over certain periods of time. Suppose a user is watching a live football match on the internet through P2P media streaming network from 14:00 to 16:00. Service reliability for viewing this football match online can be evaluated by equation (5.14) as:

$$R_{service}(v) = \frac{\int_{t_a}^{t_b} R(v,t)dt}{t_b - t_a} = 0.9794$$

This shows that the service will be provided successfully about 97.94% time of the entire match time. It also represents users are satisfied with the performance in 97.94% time.

5.6 Reliability Model with Buffer Technique

5.6.1 Problem Statement

In various implements of P2P media streaming systems, buffer techniques are applied to ensure the systems are tolerant to a satisfactory level against a sudden interruption or temporary transmission failure over the internet for a short period. This improves the reliability of the whole system dramatically by avoiding loss of transmitting data. In the following subsection, we formulate a new reliability model taking into account the significant effect from buffer techniques applied in the P2P systems.

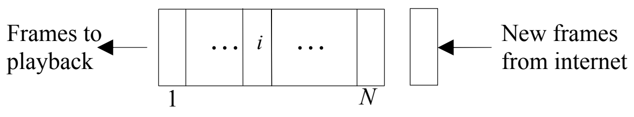

Figure 5.5 Structure of frame-buffer in P2P Systems

The figure above describes the structure of buffer for a P2P media streaming system. The frame buffer receives new frames from its parent peers and queues the coming frames in the buffer in terms of frame sequence for playback. The newest frames are stored at the end of the queue while the oldest one, which is placed at the beginning unit of the buffer, say *buffer*(1), is ready to playback for the users. After the frames in buffer(1) is pushed to playback, the rest frames in the buffer move forward to previous buffer, that is, the frame in *buffer*(*i*) moves to *buffer*(*i*-1). The frames in buffer can also be forwarded to their child peers as researched by Yeh and Pui (2005), but this is not the main concern of this section.

In our model, we assume constant playback rate while data transmission rate depends on the unstable internet condition. Video frames are pushed to playback in packet size, that is, the size of a buffer. So the time to consume video frames in size of a buffer is buffer size/playback rate. During this period, frames in buffer grow in data transmission rate until all buffers are full. We consider two types of failures in the process of data transmission and consumption: overflow and starvation.

Overflow

Buffer overflow happens if the streaming data transmission rate from its parent peers through internet is higher than playback rate of the video streaming. In our model, we adopt the policy of dropping all new frames when the buffers are full. New frames are accepted only when one or more buffers are empty as a consequence that the frames in the earlier buffers have been consumed by video application. The request for new frames from this peer will be resent to its parent peers (periphery peers in the previous model) when one or more buffers are ready to receive new frames. The delay of starting a new connection to other peers is considered to be negligible in this section.

Starvation

As stressed in the previous section, due to dynamically unstable internet, data transmission rate often goes well below the required level that P2P software applications starve from new frames. Starvation occurs when the P2P software applications have used up all the frames in buffer but no new frames are received because of unstable internet condition or unexpected events (internet traffic jam, peers leaving, etc). In this case, the video player has to be paused until the internet condition improves and enough new frames are transmitted to this peer. We consider starvation as the only source of failure to P2P media streaming system in our reliability model for simplicity. The failure is defined mathematically as an event: buffer(1)=0 & data transmission rate<playback rate. The failure will continue for a while until buffer(1) becomes 1.

Following figure is depicted to illustrate the diagram of processing video frames with two failures above.

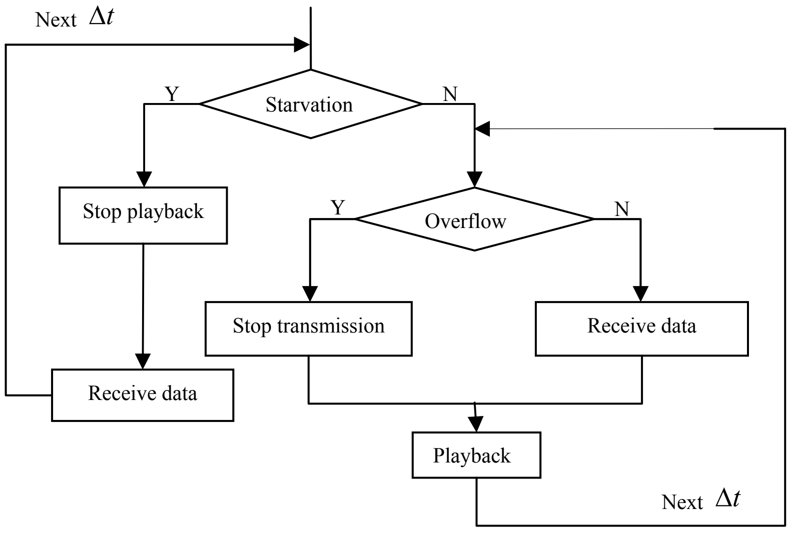

Figure 5.6 Diagram of processing video frames

Reliability Definition

To differentiate from the previous models, we re-define reliability of P2P media streaming system with taking into account the effect of application of buffer scheme in the P2P system.

The reliability is redefined as the probability that video frames can be processed smoothly to the users' end, that is, mathematically,

$$Reliability = 1 - p(buffer(1) = 0) \tag{5.17}$$

5.6.2 Markov model

Implementation of buffer scheme can dramatically improve the performance of P2P media streaming system. When network bandwidth drops suddenly due to unknown events which always happen, the video system is still able to playback the frames buffered in advance till the network system recovers itself to a normal or high level.

When network condition becomes better, that means, P2P media streaming system can receive video data at high rate. The surplus frames will be put in the buffers for playback in future. In this sense, the P2P system, with buffer scheme, prepares frames when the network condition is good and consumes the buffered frames when it turns to bad condition. In this subsection we discuss the calculation of reliability of P2P system with buffer technique and compare this result to the case with no buffer technique considered to find out how the buffer technique improves the P2P media streaming system.

From the analysis above, we know the reliability is highly related to the factors defined as follows: video playback rate, video volume, buffer size, and network condition. For the sake of simplicity, in our thesis we consider the network condition as the sole unknown factor whose value varies over time and on which the instantaneous performance of P2P media streaming system is dependent, while other factors are considered as environment parameters for the model which are constant during the playback time by our assumption.

When the video system is using up the frames, it pushes new frames from the buffer. As we stated in our assumption, the playback rate and frames size are considered to be constant. The time of consuming a packet of frames is thereafter constant, that is $\Delta t = t_{n+1} - t_n = \dfrac{Frame\ Size}{Playback\ Rate}$. By this means, the number of packets in the buffer reduces by 1 every Δt. At the same time, the buffer receives video data at a changing rate from other peers over the internet. So, comparing the Playback rate and receiving rate, packets of video data in buffer have 3 possible trends to go: decreasing, keeping unchanged, and increasing.

Consider a stochastic process $\{X_n,\ n = 0,1,2\ldots\}$ to represent the buffer status of P2P media streaming system and let $X_n = i$ denote the state that *buffer*(i) is full but *buffer*(i+1) is empty at time n. Time interval is set as Δt in this chapter. Event starvation only occurs at X_n=0.

As we assumed in section 2, the connecting states of all the peers are not time-dependent. By this assumption, the probability that the stochastic process is in state X_n=i at time t=n is independent of the network condition given the fact the X_{n-1}=i_{n-1}. From this, we conclude the buffer status satisfies Markov property and in the following part, we will model P2P media streaming system reliability with this property.

Due to the change of network condition, buffer state will also change from $X_n = i$ to $X_{n+1} = j$ with probability P_{ij}:

$$\begin{aligned} P_{ij} &= P\{X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, X_1 = i_1, X_0 = i_0\} \\ &= P\{X_{n+1} = j \mid X_n = i\} \end{aligned} \tag{5.18}$$

for all states.

After defining the probability P_{ij}, the next step is to calculate P_{ij} for each i and j and then to find P_0 which states the unreliability of this P2P media streaming system.

From eq. (5.11), we describe the unstable network condition at different levels in different probabilities. If at time t=n, the buffer is in state X_n=i, then at t=n+1, the buffer may transit to state X_{n+1} =j_{n+1} with a different probability which is dependent on network condition (the effect of network condition is assumed constant to guarantee its Markov property) and current state i simultaneously.

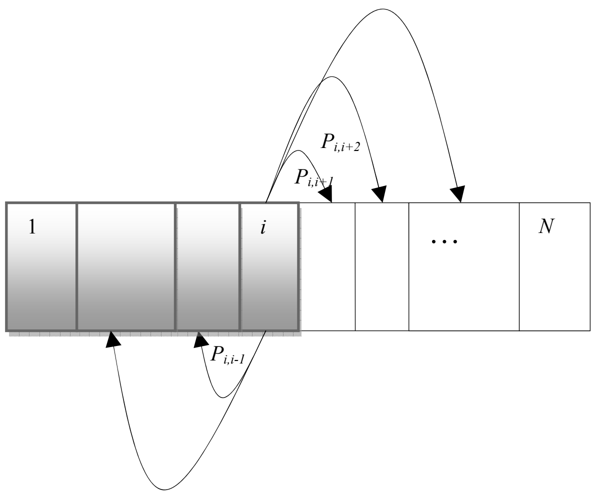

Figure 5.7 Frames states transition

As we stated earlier, unreliability of the P2P system is the probability of starvation occurring, that is the probability no frames is buffered. As the Markov chain above for the P2P buffer system is irreducible ergodic, this probability can be calculated by finding its limiting probability π_1.

So we have

$$\pi_j = \lim_{n\to\infty} P_{ij}^n \tag{5.19}$$

and the solution is obtained from ($j \geq 0$) solving the equations

$$\pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij}$$

$$\sum_{i=0}^{\infty} \pi_j = 1 \tag{5.20}$$

Then reliability is represented by

$$R = 1 - \pi_0 \cdot P(DTR \leq PR) \tag{5.21}$$

5.6.3 Numerical Example

In this subsection, we apply a numerical example to illustrate the process of calculating reliability of P2P media streaming system with buffer technique.

Suppose a P2P system playback the video frames at rate 50 kb/s. The size of a frame is 25 kb and the buffer accommodates 4 seconds of frames, that is, 200 kb in all. So time interval is $\Delta t = 0.5s$. To study the effect of unstable network environment, we use the results of estimating total data transmission rate to center peer from eq. (5.11) in the earlier section.

The transition probability matrix is obtained as follows.

$$\mathbf{P} = \begin{bmatrix} 0.095 & 0.755 & 0.150 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.095 & 0.755 & 0.150 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.095 & 0.755 & 0.150 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.095 & 0.755 & 0.150 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.095 & 0.755 & 0.150 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.095 & 0.755 & 0.150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.095 & 0.755 & 0.150 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.095 & 0.755 & 0.150 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.095 & 0.905 \end{bmatrix}$$

With the transition matrix, the limiting probabilities for each state are obtained by eq. (5.20):

$$\pi = [0.0015 \quad 0.0139 \quad 0.0243 \quad 0.0384 \quad 0.0606 \quad 0.0956 \quad 0.1510 \quad 0.2384 \quad 0.3764].$$

Hence reliability is calculated as $R = 1 - \pi_0 = 0.9985$. Compared to the result in section 5.2, for the P2P system implemented with buffer technique, service reliability is highly improved.

As the P2P system is a large scale distributed system, quantitative validation of the reliability model is quite expensive. Qualitative validation can be done by choosing different number of buffers and comparing the results of different systems. Obviously, the systems with more buffers appear to be more robust. This conclusion can also be proved by our Markov model: the probability of entering absorb state will be increasing if some buffers are removed.

5.7 Summary

Peer-to-peer systems have recently received increasing and extensive attention both from research and industry. However, most of the recent research focuses on the structures and algorithms of P2P system and little research has been done on the reliability analysis of the systems.

As performance becomes a critical issue in the highly dynamic environment where peers leave or fail unpredictably, the condition of the internet is highly unstable. P2P system is a large-scale distributed network system with huge complex topology. It is very hard to formulate the reliability model to evaluate the system performance quantitatively because of its vast complexity. However, from the users' perspective, the P2P media-streaming network can be simplified as star topology where current user is the center peer and other peers are the periphery peers. From this topology, we formulated a simple reliability model to estimate the service reliability as a measure of the system performance with service quality considerations. With this reliability model, the performance of service provided by the P2P media streaming system can be obtained easily with information on the internet conditions and user profiles, which can be collected from survey or database of some public agencies.

As the condition of internet is highly dynamic over different times of the day, further analysis on the reliability modeling of the P2P media streaming system is proposed to account for the time effects. The universal generating function (UGF) is then used as the method to calculate the reliability.

Buffer techniques are commonly applied to store a few segments of media data ahead to hide transient extra delays in packet arrivals, improving the performance of the P2P media streaming systems (Hefeeda and Bhargava, 2003, and Zhang *et al.*, 2005). In this thesis, we further build a reliability model to take into account the effect from buffer technique on P2P system reliability. The real performance of the P2P media streaming system are better than what we concluded in the earlier part of this chapter because of the application of the replication scheme in the real systems. A numerical example is used to illustrate the computations.

CHAPTER 6 UNCERTAINTY ANALYSIS IN RELIABILITY MODELING

6.1 Introduction

Reliability modeling has gained considerable interest and acceptance by applying probabilistic methods to the real-world situation. A software usually contains one or more basic modules or components that are functioning together to achieve some tasks. These modules can be of various types resulting in a wide range of software and system reliability models proposed, e.g. Pham (2000), and Xie *et al.* (2004), Myrtveit *et al.* (2005).

Except for non-parametric models (Xie *et al.*, 1997), most reliability models require some estimates of parameters. For example, the exponential model widely used for reliability analysis during the operational phase (Yang & Xie, 2000 and Dai *et al.*, 2003a), has a failure rate parameter (λ); the JM model (Jelinski & Moranda, 1972) is a traditional Markov model for software reliability with two parameters: the initial number of faults (K_0) and the failure intensity contributed by one fault (ϕ); the Goel-Okumoto model (Goel & Okumoto, 1979) is a classical *NHPP* model with two parameters (*a* and *b*) etc.

In order to apply the models to predict the reliability of the component, the parameters of the models need to be known or estimated. Field data or data from components with similar functionality are usually available to help estimate these parameters, but the estimators are subject to random variation because they are functions of random phenomena. Parameter uncertainty arises when the input parameters are unknown. Moreover, the reliability computed from the models which are functions of these parameters is not sufficiently precise when the parameters are uncertain, see e.g. O'Connor (1995) and Wooff *et al.* (2002).

Uncertainty analysis aims to quantify uncertainty associated with performance output as a result of uncertainties in the parameters. Hence, it is necessary to determine the uncertainty in the parameters for the modeling work. The uncertainty can be better described with a probability model than with a single point estimate. From the probability model, measures of the uncertainty (such as variance, confidence interval) can be obtained. To describe a probability model requires more data than to obtain the point estimate of the parameter. Jewell (1985) analyzed uncertain parameters for the JM model by using the Bayesian analysis. Masera (1987) proposed a method for computing the uncertainty propagation in fault trees when the lognormal distribution is used for failure rates of the components. Haverkort & Meeuwissen (1995) dealt with uncertain parameters for the Markov-reward models. Adams (1996) presented a mathematical model to calculate the confidence intervals that account for any uncertainty concerning the operational profile of the system. Yin & Trivedi (1999) studied the uncertain parameters for Goel-Okumoto model and S-shaped models by implementing a simplified Bayesian approach. Then, Yin *et al.* (2001) addressed the parameter uncertainty problem in reliability modeling with Markov models. Soundappan *et al.* (2004) compared the differences and relationships between evidence theory and Bayesian theory in uncertainty reliability modeling. To achieve sufficient accuracy in the uncertainty analysis, the above approaches need to collect more test data, e.g. Yin & Trivedi (1999) used 97 failure times for the uncertainty analysis.

However, one special characteristic of software reliability modeling or testing is insufficient failure data, see e.g. Miller *et al.* (1992). Failure data are usually scarce and limited to a single test. The more reliable the software is, the less failure data the testers can collect. Insufficient failure data makes software reliability modeling difficult, and makes its uncertainty analysis much more challenging. It is different from the uncertainty analysis in other areas with sufficient data, as summarized by Kurowicka & Cooke (2006). Though some previous research (e.g. Jewell, 1985, Yin & Trivedi, 1999) note this problem and suggest using the Bayesian approach to incorporate historical data into prior distributions, however they do not propose a systematic and practical approach on how to incorporate experts' suggestions with historical data for uncertainty analysis. For instance, Yin & Trivedi (1999) simply assumed the prior distribution is known, using for example a uniform distribution as a prior. They do not introduce how to comprehensively derive it from experts' suggestions and historical data.

This chapter not only addresses the uncertainty problem, but more importantly presents a solution for the challenge of insufficient data during a software test and reliability. It is observed that some information such as expert knowledge, historical data from similar projects, and developmental environment can contribute to the uncertainty analysis. For example, the team for development often has information of the development process, debugging method, test plan of the component, etc., and experts, managers or consultants may know what type of distributions certain parameters should follow and what conditions they are subjected to.

Thus, this chapter combines the Maximum-Entropy Principle (MEP) with Bayesian approach (BA) to solve the above challenges. MEP (Kapur, 1989) is a technique that applies the physical principle of *Entropy*, which states that without external interference, this measure of disorder will always tend to the maximum. This provides a probability distribution that is consistent with known constraints expressed in terms of the expected values of one or more quantities. The capability of the MEP can be integrated into the Bayesian approach (BA) to derive the priori distribution which can incorporate not only the historical data but also experts' suggestions, constraints, expected values, and other information in developmental process, which was introduced by Berger (1985). This chapter is the first to apply the MEP with the BA into the uncertainty analysis of software reliability. This is specifically appropriate to highly reliable software where only a few failure data is available from a single test of the project within a limited time frame.

After exploring the uncertainty for a single software component with multiple correlated parameters, this chapter further extends the uncertainty analysis to more complicated modular-software or systems that contain multiple components/modules, each with its own respective distributions and uncertain parameters, e.g. Kim *et al.* (2004). Many tools have been proposed to evaluate the system reliability, such as the Markov models (Dai *et al.*, 2003b), Bayes Network, Graph Theory (Dai & Levitin, 2006), Stochastic Petri Net, Fault Tree Analysis (Masera, 1987), etc. All these tools are functions of the components' parameters and if the parameters are not known precisely, the uncertainty in the entire reliability obtained from these tools will be further amplified, see e.g. Haverkort & Meeuwissen (1995) and Yin *et al.* (2001). This chapter further studies the uncertainties in large and complicated systems using a Monte Carlo (MC) approach.

The rest of the chapter is organized as follows: Section 6.2 introduces the reliability modeling and then discusses the uncertainty problems; Section 6.3 presents the MEP with BA for the uncertain analysis on a software component, and introduces the MC approach for the uncertainty of modular-software and complicated systems; and in Section 6.4, some examples belonging to different modeling categories (NHPP, Markov, Graph) are illustrated, where a new model improving Dai & Levitin (2006)'s model is also exhibited. Section 6.5 summarizes this chapter.

6.2 Overview of Reliability Modeling and Uncertainty Problems

Software reliability is an important index for software systems. A software system may contain multiple components or modules. The parameters of each component have to be known before the whole software/system reliability is evaluated. Point estimation methods such as the maximum likelihood or the method of least squares are usually used to obtain estimates of these parameters. We will briefly introduce the general steps of reliability modeling for the individual components, and then review the whole system reliability and the uncertainty problems.

6.2.1 Reliability Model of a Single Component

A software system may contain one or more components which are basic units of the system. Different components may have their own reliability models with respective parameters. To study the software/system reliability, the parameters of those components should be known.

The parameters in the model of a component are usually obtained by the methods of MLE (Maximum Likelihood Estimate). The general MLE steps are given below

Test the component and record the failure times: Let t_k $(k = 1,2,...,n)$ denote the observed time between $(k-1)^{st}$ and k^{th} failure. Let s_k denote the time to failure *k*. Then s_k is given by $s_k = \sum_{i=1}^{k} t_i$.

Compute the joint density or likelihood function: Given $s_1, s_2, ..., s_n$, the likelihood function can be written by (Trivedi, 1982)

$$f_S(s_1, s_2, ..., s_n) = \exp\{-m(s_n)\} \cdot \prod_{i=1}^{n} \lambda(s_i) \qquad (6.1)$$

where $m(t)$ is the mean value function and $\lambda(t) = \dfrac{dm(t)}{dt}$ is the failure intensity function. Different mean value functions $m(t)$ may contain different parameters, see e.g. Xie & Dai (2004: pp. 101-109).

Get the parameters that maximize the likelihood function: Take the derivative with respect to each parameter, and then let it be 0. By solving these equations, the parameters are obtained. Usually, these equations are numerically tractable.

6.2.2 System Reliability Model with Multiple Components

After obtaining the models and parameters of those components, the system reliability can be evaluated according to the architecture and relationship of the components. The Markov model, SPN (Stochastic Petri Net), fault tree analysis, and reliability block diagram, are some popular tools to evaluate the system reliability given the parameters of its contained components.

For example, a Markov chain is characterized by its state space together with the transition probabilities over time between these states. Usually, there are four steps to construct and solve the Markov chain models: 1) Setting up the Markov chain model; 2) List the Chapman-Kolmogorov equations; 3) Solve those equations to obtain state probabilities; 4) Obtain the reliability by summing up the probabilities of those reliable states. The detailed steps are shown by Xie & Dai (2004: pp. 19-36), and other tools of Bayesian network, fault tree analysis, reliability block diagram, and graph theory are introduced by Xie *et al.* (2004).

Regardless of the tools used to evaluate the system reliability, they have a common characteristic, i.e. the system reliability is a function combining the parameters of its individual components. Thus, the uncertainty in the parameters will affect the accuracy of the system reliability, which will be discussed in the next subsection.

6.2.3 Uncertainty Problems of the Parameters

In the reliability modeling described above, the parameters are estimated from the observed data using methods like the maximum likelihood. These estimators are subject to random variation as they are functions of random observations. The randomness arises from various causes. The faults in a component are initially unknown and their appearance (to cause failures) depends on the test procedures and strategies, such as random test or cluster test etc. Therefore, the data of failure times are uncertain. After collecting the test data, the reliability models to fit the data are selected subjectively, depending on the modeler's experience, knowledge, preference etc. Hence, the selection of reliability models is also uncertain.

In highly reliable and safety-critical systems (like those in nuclear power plants) where failures are extremely expensive, often failure data is scarce. With very small sample sizes, the uncertainty or error of the estimated parameters is large.

Thus, system reliability computed from the function of the uncertain parameters is also uncertain. For large complex systems with many components, the uncertainty of each individual parameter amplifies the uncertainty of the system reliability. Ignoring the parameter uncertainty can result in grossly underestimating the uncertainty in the total system reliability, which in turn leads to an overly optimistic expectation of the system reliability and an underestimation of the risk involved when using the system reliability measure for decision making.

Therefore, uncertainty analysis is necessary for the reliability modeling of both the individual components and the whole system. For a single component, the parameter uncertainty analysis focuses on the parameter estimation of the individual reliability model. For the entire system, uncertainty analysis focuses on the effect of the uncertain parameters of different components on the final system reliability. The measures of variance, confidence interval, percentiles, bounds etc can better represent the uncertainty of the reliability, and provide a more credible and more detailed result for the system reliability than only a point-estimated value.

6.3 Uncertainty Analysis by MEP and Bayesian Approach

6.3.1 Bayesian Analysis for Probability Distributions

We apply the Bayesian approach here to quantify the uncertainty in the component parameters. This approach combines the prior knowledge/information of the unknown parameter with current data/observations to deduce the posterior probability distribution of the parameter. We apply this approach here to quantify the uncertainty about the component parameters. Moreover, this approach can also handle the correlation among those parameters by using the joint distributions.

Assumptions:

1) The parameters modeling a component are denoted by $\vec{a} = \{a_1, a_2, ..., a_m\}$. The mean value function of the model is denoted by $m(t \mid \vec{a})$ and the failure intensity function by $\lambda(t \mid \vec{a})$.

2) The prior joint distribution of the parameters is denoted by $p(\vec{a})$ which is unknown.

3) The component is tested and a total of *n* failures have been observed. Let s_k denote the time to the *k*-th failure *k* (*k*=1,2,…,*n*), and $\vec{s} = \{s_1, s_2, ..., s_n\}$ the vector of failure times which are conditionally independent.

Then, given the prior distribution and observations, the posterior distribution can be obtained by

$$p(\vec{a} \mid \vec{s}) \propto p(\vec{a}) \cdot p(\vec{s} \mid \vec{a}) \tag{6.2}$$

where

$$p(\vec{s} \mid \vec{a}) = \exp\{-m(s_n \mid \vec{a})\} \cdot \prod_{i=1}^{n} \lambda(s_i \mid \vec{a}) \tag{6.3}$$

The above standard Bayesian approach is well known and straightforward. However, applying this to software reliability modeling poses several challenges specific to software testing and reliability. It is an important characteristic that the number of failure data is usually scarce in a single test. The lack of failure data in a project has challenged the modeling of software reliability, which makes estimating proper posterior distributions more difficult.

Fortunately, prior information such as expert knowledge, historical data from similar experiments are typically available. Therefore, we propose to theoretically incorporate the experts' suggestions and historical data from previous projects into the prior distribution in Eq. (6.2), i.e. $p(\vec{a})$. The following shows how to transform expert knowledge and historical data by integrating the Maximum-Entropy Principle (Kapur, 1989) into the Bayesian approach.

6.3.2 Maximum-Entropy Principle (MEP)

Though the single test in the current project lacks sufficient failure data for modeling, yet historical data, previous experiences, expert suggestions and other environmental information are useful. For example, a development team should have the knowledge of developmental process, debugging method, test procedures and so on. The related information can be transformed into a prior distribution through the Maximum-Entropy Principle (MEP) method.

MEP (Kapur, 1989) is a technique that applies the physical principle of *Entropy* which states that without external interference, the *Entropy* which measures the disorder always tends to the maximum. Entropy has a direct relationship to information theory, and in a sense measures the amount of uncertainty in the probability distribution. This measure provides a probability distribution that is consistent with known constraints expressed in terms of one or more quantities. Let *Y* be a random variable with pdf *f*, defined on $D_y \subset \mathbf{R}$ (the real number). The uncertainty concerning *Y* measured by the Entropy Function is given as

$$H(f) \equiv -\int_{D_y} f(y) \cdot \ln[f(y)]dy \,. \tag{6.4}$$

$H(f)$ also measures the quantity of information after the observation of a realization y_0 of *Y*.

Suppose g_r (r=1,2,…,m) are a group of known functions. If we have prior information concerning Y, e.g., we know that:

$$\int_{D_y} f(y) \cdot g_r(y) dy = \bar{g}_r , \tag{6.5}$$

then, the MEP states that (Kapur, 1989), considering prior knowledge about Y, the most likely distribution of Y is a distribution that maximizes $H(f)$ subject to Eq. (6.5) and Eq. (6.6).

$$\int_{D_y} f(y) dy = 1 \tag{6.6}$$

For example, suppose the prior mean is specified, and among prior distributions with this mean, in the MEP, the distribution which maximizes $H(f)$ is sought.

The MEP under the case where there is no other partial information leads to the distribution of "most uncertainty", which for certain discrete cases results in the non-informative prior. With partial prior information available, we then consider this information in the form of restrictions on the prior, hence helping us shape the prior. This partial information can come in the form of both subjective and objective information, e.g., subjective information (such as expert's prior opinion that the lifetime is exponentially distributed), and objective information (such as historical data enabling some calculation of the moments). More details and examples will be given in the following subsections 3.3 and 3.4.

6.3.3 Extract Data from MEP

To combine the MEP with the BA for the uncertainty analysis in software reliability, it is important to extract data from the experts and history using the MEP and then input them into the prior distributions of the BA. The goal of MEP is to incorporate all available information, outside of which it is desired to assume nothing about what is unknown (Berger *et al*, 1996). In MEP, the probability distribution represents information, not just frequencies. Below we describe several ways to extract data for the MEP for both discrete and continuous distribution.

6.3.3.1 Discrete distribution

Consider the case when an expert gives some information about a simple constraint, e.g. $p_1 + p_2 = 0.3$ for a discrete distribution. Then the distribution (pmf) with ME is that

$$p_1 = p_2 = 0.15 \text{ and the rest } p_i = \frac{1-0.3}{n-2} \ (i = 3,4,...n) \tag{6.7}$$

Alternatively, if provided information on the mean values F_k of certain function $f_k(x)$ of data, then this information can be expressed as m constraints:

$$\sum_{i=1}^{n} \Pr(x_i \mid I) f_k(x_i) = F_k \qquad k = 1,...,m \tag{6.8}$$

where $\Pr(x_i \mid I)$ denotes the probability for each possible state i given the information I. The entropy is $H = -\sum_{i=1}^{m} p_i \ln p_i$ which is proposed by Shannon (1948). As per MEP, we obtain:

$$\Pr(x_i \mid I) = \frac{1}{Z(\lambda_1,...,\lambda_m)} \exp[\lambda_1 f_1(x_i) + ... + \lambda_m f_m(x_i)] \tag{6.9}$$

where $Z(\lambda_1,...,\lambda_m)=\sum_{i=1}^{n} \exp[\lambda_1 f_1(x_i)+...+\lambda_m f_m(x_i)]$, and λ_k parameters are Lagrange multipliers whose values are determined by $F_k = -\frac{\partial}{\partial \lambda_k} Z(\lambda_1,...,\lambda_m)$. Further details of the procedure of MEP can be found in Jaynes (1963).

6.3.3.2 Continuous Distribution

For continuous distributions, the entropy is measured by $H_c = -\int p(x)\ln p(x)dx$, as proposed in Jaynes (1963). If we have some prior information, we can incorporate the information into the following constraints: $\int p(x)f_k(x)dx = F_k$. The MEP is presented as follows:

Maximize: $H_c = -\int p(x)\ln p(x)dx$

Subject to: $\int p(x)dx = 1$

$\int p(x)f_k(x)dx = F_k$

The solution to this MEP problem is

$$p(x)=\frac{1}{Z(\lambda_1,\lambda_2,...,\lambda_\mathrm{m})}\exp[\lambda_1 f_1(x)+...+\lambda_m f_m(x)] \tag{6.10}$$

where $Z(\lambda_1,...,\lambda_m)=\int \exp[\lambda_1 f_1(x)+...+\lambda_m f_m(x)]dx$ is used as normalization constant, and the value of λ_k is determined by the constraints according to $F_k = -\frac{\partial}{\partial \lambda_k} Z(\lambda_1,...,\lambda_m)$ (Jaynes, 1963).

For example, suppose the expert's prior opinion about the lifetime is positive ($x \geq 0$) and the mean value of the lifetime is a constant *c*. Then given this information, we have *f*(*x*)=*x* and *F_k*(*x*)=*c*. Solving Eq. (6.10) for the maximum entropy gives $p(x) = \frac{1}{Z(\lambda)}\exp[\lambda f(x)]$. As $F_k(x)=c$, $Z(\lambda)$ is an exponential function of *c* which is substituted into *p*(*x*). Then by considering the first constraint, we get that *p*(*x*) is an exponential distribution with an intensity $\lambda = \frac{1}{c}$. Incidentally, exponential distributions are mostly used in software reliability models (Xie *et al.*, 2004).

6.3.3.3 Some Examples

We illustrate the use of the data extraction method for the MEP described above with some examples in software reliability where expert suggestions and historical data are available.

Suppose the mean μ and variance σ^2 of a random variable *X* are known. By definition, $\int_{-\infty}^{+\infty}(x-\mu)^2 p(x \mid I) = \sigma^2$. Comparing this with Eq. (6.8), we obtain: $f(x)=(x-\mu)^2$, and $F_k=\sigma^2$, so the probability distribution with the maximum entropy is given as

$$p(x) = \exp\left[-1+\lambda_0+\lambda_1(x-\mu)^2\right] = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{6.11}$$

For more details about the derivation of Eq. (6.11), please refer to Kapur (1989).

MEP can further model the correlation among variables and construct joint distributions using lower order assessments (Abbas, 2006). If more information on the pairwise assessment among variables is known, the maximum entropy joint distribution can be extended. For example, the maximum entropy joint distribution of four variables is: $p^*_{i,j,k,l} = \arg\max_{p_{i,j,k,l}} \left(-\sum_{i,j,k,l} p_{i,j,k,l} \ln\left(p_{i,j,k,l}\right) \right)$. The joint probability distribution is then determined by the information of two-way joint assessments and three-way joint assessments (Abbas, 2006).

Sometimes, prior knowledge is likely to be given as inequality constraints such as $a_i \le p_i \le b_i$. It is often easier for the system managers to give a range on the probability than to predict a specific point-value. For example, they may only know the probable ranges of some parameters according to knowledge of previous similar products, from which they can easily set upper bound and lower bound as the maximum and minimum values in history.

This problem can then be solved from the MEP as:

Maximize: $-\sum_{i=1}^{n} p_i \ln p_i$

Subject to: $\sum_{i=1}^{n} p_i = 1, \quad p_i \ge 0$

$$\sum_{i=1}^{n} p_i f_k(x_i) = F_k \qquad k = 1,\dots,m$$

$$a_i \le p_i \le b_i,\ a_i \ge 0,\ b_i \le 1 \tag{6.12}$$

Kapur (1989) presented some algorithms to solve the above MEP.

6.3.4 Non-informative priori

Nevertheless, it is also possible that in some cases, no prior information exists. For example, it may be the first time a new group develops/tests a component so no historical data is available, and the component and group members are new so no past information or experience is applicable. However, the lack of useful prior knowledge will not affect the generality of the proposed Bayesian approach. The simplest way without useful information is to use a non-informative prior distribution (Bernardo and Smith, 1994).

Jeffrey’s non-informative prior (Robert, 1994) is one of the well known distributions. The Jeffrey prior was designed to solve the invariance under parameter transformations problem. According to the Jeffrey principle the following equation should hold:

$$p(\theta) \propto \sqrt{I(\theta)} \tag{6.13}$$

where $I(\theta)$ is the Fisher information for the parameter θ (Bernardo and Smith, 1994).

For example, the joint priori-distribution of the two parameters of the normal distribution, according to the Jeffrey’s principle, is

$$p(\mu, \sigma) \propto \frac{1}{\sigma^2} \tag{6.14}$$

The main idea here is to have a prior which contains no information on $\vec{a}$. Then, we can get the posterior distribution as Eq. (6.2).

6.3.5 Measures for Uncertainty

After deriving the priori distribution from MEP, and observing failure times $\vec{s}$, the posterior distribution can be obtained by Eq. (6.2). Then, the marginal density function with respect to each parameter can be obtained by

$$p_i(a_i \mid \vec{s}) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty} p(\vec{a} \mid \vec{s}) \cdot d(a_1)\cdots d(a_{i-1})d(a_{i+1})\cdots d(a_m)\,, \quad (i=1,2,\ldots,m) \qquad (6.15)$$

and the mean value of the corresponding parameter can be obtained by

$$\hat{a}_i = E(a_i) = \int_{-\infty}^{+\infty} a_i \cdot p_i(a_i \mid \vec{s}) \cdot d(a_i)\,. \quad (i=1,2,\ldots,m) \qquad (6.16)$$

The mean value can serve as a point estimate for the unknown parameter. Alternative Bayesian estimators are the maximum a posterior (MAP) of the parameters (Bernardo & Smith, 1994).

The uncertainty of the estimated parameters can be described by the variances and confidence intervals. The variance of the estimated parameter $\hat{a}_i$ is computed by

$$\sigma^2(a_i) = \int_{-\infty}^{+\infty} (a_i - \hat{a}_i)^2 \cdot p_i(a_i \mid \vec{s}) \cdot d(a_i) \qquad (6.17)$$

To compute the confidence interval for the parameter $\hat{a}_i$, suppose the confidence probability is β_i and the lower bound and upper bound of the interval are (low_i, up_i). Here we adopt the Highest Posterior Density (HPD) credible set based on the posterior distribution (DeGroot & Schervish, 2002)) to derive the narrowest confidence interval as

$$\textbf{Minimize } (up_i - low_i) \text{ Subject to } \int_{low_i}^{up_i} p_i(a_i \mid \vec{s}) \cdot d(a_i) = \beta_i \qquad (6.18)$$

Eq. (6.18) is numerically solvable in principle. Then, we state with β_i confidence, the exact value of the parameter a_i is located within the interval. Similarly, from the posterior joint distribution of Eq. (6.2), we can derive the credible set of the parameters using the HPD given the credible-level β. Denote C_β as a range of the n dimensions for $\vec{a}$. The HPD derives

$$C_\beta^{\ *} \text{ that maximizes the integral of } \oint_{C_\beta} p(\vec{a} \mid \vec{s}) d\vec{a} \text{ equal to } \beta \tag{6.19}$$

There are several other ways to obtain the confidence intervals such as the symmetric intervals suggested by Yin & Trivedi (1999), where the upper bound and lower bound are symmetric to the mean. Though this symmetric interval method is simple and straightforward, it cannot guarantee the narrowest interval unless the distribution is symmetric as well. In contrast, the HPD provides the narrowest confidence interval.

6.3.6 Monte Carlo Approach for System Uncertainty

In the previous subsections, we have analyzed the parameter uncertainty of reliability model for one component based on the MEP and BA. A complicated system (software) may contain multiple components (modules). As introduced in the subsection 6.2.2, many tools can be implemented to evaluate the system reliability given the parameters of its contained components, such as the Markov models, Bayesian Network, Graph Theory, Fault-Tree Analysis etc. Regardless of the tools used, the system reliability is a function combining the parameters of its components. As a result, the uncertainties in the parameters affect the whole system reliability.

The purpose of this section is to study and quantify the uncertainty in the reliability of the complex system due to the uncertainty of the parameters in the numerous components of the system. Some general assumptions of the system-level analysis are listed below.

1) Suppose the system contains a total K components that are *statistically*-independent.

2) Each component has its own reliability model and let Λ_i denote the set of parameters for the i^{th} component's model, where i=1,2,…,K.

3) For each component, the probability distribution of its model's parameters is known, which can be derived from the above section 6.3.2. Denote the posterior joint distribution for the parameters of the i^{th} component by $p_i(\Lambda_i)$, i=1,2,…,K.

4) Given the failures of different components are s-independent, the system reliability can be expressed by the function of components' parameters, as $R_s = f(\Lambda_1, \Lambda_2, ..., \Lambda_K)$.

Based on the above assumptions, a Monte Carlo simulation is presented for generally analyzing the uncertain system reliability. It is difficult to use analytic methods to combine the distributions of numerous parameters to derive the probability density function of the system reliability, especially for complicated systems with complex architecture and many components. Hence, the Monte Carlo simulation becomes a practical way to make the uncertainty analysis of the complicated system tractable. Algorithm 1 provides a general Monte Carlo approach for the uncertainty analysis in complicated system.

Algorithm 1: Monte Carlo approach

```
1. begin
2. for j=1 to J            //J is the total number of the iteration
3.    for k=1 to K             //Generate the parameters for all the K
                               //components
4.       $P_k \leftarrow SAMPLE(p_k(\Lambda_k) | \vec{s})$
            //The function of $SAMPLE(p_k(\Lambda_k) | \vec{s})$ is to draw a sample of the
            //parameters from the posterior pdf $p_k(\Lambda_k | \vec{s})$, and then put the value
            //into the vector $P_k$.
5.    end
6.    $R_s[j] \leftarrow f(P_1, P_2, ..., P_K)$     //function $f(P_1, ..., P_K)$ computes system
                               //reliability
7. end     //Now, J sample points of the system reliability are saved in $R_s$.
8. $Summarize(R_s)$      //function $Summarize(R_s)$ calculates some statistics for
   //uncertainty analysis from the sample points of $R_s$, such as average, variance,
   //quantile and so on.
9. end (*Algorithm 1*)
```

Using the above algorithm of Monte Carlo (MC) simulation, the uncertainty of the system reliability can be analyzed, e.g. the mean and confidence intervals can be approximated by the average value and percentiles, respectively. The above approach is widely applicable. Some cases will be studied in the next section, where a novel model based on the uncertainty approach will be presented for large-scale system reliability in subsection 6.4.3.

6.3.7 Information Filtering, Adjustment and Validation for MEP

As we discussed at the end of the above subsection 6.3.3, the MEP can incorporate not only objective information but also subjective information into the Bayesian Approach. Objective information is more credible (such as the information that failure time $t \geq 0$) than some subjective information (such as the experts' opinions) that might be wrong or deviate too much from the reality. Wrong information can be worse than no information. Nevertheless, the correct subjective information is obviously helpful. Therefore, this section attempts to complement the MEP approach to reduce the chance of incorporating the wrong information.

Three stages are proposed in this framework: 1) Information Filtering at the beginning; 2) Information Adjustment during the test; 3) Information validation after the test.

6.3.7.1 Information Filtering

Information filtering means that the collected information needs to be checked before it is used to formulate the constraints as the basis of the MEP. Objective information (without any subjective influence) can be directly included. However, subjective information (such as the suggestions from the experts) should be checked out. We propose to prepare a survey form for the experts to fill when their suggestions are collected. This form includes not only the suggestions but also confidence levels associated with the corresponding suggestions. However, we cannot simply use the ranking of confidence levels to filter different experts' opinions, because some experts may be conservative while some others may be aggressive or neutral.

Therefore, the following method is suggested. First, the *credible degree* is defined as the probability for a certain expert's suggestion to be correct. Thus, we should set up a threshold of credible degree, denoted by $c*$. Then, each expert has previous records of their suggestions in the previous projects with the ranking of confidence levels. Suppose there are *K* ranks, then at each rank, the credible degree of the *i*-th expert (*i*=1,2,…,*N*) can be calculated as

$$c_i^k = \frac{\text{Number of correct suggestions at rank k in all prior projects of expert i}}{\text{Total number of suggestions at rank k in all prior projects of expert i}}, \quad k=1,2,\ldots,K \qquad (6.20)$$

Thus, regarding this expert *i*, those suggestions at the ranks of confidence levels where $c_i^k < c*$ should be filtered. Different experts have their own records, i.e. c_i^k ($i = 1,2,...,N$), so according to this criterion of credible degree, the different experts' opinions (no matter whether conservative or aggressive) can be filtered, which is more fair and more reasonable than simply using an absolute rank to filter the suggestions. The threshold value $c*$ could be initialized according to the specific requirement of the user, such as the minimum credibility the user trusts. The minimum credibility here means that the threshold of credible degree can be increased after the adjustment steps as follows.

6.3.7.2 Information Adjustment

After the above step of filtering the information, the MEP can be used to derive a prior distribution. Hereby, we propose the second step to further check and adjust the information during the test on the current software. The newly observed data can be utilized as per this step.

The prior distribution is not only a factor in the Bayesian formula, but also a prediction of the current project/software. If the prior distribution is much different from the real tendency of the newly observed data, it means the prior distribution might be inappropriate for the current project/software. Therefore, adjustment should be carried out according to this criterion. The details are elaborated in the following steps.

Step 1: There is a prior distribution $p(x)$ with respect to a certain parameter, and given a set of newly observed data. Also, set up a threshold for adjustment, which is a probability α.

Step 2: With the newly observed data, estimate the parameter (such as using MLE), x'.

Step 3: Derive $x_{\alpha/2}$ by solving $\int_{-\infty}^{x_{\alpha/2}} p(x)dx = \frac{\alpha}{2}$ and $x_{1-\alpha/2}$ by solving $\int_{-\infty}^{x_{1-\alpha/2}} p(x)dx = 1-\frac{\alpha}{2}$.

Step 4: If $x' < x_{\alpha/2}$ or $x' > x_{1-\alpha/2}$, the adjustment should be triggered. It means that the estimated parameter x' is located at the two extreme tails of the prior distribution, which is out of the middle interval with the probability $1-\alpha$.

Step 5: Do the adjustment (such as increasing the credibility degree $c*$ for information filtering and then recalculate the prior distribution with the newly filtered information via MEP), and then repeat the Step 1.

The worst case is that $c*$ finally increases to 1, which means all information has to be filtered. Under this condition, the above steps should be terminated, and then switched to the non-informative prior as suggested in the above subsection 6.3.4. The value of α can be set up in accordance with the user's requirement of credibility degree, such as the probability for the estimated parameter not to be at either tails of the prior distribution.

6.3.7.3 Information Validation

Finally, when the test finishes, the posterior distributions can be derived based on the MEP and Bayesian Approach. Then, we can validate the posterior distribution/model by using the following method. The mean values of the posterior distributions can be used to derive the parameters, as Eq. (6.16). Then, these parameters can be applied to predict the time to failures (TTFs) during the test. Compare the predicted TTFs with the real observed TTFs, such as calculating the mean square error. If the mean square error is too large over a certain preset threshold, it means the model does not fit the observed data. Then, the adjustment (such as trying another model or further filtering the subjective information) should be applied.

Thus, the above three stages can help to reduce the negative influence of the wrong subjective information as a complementation to the MEP. However, note that we propose a three stage framework (information filtering, information adjustment, and information validation), but the specific methods described in each stage can vary. Developing other methods for each stage are areas for future research.

6.4 Case Study

The above uncertainty analysis is applicable not only to one single component with correlated parameters, but also to a system with multiple components. The proposed approaches are general enough to accommodate different types of models, as illustrated here. Subsection 6.4.1 illustrates an NHPP model for software reliability. Subsection 6.4.2 illustrates a Markov model that contains three components, and finally a new model and more complicated case of large-scale distributed system are studied in subsection 6.4.3. Note, the numerical examples in this section are exhibited only for the illustrative purpose on the general approach presented in above Section 6.3.

6.4.1 Component Uncertainty of an NHPP Model

The Software Reliability Growth Model (SRGM) based on Nonhomogeneous Poisson Process (NHPP) is commonly used. Here, we illustrate a classical NHPP model presented by Goel & Okumoto (1979). In the Goel-Okumoto (GO) model, the mean value function is given by

$$m(t) = a[1 - \exp(-bt)], \; a > 0, b > 0 \tag{6.21}$$

and the failure intensity function is

$$\lambda(t) = \frac{d}{dt} m(t) = ab\exp(-bt) \tag{6.22}$$

in which there are two parameters *a* and *b*.

The observation of failure data set is from a simulation of the GO model where the preset parameters $a = 100$ and $b = 0.001$. From the simulation, 50 points of Time to Failure (TTF) are collected as shown in the following Table 6.1 in unit of hour. For the details of the simulation with the GO model, please refer to Xie *et al.* (2004).

Table 6.1 50 Time to Failure (TTF) from a simulation of the GO model

1-10	7.51157	15.166	20.238	27.601	38.784	51.58	69.674	81.187	83.32	92.128
11-20	97.451	100.02	100.92	114.28	128.55	133.75	134.52	145.51	155.71	167.01
21-30	180.01	192.49	214.99	251.13	286.51	297.55	319.3	326.25	344.28	367.07
31-40	427.43	431.91	444.38	445.48	457.4	471.16	473.83	494.18	510.08	516.7
41-50	519.66	585.74	592.64	610.16	613.55	626.25	632.67	648.61	671.35	713.41

By using the MLE in the Table 6.1, a and b are estimated as $\hat{a} = 89.508$ and $\hat{b} = 0.00115$ both of which have more than 10% error from the preset real values. Then, we implement the Bayesian Approach with MEP to analyze the same data set.

6.4.1.1 BA with MEP

First, suppose there is knowledge of some statistics archived from previous projects or other similar projects, the mean of a is μ_a=100 and the standard deviation is σ_a=10 while the mean of b is μ_b=0.001 and the standard deviation σ_b=0.0001.

By using the MEP (Kapur, 1989) as Eq. (6.11), we can get the priori distribution for a~$N(100,10^2)$ and b~$N(0.001, 0.0001^2)$, respectively. Thus, the prior joint distribution satisfies

$$p(a,b) \propto \frac{1}{2\pi\sigma_a\sigma_b}\exp\left(-\frac{(a-\mu_a)^2}{2\sigma_a^{\ 2}}-\frac{(b-\mu_b)^2}{2\sigma_b^{\ 2}}\right) \tag{6.23}$$

Note that Eq. (6.23) is just used for this example as an illustration, and other prior distributions can also be derived from MEP according to their own specific conditions.

Then, according to Eq. (6.2), we get the posterior distribution

$$
\begin{aligned}
p(a,b\,|\,\vec{s}) &\propto p(a,b)\cdot p(\vec{s}\,|\,a,b) \\
&\propto \frac{1}{2\pi\sigma_a\sigma_b}\exp\Big(-\frac{(a-\mu_a)^2}{2{\sigma_a}^2}-\frac{(b-\mu_b)^2}{2{\sigma_b}^2}\Big)\cdot p(\vec{s}\,|\,a,b) \\
&\propto \frac{1}{2\pi\sigma_a\sigma_b}\exp\Big(-\frac{(a-\mu_a)^2}{2{\sigma_a}^2}-\frac{(b-\mu_b)^2}{2{\sigma_b}^2}\Big)\exp\{-a[1-\exp(-bs_{50})]\}a^{50}b^{50}\exp(-b\sum_{i=1}^{50}s_i) \\
&= 1.5032\times10^{81}a^{50}b^{50}\exp\left(-\frac{(a-100)^2}{200}-\frac{(b-0.001)^2}{2\times10^{-8}}-a[1-\exp(-713.41b)]-15431b\right)
\end{aligned}
\tag{6.24}
$$

where 1.5032×10^{81} is obtained by substituting the values of parameters into Eq. (6.24). Then, according to Eq. (6.15), the marginal density functions with respect to a and b can also be obtained as shown by Figure 6.1 and Figure 6.2, respectively.

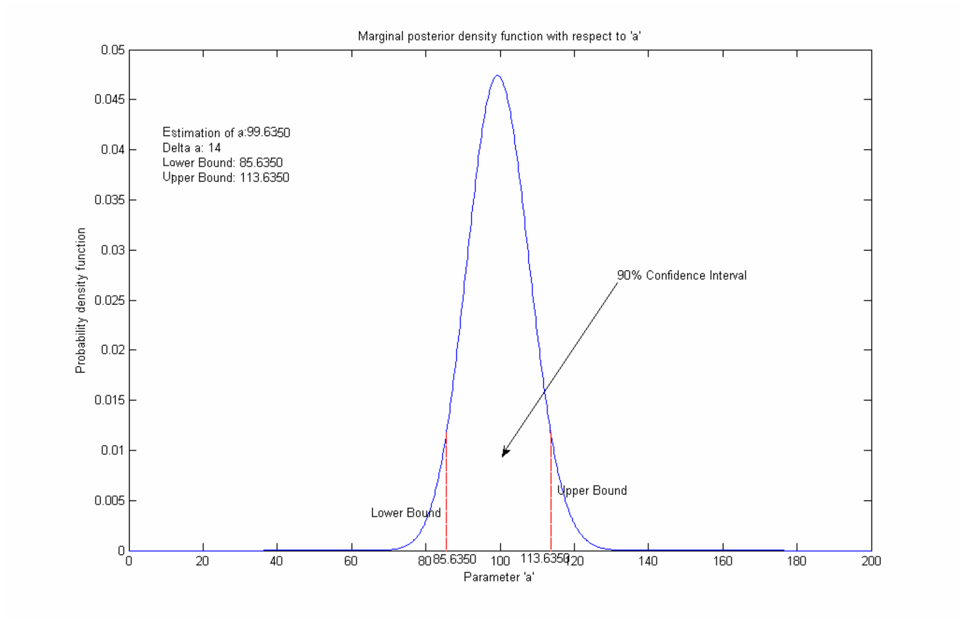

Figure 6.1 Marginal posterior density function with respect to *a*

Then, solving Eq. (6.18), the 90% HPD intervals for a and b can be obtained as (85.64,113.64) and (0.00085, 0.00115), depicted in Figure 6.1 and Figure 6.2 as well. The mean values for a and b are estimated as $\hat{a}=99.64$ and $\hat{b}=0.001$ with only 0.4% error from the preset parameters. It is better than those from the MLE with 10% error. Also, the 90% confidence interval offers more sound ranges than point values from MLE.

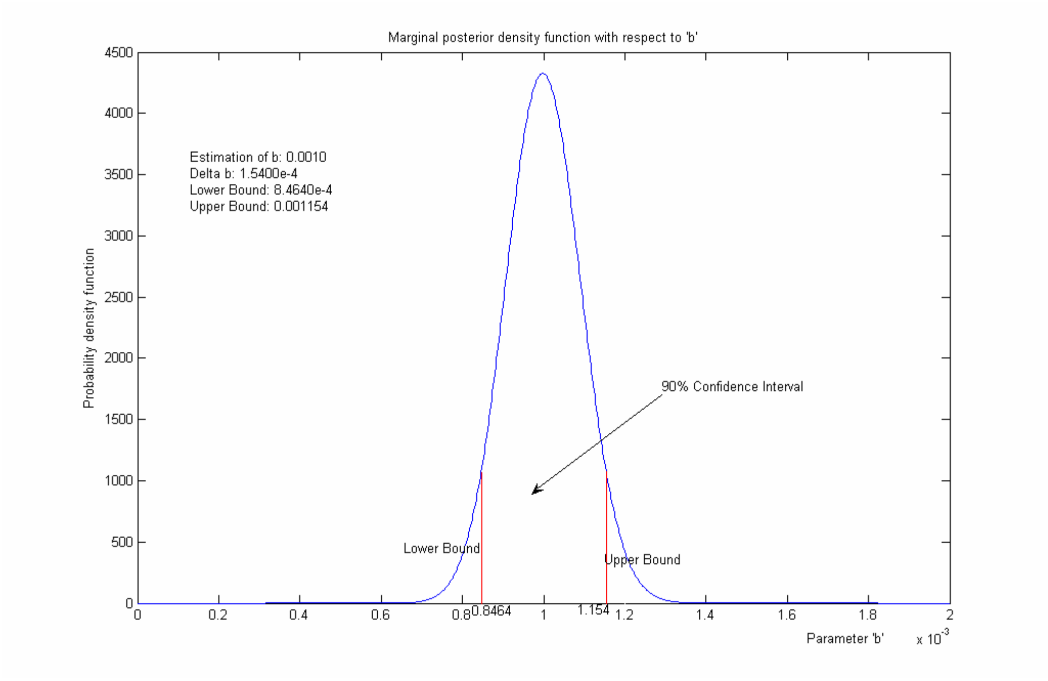

Figure 6.2 Marginal posterior density function with respect to *b*

The reliability function of the component with the GO model can be computed by

$$R(t \mid s_n) = \exp\{-[m(t+s_n) - m(s_n)]\} \tag{6.25}$$

where s_n is the time of the last failure and *t* is the time measured from the last failure.

The reliability prediction from the MLE and the posterior mean obtained by Eq. (6.16) are compared in Figure 6.3. As we can see, both point estimate methods of MLE and posterior mean can predict a close reliability trend, so the new method using the posterior mean can be an alternative way for the point estimate of the parameters. More importantly, using the posterior probability distribution, we can further analyze the uncertainty of the predicted reliability, e.g. the 90% HPD interval of the reliability prediction is also shown in Figure 6.3. In addition, the confidence intervals depend on both the prior distribution and the new observations. This approach can also help evaluate the quality of the estimation, as the wider range of the confidence interval indicates a worse estimation while the narrower range indicates a better one.

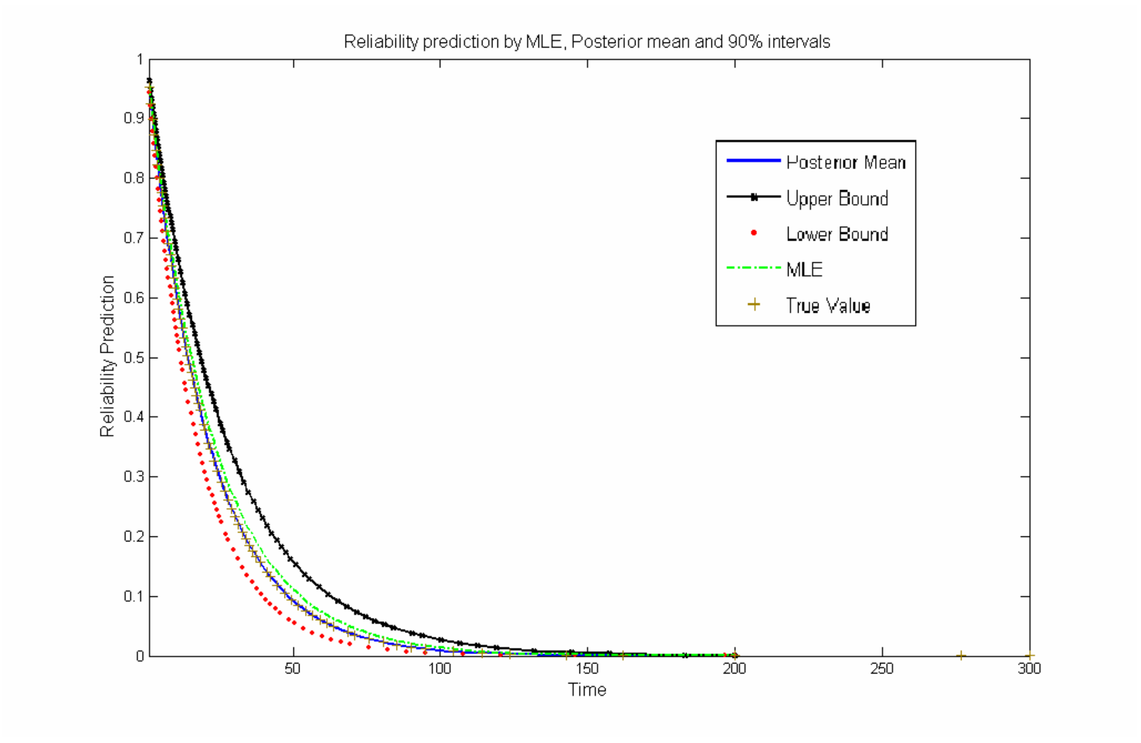

Figure 6.3 Reliability prediction with MLE, Posterior Mean and 90% interval

6.4.1.2 BA with Jeffreys' non-informative Priori

Suppose there is no priori knowledge or expert suggestions, then according to Jeffreys' non-informative Principle given by Eq. (6.15), we have the following

$$p(a,b) \propto \frac{1}{\sigma_a{}^2 \sigma_b{}^2} \tag{6.26}$$

According to Eq. (6.15), the marginal density functions under non-informative prior with respect to *a* and *b* are plotted by Figure 6.4 and Figure 6.5, respectively.

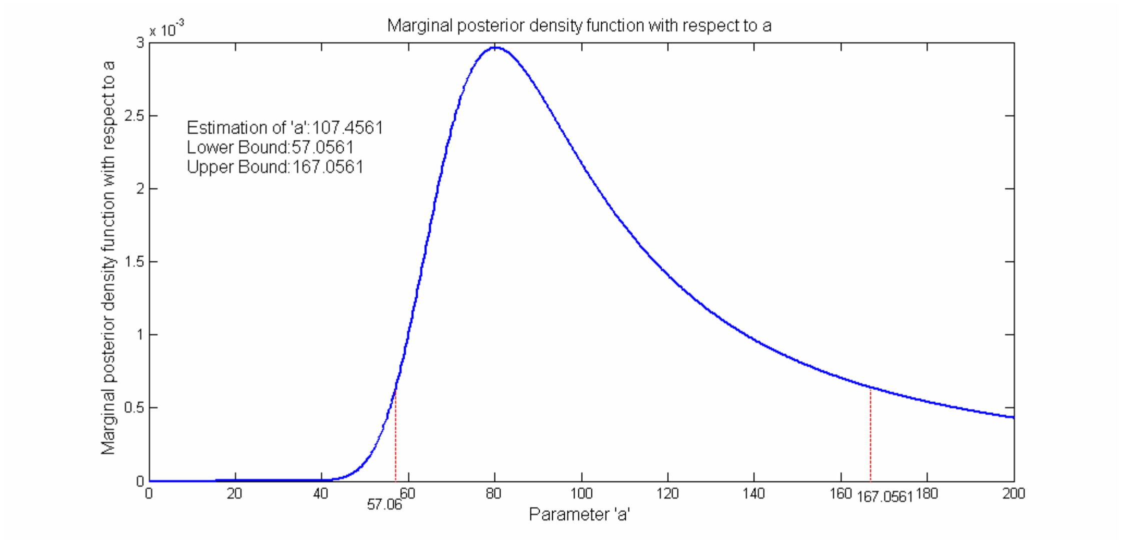

Figure 6.4 Marginal posterior density function regarding a under noninformative prior

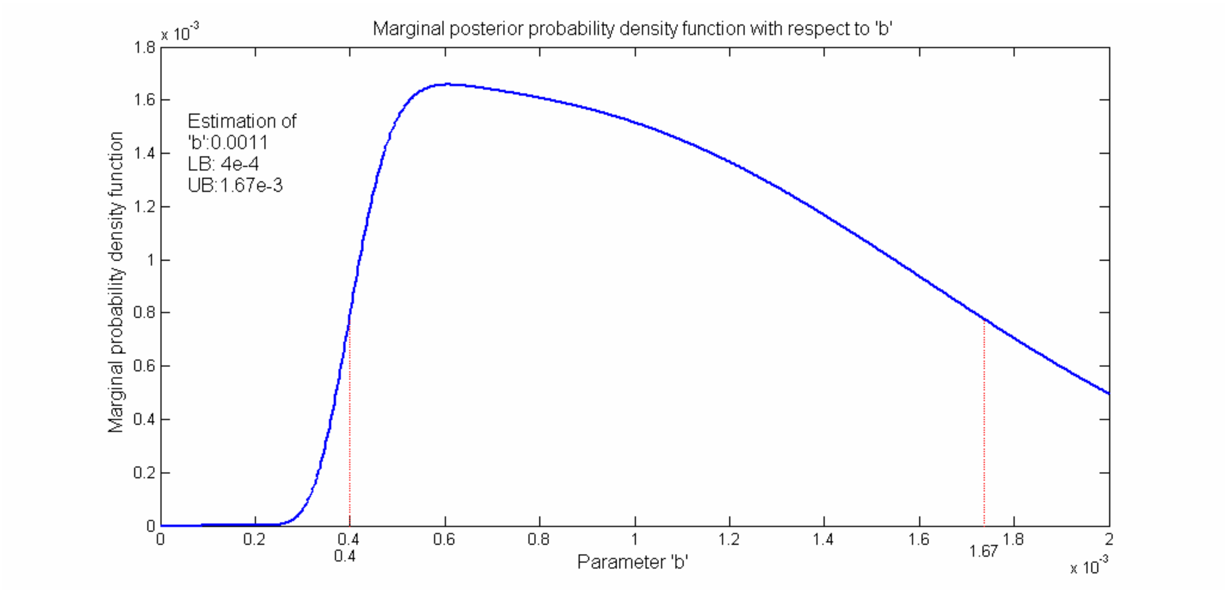

Figure 6.5 Marginal posterior density function regarding b under noninformative prior

The mean value for a and b are estimated as $\hat{a} = 107.46$ and $\hat{b} = 0.0011$ according to Eq. (6.16). To derive the 90% confidence intervals for a and b, HPD is adopted according to Eq. (6.18). The intervals are (57.06, 167.06) and (0.0004,0.00167) for a and b respectively.

In Figure 6.6, the reliability prediction by MLE and the posterior mean are plotted and compared (*True value* in Figure 6.6 represents the reliability prediction by the preset parameters which are a=100 and b=0.001). The 90% confidence interval by

Bayesian method, which is obtained according to Eq. (6.19), is also depicted in Figure 6.6. All the plots are in unit of hour in Figure 6.6.

Comparing Figs. 4,5,6 with Figs. 1,2,3, it is obvious that the MEP that combines the prior knowledge and other information into the BA is more precise in the modeling and estimate, as reflected in the narrower confidence interval than the non-informative prior.

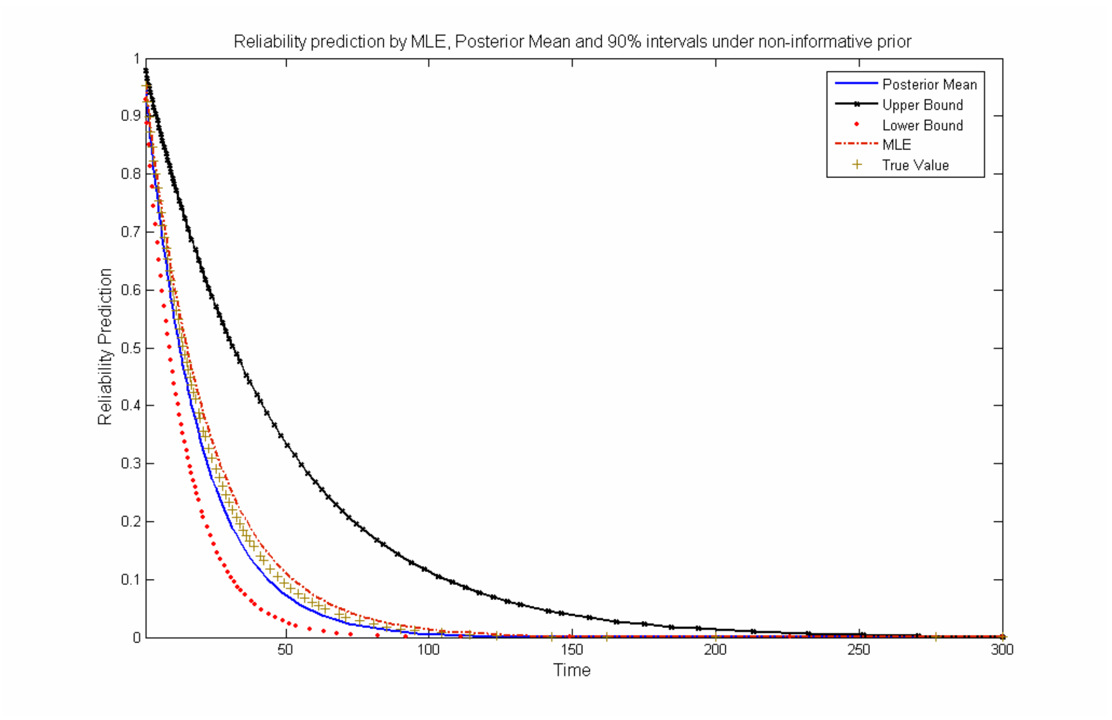

Figure 6.6 Reliability with True Value, MLE, Posterior Mean and 90% Interval

6.4.2 Case Study on Markov Models

An example of modular software is illustrated here to show the Monte Carlo Simulation for uncertainty analysis of the software reliability with multiple modules. This example is based on a simple Markov model.

Suppose the software contains three modules: two parallel modules to fulfill the same function and another module to handle the switch between the two parallel modules. The two parallel modules are running to finish a same task respectively with the failure rate of λ. If a module fails, the switching module will work and transfer the workload of the failed module to the other module. A coverage factor c is used to denote the probability that the switching action is successful. If not successful, the software fails, denoted as the imperfect coverage. Otherwise, the software is still running, while the other module failure will make it failed, or the restart of the failed module will bring the software back to the original. Also, suppose the time to restart a module is exponentially distributed with the parameter μ. Thus, the parameters for the two modules are failure rate λ and restart rate μ, and the parameter for the switching module is the switching success probability c.

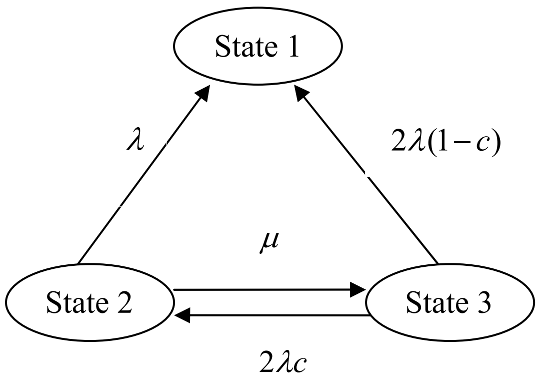

Figure 6.7 Markov chain for the modular software with three modules

The software reliability can be derived from a Markov model that easily combines all the three modules together. The CTMC is depicted by Figure 6.7 where state 1 is down and system is up in state 2 (one module works) and 3 (two modules work). The software initially begins at state 3. If either one of the two modules fails with the rate 2λ, it leaves State 3 to State 1 with the probability $(1-c)$ due to the switching failure, and to state 2 with the probability of c for successful switching. At

State 2, it can enter state 1 if the remaining module fails with the failure rate λ, while it can return to State 3 with the repair rate μ to recover the failed module.

Let $P(t) = \{P_1(t), P_2(t), P_3(t)\}$ denote the state probability. The *Chapman-Kolmogro*v equations, see e.g. Xie & Dai (2004), can be obtained from Figure 6.7:

$$P_2'(t) = -(\lambda + \mu)P_2(t) + 2\lambda c P_3(t)$$

$$P_3'(t) = \mu P_2(t) - 2\lambda P_3(t)$$

with the initial condition: $P_2(0) = 0, P_3(0) = 1$. Then, the software reliability can be obtained by

$$R_s(t) = P_2(t) + P_3(t) \tag{6.27}$$

In this example, we assume these parameters (λ, μ and c) are independent. Suppose that the distributions of the three parameters have been derived from the BA plus MEP given in the above section 6.4.1 for respective components, and they are assumed as: 1) $\frac{1}{\lambda}$ follows a normal distribution with mean 5000 and standard deviation 500; 2) $\frac{1}{\mu}$ follows a normal distribution with mean 20 and standard deviation 2; 3) c follows a normal distribution with mean 0.8 and standard deviation 0.05.

We apply the Monte Carlo approach given by the *Algorithm 1* to simulate 1000 sample points of system reliability at each of the 5000 time points, and the final results of software reliability and uncertainty analysis (including sample average, 5% and 95% quantiles) are shown in Figure 6.8. Then, the analytic method is implemented for this problem and the uncertainty analysis result with the mean value is also plotted in Figure 6.8, where 'MC' denotes 'Monte Carlo' and 'AM' means 'Analytic Method' from Eq. (6.27).

From the results of uncertainty analysis in Figure 6.8, we find that during the initial period of the reliability prediction, the confidence interval is small indicating that the uncertainty of the software reliability is low. Then, the confidence interval increases and reaches the maximum around the middle part. At the latter part, the confidence interval becomes small again but the mean value of system reliability is also small so that the comparative uncertainty is still large. We also observe from those curves in Figure 6.8 that the sample average from the Monte Carlo Simulation is very close to the mean calculated by the Analytic Method of Eq. (6.27).

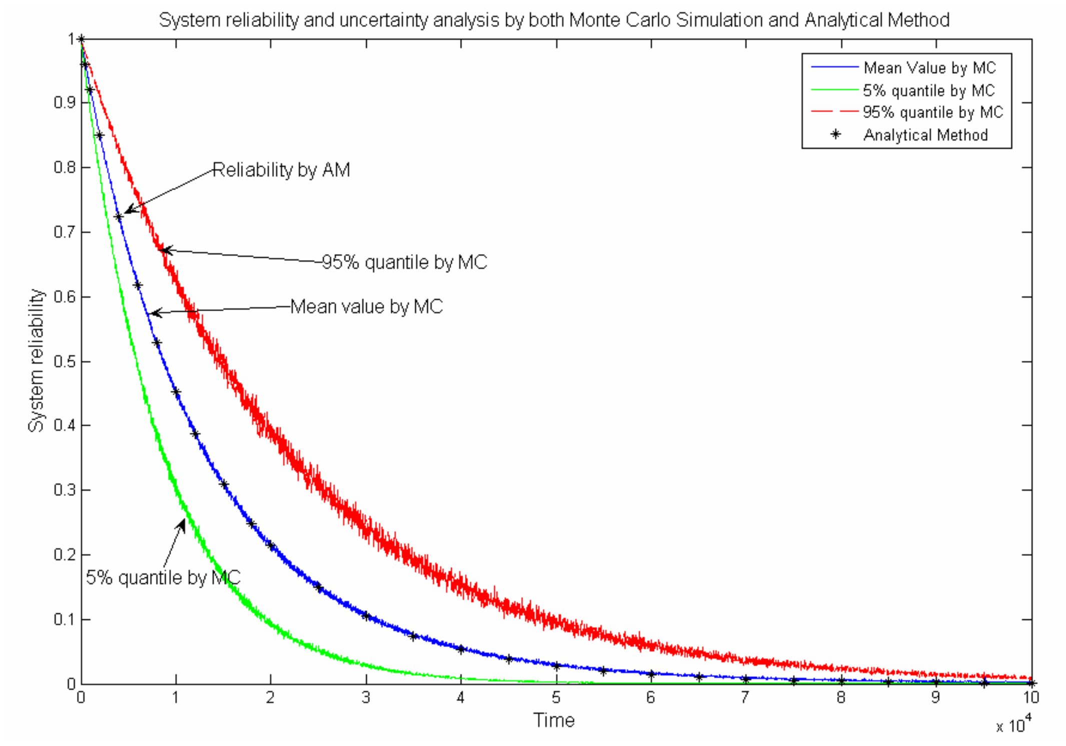

Figure 6.8 Modular software reliability and uncertainty analysis

6.4.3 Improved Model on Large-Scale System Reliability

The above three-component Markov model is relatively simple for illustrative purpose. Here, we study a more complicated case with regard to the large-scale system in order to show the generality and efficiency of the Monte Carlo approach. Although this model is illustrated to show how to incorporate the uncertainty, more importantly, through this incorporation, it improves the original model of Dai & Levitin (2006) by relaxing several impractical assumptions.

6.4.3.1 The Model based on Graph Theory and Bayesian Theorem

Dai & Levitin (2006) presented a novel reliability model for Grid computing which is a new emerging technology aiming at large-scale resource sharing and global-area collaboration. This model is representative for large-scale software systems (Selby, 2005) where different modules or resources are distributed all over the Internet under the coordination of the RMS (Resource Management System). Dai & Levitin (2006)'s model is very general and is the first to make the modeling on large-scale system reliability tractable. A virtual tree structure is proposed to model the problem, where the root of the tree structure is the RMS, and the leaves are resources, while the branches of the tree represent the communication channels linking the leaves and the root.

However, the generality causes the complexity in the modeling, and the uncertainty becomes more prominent due to the largeness and dynamicity of the Internet. We hereby improve the large-scale system reliability model to be more realistic by considering the uncertainty factors. The original model is briefly introduced here. More details can be found in Dai & Levitin (2006).

The set of all nodes and links involved in performing the given task form a task spanning tree. This task spanning tree can be considered to be a combination of minimal task spanning trees (MTST), where each MTST represents a minimal possible combination of available elements (resources and links) that guarantees the successful completion of the entire task. The failure of any element in a MTST leads to the entire task failure.

For any subtask *j*, and any resource *k* assigned to execute this subtask, one has the amount of input and output data, the bandwidths of links, belonging to the corresponding paths γ_k, and the resource processing time. Thus, one can obtain the completion time , see Dai & Levitin (2006).

A MTST completes the entire task if all of its elements do not fail by the maximal time needed to complete subtasks in performing which they are involved. Thus, when calculating the element reliability in a given MTST, one has to use the corresponding record with maximal time.

Having the MTST, and the times of their elements involvement in performing different subtasks, one can determine the pmf (probability mass function) of the entire service time.

First, the conditional time of the entire task completion given only MTST S_i is available as

$$Y_{\{i\}} = \max_{1\le j\le h}(y_{ij}) \text{ for any } 1\le i\le N: \tag{6.28}$$

For a set ψ of available MTST, the task completion time is equal to the minimal task completion times among the MTST.

$$Y_{\psi} = \min_{i\in\psi}(Y_{\{i\}}) = \min_{i\in\psi}\left[\max_{1\le j\le h}(y_{ij})\right] \tag{6.29}$$

Now, we can sort the MTST in an increasing order of their conditional task completion times $Y_{\{i\}}$, and divide them into different groups containing MTST with identical conditional completion time. Suppose there are K such groups denoted by $G_1, G_2, ..., G_K$ where $1 \le K \le N$, and any group G_i contains MTST with identical conditional task completion times Θ_i ($0 \le \Theta_1 < \Theta_2 < ... < \Theta_K$). Then the probability $Q_i = \Pr(\Theta = \Theta_i)$ can be obtained as

$$Q_i = \Pr(E_i, \overline{E}_{i-1}, \overline{E}_{i-2}, ..., \overline{E}_1) \tag{6.30}$$

where E_i is the event when at least one of MTST from the group G_i is available, and $\overline{E}_i$ is the event when none of MTST from the group G_i is available.

Suppose the MTST in a group G_i are arbitrarily ordered, and F_{ij} (j=1,2,…, N_i) represents an event when the j-th MTST in the group is available. Then, the event E_i can be expressed by

$$E_i = \bigcup_{j=1}^{N_i} F_{ij} \, , \tag{6.31}$$

and (6.30) takes the form

$$\Pr(E_i, \overline{E}_{i-1}, \overline{E}_{i-2}, ..., \overline{E}_1) = \Pr(\bigcup_{j=1}^{N_i} F_{ij}, \overline{E}_{i-1}, \overline{E}_{i-2}, ..., \overline{E}_1) \, . \tag{6.32}$$

Using the Bayesian theorem on conditional probability, we obtain from (6.32) that

$$Q_i = \sum_{j=1}^{N_i} \Pr\left(F_{ij}\right) \cdot \Pr\left(\overline{F}_{i(j-1)}, \overline{F}_{i(j-2)}, ..., \overline{F}_{i1}, \overline{E}_1, \overline{E}_2, \cdots, \overline{E}_{i-1} \middle| F_{ij}\right). \tag{6.33}$$

The probability $\Pr(F_{ij})$ can be calculated as a product of the reliabilities of all the elements belonging to the j-th MTST from group G_i.

Then, the service reliability $R(\theta^*)$ is defined according to the performability concept as a probability that the correct output is produced in time less than θ^*. This index can be derived by

$$R(\theta^*) = \sum_{i=1}^{K} Q_i \cdot 1(\Theta_i < \theta^*). \tag{6.34}$$

More details can be found in Dai & Levitin (2006).

6.4.3.2 Model Improvement Considering Uncertainty

In these large scale systems, uncertainty is common due to the unexpected and dynamic natures of the Internet. The parameters in the above analytical model for large-scale system reliability are also dynamical, for instance communication speed and processing speed are not constant, but varied under different workloads or mutable bandwidths. The original model (Dai & Levitin, 2006) assumed the parameters of communication speed and processing speed are constant, which is just an approximation to reality. Here, we propose an improved version of the model that relaxes such approximated assumption to consider the uncertainty of the speed parameters, i.e. assuming the speed parameters themselves are random variables instead of the constants.

With this improved model, we study the same problem described as the Case D of Dai & Levitin (2006). Consider a grid service with nine resources. The complexity of the entire service task is C = 6000 mega operations (MO). The backbone structure of the service, and the parameters of the resources & communication links are depicted in Figure 6.9. The task is divided into three subtasks with equal complexity $c_1 = c_2 = c_3 = 2000$ (MO). The subtask distribution is: c_1 is assigned on resources R1, R4, R7; c_2 is assigned on resources R2,R5,R8; and c_3 is assigned on resources R3,R6,R9;. The amount of transmitted data for each subtask is 100 Kbits.

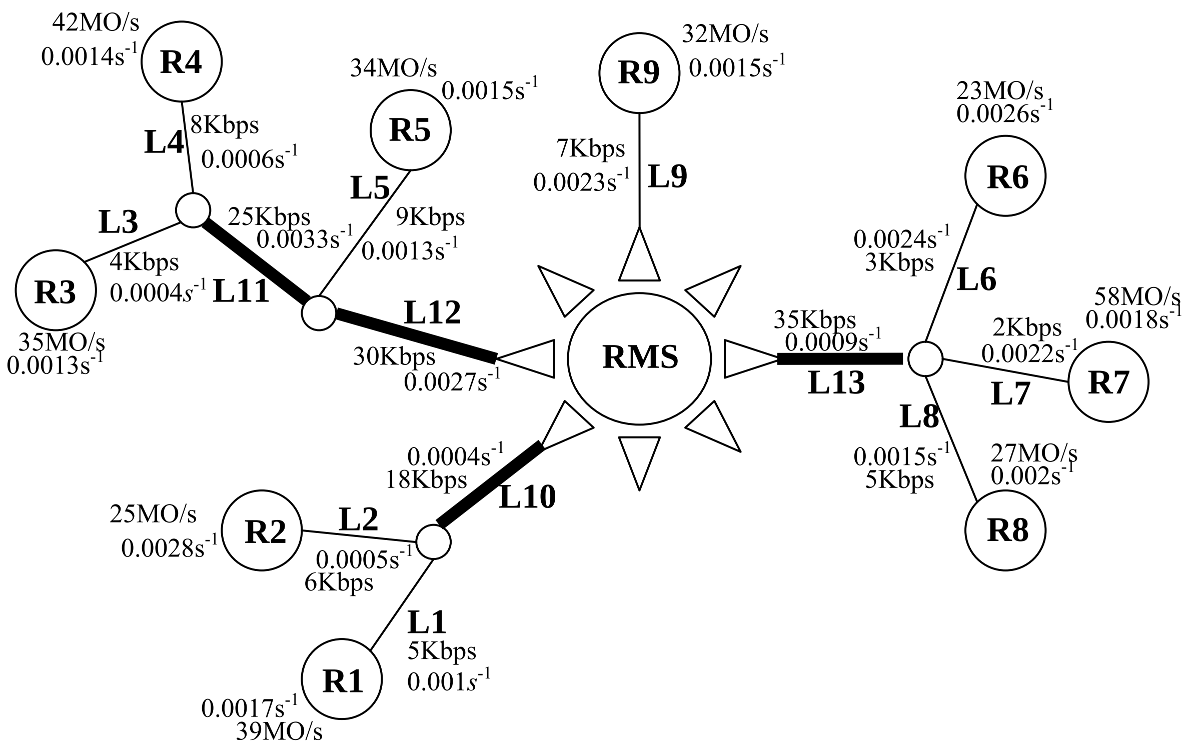

Figure 6.9 The structure and parameters of a Grid service

In Dai & Levitin (2006)'s analytical model, they assumed the estimated parameters for communication speed (Kbps) and processing speed (Mops) are not changing. Here, we relax this strict assumption and model the uncertainty using the proposed Monte Carlo approach. Suppose communication speed and processing speed are random variables governed by Normal distributions with means listed in Figure 6.9 and variance of 10.

The results are depicted in Figure 6.10 with the Original Model, Improved Model, and 5% and 95% percentiles. The variance is shown in Figure 6.11. These results are in unit of second.

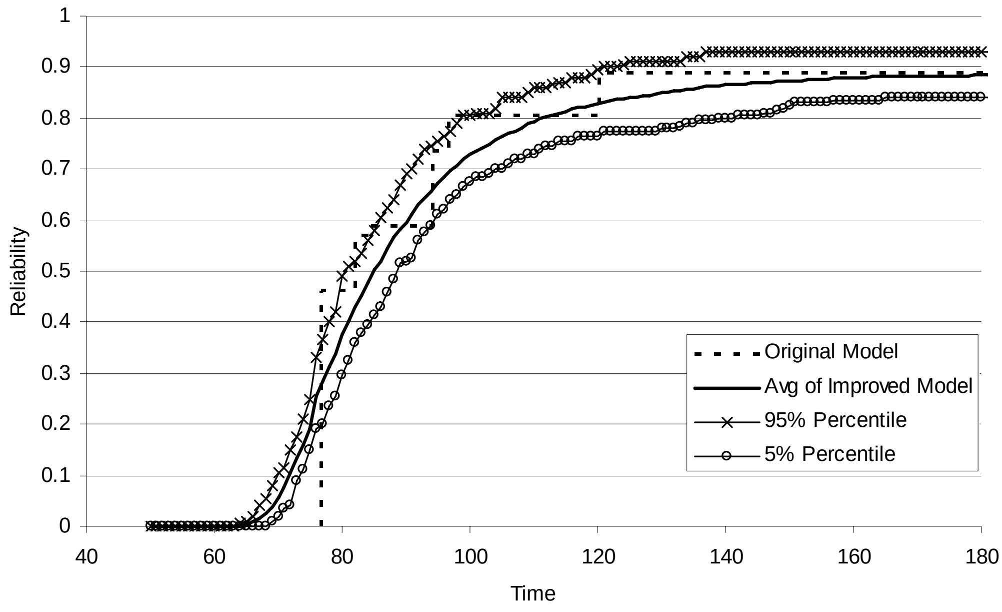

Figure 6.10 Improved model for Large-scale system reliability

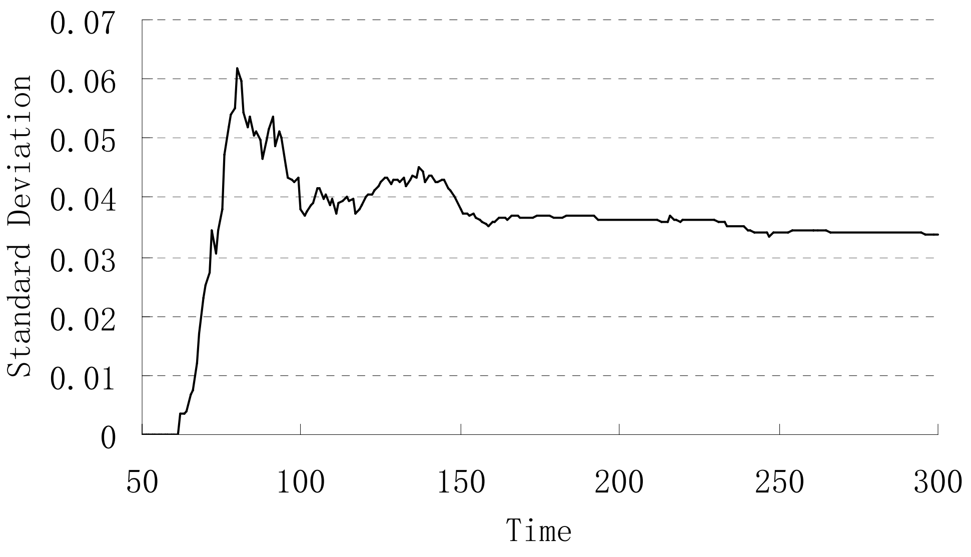

Figure 6.11 Standard Deviation for the model with uncertain parameters of speed

From Figure 6.10, we find that the original model of Dai & Levitin (2006) with constant speed assumptions is just an approximation to reality. The result of performance and reliability from the original model is plotted *Stepwise* while the improved model outputs a smoother curve. Although the original model with constant assumptions depicts the correct trend of the reliability, it is not as realistic as the improved model because intuition and experimental observations depict that the performance (execution time) or reliability should smoothly change instead of the abrupt jump. Therefore, the improved model which accounts for the uncertainty in the factors is more realistic and practical. Moreover, the improved model can provide another measure that the original model is unable to give, i.e. the percentiles that offer the confidence range to the practitioners about the model's output which is more credible than only one-point estimate. In addition, the original model is a special case of the improved model with variance equal to 0.

6.5 Summary

This chapter studied the uncertainty problems in reliability modeling at both component-level and system-level. It not only addressed the uncertainty problem using the Bayesian Approach (BA), but more importantly solved the challenges for the dearth of data by embedding the Maximum-Entropy Principle (MEP) into the BA. By using MEP with BA, expert knowledge, historical data from similar experiments and developmental environments could thus be incorporated in analyzing the uncertainty and used for compensating insufficient failure data.

After exploring the uncertainty for software component, this chapter further extended the uncertainty analysis to more complicated systems that contain numerous components, each with its own respective distributions and uncertain parameters. A Monte Carlo approach was proposed to solve it. This method is broadly applicable for many systems that can be modeled with different modeling tools. The approach was then illustrated with a case of three-module software on a Markov model, and another case of a grid service reliability (a type of large-scale system) based on Graph theory. These examples with distinguished characteristics exhibited the generality and effectiveness of the MC approach to analyze not only simple module-based systems but also complicated systems with numerous uncertain parameters.

This approach was further illustrated with a recently published model (Dai & Levitin, 2006) for large-scale system reliability. Adopting this approach allowed the relaxation of the assumptions of constant parameters in communication speed and processing speed, thus improving the practicality of the model. This demonstrated another novel application of the proposed uncertainty analysis. The results showed that the improved model accounting for the uncertainty factors was more realistic and more reasonable than the original model. Moreover, the improved model could further provide the percentiles and variance that exhibit the confidence range to the practitioners about the model's output.

CHAPTER 7 UNCERTAINTY ANALYSIS ON DDS RELIABILITY

Weighted voting systems (WVS) have attracted a lot of attention in the recent decades as they have been widely used in human organization systems, pattern recognition, detection systems, and etc. In chapter 3 and chapter 4, we have discussed some reliability models of WVS by taking into account continuous input and other properties in details. In this chapter, we study a specific realization of WVS - distributed detection system (DDS) and discuss the problem of uncertainty analysis and parameter estimation in reliability modeling of this system.

As a special type of weighted voting system, distributed detection systems are generally used for fault detection, which were first studied in Tenney and Sandell

(1981). Naim *et al*. (1991) propose a learning binary Bayesian distributed detection system where the probabilities are estimated on-line. Guo *et al*. (1991) use a distributed fault-detection and diagnosis system based on a distributed system identification approach to detect the fault in an intelligent control system to reduce the maintenance cost and increase system availability. Yao and Liu (1998) use a similar the structure of distributed detection system in evolutionary artificial neural networks to make best use of the population information. Schnier and Yao (2003) propose an algorithm of using negative correlation to evolve fault-tolerant circuit in the same structure.

Reliability modeling has gained considerable interest and acceptance by applying probabilistic methods to the real-world situation. For distributed detection systems, its reliability is defined as the probability the system provides correct output given corresponding input, which is determined by the accuracy and number of the local detectors, the threshold factor, and etc. These uncertain parameters are subject random variation as the performance of distributed detection system degrades and the detectors fail at certain rate. To calculate the system reliability, we have to estimate these system parameters precisely. Point estimation methods are often used to obtain these parameters; however, the uncertainty of parameters will be ignored which may cause underestimating the uncertainty in the system reliability.

The uncertainty of the parameters can be described with a probability model. From that, measures of the uncertainty can be obtained. To describe a probability model, we need more data than to obtain the point estimate of the parameter (Dai *et al*., 2005). This chapter applies the Bayesian approach to quantify and analyze the uncertainty of the unknown parameters. This approach initially describes the uncertainty of the parameter by a probability model known as the prior probability.

The parameter is assumed to be inferable from the data and observed current data can be used to get a more precise estimates of the parameter, whose uncertainty is then characterized by the posterior distribution. This approach is useful in reliability modeling as it can formally incorporate prior information in the analysis. This is especially useful when observed data is scarce or expensive to obtain. Moreover, the Bayesian method can also treat the correlation among those parameters by using the joint distributions.

In this chapter the effect of the uncertainty of the parameters on the reliability of the entire distributed detection system is studied. We quantify this uncertainty by the variance. A simulation is conducted as well to calculate the effect on the system reliability from the uncertainty of the parameters. We use an example to illustrate the parameter estimation by Bayesian approach.

This chapter is organized as follows: section 7.1 introduces the reliability modeling of distributed detection systems; in section 7.2, the parameter estimation of failure rate of distributed detection system is studied and an example is applied to illustrate the process. Section 7.3 studies the uncertainty problem in reliability modeling of DDS. Section 7.4 illustrates the procedures with a numerical example. And section 7.5 introduces briefly the parameter estimation problem on the threshold factor. Finally, section 7.6 concludes this chapter and discusses some possible future research.

7.1 Reliability Model

A distributed detection system functions exactly like a weighted voting system discussed in the chapters 3.4. It consists of *N* independent detectors, each of which submits a binary decision 0 or 1 to a data fusion center to calculate the system decision by comparing the cumulative weights to the preset threshold τ. The decision 0 or 1 represents rejection or acceptance of a proposition, respectively, as well as decision of alarming or not alarming on detection of faults (Guo *et al*, 1991). It slightly differentiates from the WVS systems we discussed in earlier chapters by considering discrete inputs only. The structure of the distributed detection system for fault detection in system engineering is shown in figure 1.

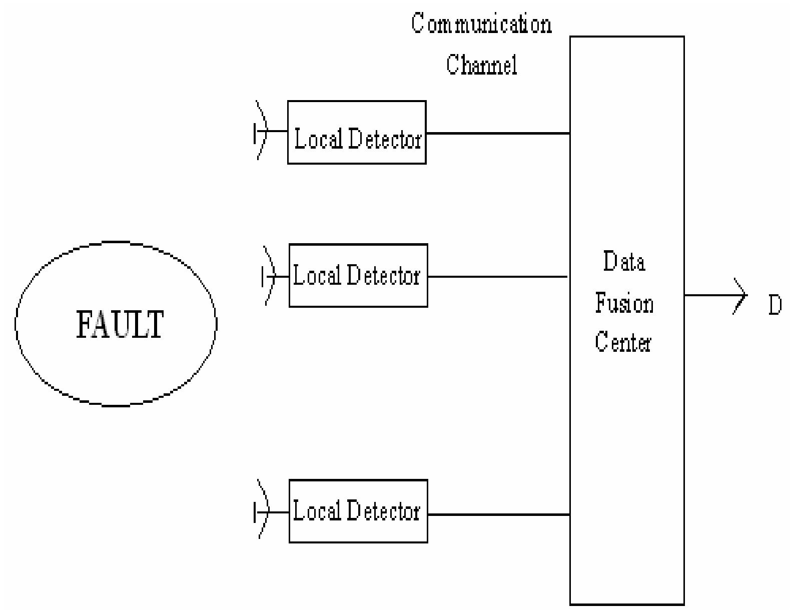

Figure 7.1Structure of DDS for fault detection

The following assumptions are made for the reliability modeling of such systems:

1) Each local detector works independently;

2) No detectors abstain from detection;

3) All detectors receive the same input *I* (0 or 1), which is a priori right or wrong in implicit information;

4) Decision transmission to the data fusion center is an error-free process.

The output of local detector is considered as correct output if it is equal to the input. Hence, two errors for the local detector are defined by the detector output, which is represented by $d_j(I)$, when it is in conflict with the input:

$$\begin{cases} d_j(0)=1 \\ d_j(1)=0 \end{cases} \tag{7.1}$$

We use terms q_{01}^j and q_{10}^j to represent the probability of the occurance of two errors of detector j, respectively. Let p_{00}^j and p_{11}^j denote the probability the detector j makes a correct decision corresponding to the given input, that is $p_{00}^j = \Pr(d_j(I)=0 \mid I=0)$ and $p_{11}^j = \Pr(d_j(I)=1 \mid I=1)$. So we have the following relations: p_{00}^j=1-q_{01}^j and p_{11}^j=1-q_{10}^j as no abstention from detection is allowed in the reliability modeling.

To make the system decision, the data processing center incorporates all the detector outputs into a unanimous system output D with following rule:

$$D(I)=\begin{cases} 1, & \text{if } \sum_{j=1}^{N} w_j \cdot d_j(I) \ge \tau \sum_{j=1}^{N} w_j; \\ 0, & \text{if } \sum_{j=1}^{N} w_j \cdot d_j(I) < \tau \sum_{j=1}^{N} w_j \end{cases} \tag{7.2}$$

where w_j is the weight assigned to detector j, which indicates its relative importance in the DDS and τ is a threshold factor in range of (0,1).

The system fails if $D(I) \neq I$. Hence, reliability of the distributed detection system is defined as the probability that the output of the entire system is equal to the input, which can be mathematically represented as:

$$R = \Pr(D(I)=I) \tag{7.3}$$

The expression (7.3) can be expanded to

$$R = \Pr(I = 0)\Pr(D = 0 \mid I = 0) + \Pr(I = 1)\Pr(D = 1 \mid I = 1) \tag{7.4}$$

From this expression, the reliability is a complex function with respect to the system parameters such as threshold value and the number of detectors. Levitin and Lisnianski (2001) present an efficient algorithm based on universal generating function technique to evaluate the reliability of the entire system and optimize the system reliability by finding an optimal solution of the threshold parameter in genetic algorithm.

7.2 Parameter Estimation

7.2.1 Problem Statement

Distributed detection system composes of a number of local detectors. The system has a lifetime which depends on the detectors functioning normally. With the degradation, the detectors may fails at some time which causes the performance of entire system to degenerates to a lower level. The lifetime of the detectors and the entire system can be described by widely used continuous parametric distributions: exponential distribution, Weibull distribution, lognormal distribution and etc (Kuo and Zuo, 2002). Hence, the performance of such system which is associated with the number of normally working detectors depends highly on the value of the parameters applied in the reliability models.

Distributed detection system is widely applied as well to detect faults and monitor potential dangers for military purposes and in extreme conditions such as outer space exploration (Tenney and Sandell, 1981). In this sense, the distributed detection system suffers a certain intensity of intentional and unintentional attack, for example, terrorists' bombing and too much exposure to the space radiation. Additionally, in such extreme condition, the necessary maintenance can not be provided immediately and sufficiently, even when some fatal failures have been reported. Therefore, the parameters of intensity of attack as well as the level of maintenance influence the reliability prediction to an important degree.

Some detectors are likely to abstain from detection or refuse to report their decisions to the data fusion center (Levitin and Lisnianski, 2001). The performance of the entire detection system is considered to be degenerate in this case. The probability that the detector refuses to work should be considered as well in estimating the system reliability.

Based on the analysis above, the reliability of distributed detection system depends highly on the various parameters associated with the lifetime distribution, intensity of attack, level of maintenance and probability of abstaining. However, these unknown parameters are subject to random variation or vary (or degrade) due to the different background or the systems in use. To predict the system reliability accurately, it is necessary to estimate the parameters by incorporating available history data set or expert advice and previous experience.

7.2.2 Parameter Estimation

The reliability of the distributed detection system is directly dependent on the number of available detectors which is defined as a function of parameter set $\lambda_s = \{\lambda_1, \lambda_2, ..., \lambda_t, ..., \lambda_T\}$ (T is the number of parameters) which can be described by conditional probability $p(n \mid \lambda_s)$. Denote the prior distribution of the parameter λ_s as $p(\lambda_s)$. This prior information on the parameters can be obtained from historical data.

Suppose *M* observations on performance of the detection system are made in the independent experiments and the number of working detectors in *m*-th experiment is recorded as c_m. Denote $\vec{c} = \{c_1, c_2, ..., c_M\}$. Given $\vec{c} = \{c_1, c_2, ..., c_M\}$, the joint density or likelihood function can be written by

$$p(\vec{c} \mid \lambda_s) = \prod_{m=1}^{M} p(c_m \mid \lambda_s) \tag{7.5}$$

Given the observation data and prior distribution of parameter λ_s, the posterior distribution can be calculated as

$$p(\lambda_s \mid \vec{c}) \propto p(\lambda_s) \cdot p(\vec{c} \mid \lambda_s) \tag{7.6}$$

From eq. (7.6), the marginal density function with respect to parameter λ_t can be obtained by

$$p(\lambda_t \mid \vec{c}) = \int\int \cdots \int p(\lambda_s \mid \vec{c}) d\lambda_1 d\lambda_2 \cdots d\lambda_{t-1} d\lambda_{t+1} \cdots d\lambda_T \tag{7.7}$$

the expectation of the parameter λ_t given the observations can be derived by

$$\hat{\lambda}_t = E(\lambda_t \mid \vec{c}) = \int \lambda_t \cdot p(\lambda_t \mid \vec{c}) d\lambda_t \tag{7.8}$$

and the variance of the estimated parameter λ_t is calculated by

$$\sigma^2(\lambda_t) = \int \left(\lambda_t - \hat{\lambda}_t\right)^2 \cdot p(\lambda_t \mid \vec{c}) d\lambda_t \tag{7.9}$$

7.2.3 Poisson Distribution

For a large distributed detection system consisting of a sufficiently large number of detectors, the number of failed detectors n_f at a certain time point can be approximately represented by Poisson distribution with single parameter λ :

$$p(n_f \mid \lambda) = \frac{\lambda^{n_f}}{n_f!} e^{-\lambda} \tag{7.10}$$

The distribution of the number of available detectors n in the detection system take the form

$$p(n \mid \lambda) = \frac{\lambda^{(N-n)}}{(N-n)!} e^{-\lambda} \tag{7.11}$$

By eq. (7.5), the likelihood function is calculated by

$$\begin{aligned} p(\vec{c} \mid \lambda) &= \prod_{m=1}^{M} p(c_m \mid \lambda) = \prod_{m=1}^{M} \frac{\lambda^{(N-c_m)}}{(N-c_m)!} e^{-\lambda} \\ &\propto \lambda^{MN-\sum_{m=1}^{M} c_m} e^{-M\lambda} \end{aligned} \tag{7.12}$$

According to eq. (7.6), we obtain the posterior distribution

$$\begin{aligned} p(\lambda \mid \vec{c}) &\propto p(\lambda) \cdot p(\vec{c} \mid \lambda) \\ &\propto \frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \cdot \lambda^{MN-\sum_{m=1}^{M} c_m} e^{-M\lambda} \end{aligned} \tag{7.13}$$

Hence, by simplifying eq. (7.13), we obtain the posterior distribution of parameter λ

$$p(\lambda \mid \vec{c}) = Gamma\left(\alpha + MN - \sum_{m=1}^{M} c_m, \beta + M \right) \tag{7.14}$$

By incorporating eq. (7.6), eq. (7.8), eq. (7.9) and eq. (7.14), the expect value and variance of the parameter λ are

$$\hat{\lambda} = E(\lambda \mid \vec{c}) = \int \lambda \cdot p(\lambda \mid \vec{c}) d\lambda = \frac{\alpha + MN - \sum_{m=1}^{M} c_m}{\beta + M} \tag{7.15}$$

$$\sigma^2(\lambda) = \int (\lambda - \hat{\lambda})^2 \cdot p(\lambda \mid \vec{c}) d\lambda = \frac{\alpha + MN - \sum_{m=1}^{M} c_m}{(\beta + M)^2} \tag{7.16}$$

After obtaining the posterior distribution of parameter λ, the discrete distribution of the number of available detectors at time T_p given the observed data can be calculated

$$\begin{aligned} p(n) &= \int p(n \mid \lambda) p(\lambda \mid \vec{c}) d\lambda \\ &= \int \frac{\lambda^{(N-n)}}{(N-n)!} e^{-\lambda} Gamma\left(\alpha + MN - \sum_{m=1}^{M} c_m, \beta + M \right) d\lambda \end{aligned} \tag{7.17}$$

7.3 Uncertainty on System Reliability

In the distributed detection systems, each local detector is assigned its corresponding weight to indicate its relative importance and influence to the entire system reliability. Hence, the reliability is actually a function of the number of available detectors as well as weight allocation policy which, from another perspective, is related to the distribution of survival detectors. This means the reliability will be estimated with different value even for the same number of available detectors because different detectors with different weights allocated may fail at different rates. For the sake of simplicity, in our chapter, we assume the hazard rate to each local detector is the same. That is, at any time point, the possibility that any detector fails to work is assumed to be the same. This assumption simplifies the calculation of the probability of different sets of failed detectors. Let $U_n=\{U_1, U_2,\dots,U_{vn},\dots U_{Vn}\}$ represent the sets of failed detectors when n detectors still work properly in the system. Event v_n in the failure set occurs with probability $\Pr(U_{vn})$. The reliability of distributed detection system in failure set v_n given n detectors working normally is calculated by the conditional probability $\Pr(U_{vn}|n)$.

Denote $R(n)$ as the reliability function with respect to the number of detectors n. Therefore, the term $R(n)$ used in this chapter estimates the average effect on reliability of the different cases of failed detectors given the number n

$$R(n) = \sum_{U_{vn} \in U_n} \Pr(U_{vn} \mid n) \cdot \Pr(U_{vn}) \tag{7.18}$$

Obtaining the reliability of distributed detection system with n available detectors from eq. (7.18), we calculate the mean value of the reliability of the entire distributed detection systems by

$$\hat{R} = E(R(n)) = \sum R(n) \cdot p(n) \tag{7.19}$$

and the variance of reliability by

$$\mathrm{var}(R) = E\left[\left(R(n) - \hat{R}_n\right)^2\right] = \sum\left(\left(R(n) - \hat{R}_n\right)^2 \cdot p(n)\right) \tag{7.20}$$

The system reliability is a function of parameter λ

$$R(\lambda_s) = \sum R(n) \cdot p(n \mid \lambda_s) \tag{7.21}$$

From the eq. (7.21), the mean value of the reliability can be alternatively calculated by

$$\begin{aligned}\hat{R} &= \int R(\lambda_s) p(\lambda_s \mid \vec{c}) d\lambda_s \\ &= \int \sum R(n) \cdot p(n \mid \lambda_s) \cdot p(\lambda_s \mid \vec{c}) d\lambda_s\end{aligned} \tag{7.22}$$

which can be deduced directly from eq. (7.19) as well. The variance of entire system reliability can be alternatively obtained by

$$\mathrm{var}(R) = \int \left(R(\lambda_s) - \hat{R}\right)^2 p(\lambda_s \mid \vec{c}) d\lambda_s \qquad (7.23)$$

As the reliability function has combinatorial complexity and the closed form of *R*(*n*) is hard to obtain, we design a generic Monte Carlo simulation to estimate the mean value and the variance of the reliability of entire system. In the simulation, system reliability is estimated by following the recursive method proposed by Levitin and Lisnianski (2001) based on universal generating function. The following algorithm provides the generic Monte Carlo Simulation method to estimate the effect on reliability of entire system of uncertainty in parameter *n*.

```
1. begin
2. for $i_1$=1 to L    //$L$ is the total number of the iterations
3. for $i_2$= 1 to N
4. Generate a sample $n$ from the posterior distribution of $p(n \mid \vec{c})$
5. Select randomly $n$ detectors out of total $N$ detectors to work normally
6. for $i_3$= 1 to M
7. Generate a sample $I$ from the distribution $f(I)$
8. Calculate system reliability $R(n)$ given the input and system configuration
   //using the estimate algorithm in Levitin and Lisnianski (2001)
9. end         // a simulation on $I$ following the distribution of $f(I)$
10. end  // a simulation on $n$ following the distribution of $p(n \mid \vec{c})$
11. end              //Now, $L$ sample points of the system reliability are saved in $R$.
```

12. Calculate $\hat{R}_n = \frac{\sum_1^N R_n}{N}$

13. Calculate $\operatorname{var}(R_n)$

14. **end**

By the above method, we can calculate the mean value and variance of reliability function accurately and efficiently.

7.4 Numerical Example

In this subsection, a numerical example of a detector system in spaceship is applied to illustrate parameter estimation by the Bayesian approach. On a spaceship launched into outer space, a detection system consisting of 5 detectors is installed to monitor the faults of a device. Susceptible to possible collisions from unknown objects and much exposure to radiation, each detection system has a limited lifetime. From previous data we assume that the prior distribution of failure rate λ to the entire distributed detection system follows a Gamma distribution with parameters $\alpha = 1.5$ and $\beta = 1$: $p(\lambda) = 1.128\lambda^{0.5}e^{-\lambda}$. At the beginning of the project, we have to forecast the system reliability at time T_p after the successful launch of the entire system when the performance of distributed detection system is degraded as a number of detectors fail working in the outer space environment without necessary maintenance. In this example, the effect on system reliability of time factor is not considered, so T_p here is treated as a hyper-parameter that is not of interest. Each detector is assumed to by two-state system: working perfectly or failing working. The weights allocation and the probability of the detector producing correct decision are both shown in the following table. We assume that the fault comes out at 70% percent of time that $p(I=1)=0.7$ in this example.

Table 7.1 Configuration of distributed detection system

No.	1	2	3	4	5
Weight	2	2	3	4	5
p_{00}^{j}	0.8	0.75	0.8	0.85	0.9
p_{11}^{j}	0.8	0.8	0.75	0.8	0.85

The failure data of the distributed detection system is collected at a fixed time T_p after the detection system is put into use. In each observation, we record the experiments results in the following table.

Table 7.2 Observations on the number of available detectors

No.	1	2	3	4	5	6	7	8	9	10
No. of detectors	4	3	5	4	4	2	4	3	4	5

From eq. (7.12), the joint distribution can be calculated as $p(\vec{c} \mid \lambda) \propto \lambda^{MN-\sum_{m=1}^{M} c_m} e^{-M\lambda} = \lambda^{12} e^{-10\lambda}$, and the prior distribution of parameter λ is $p(\lambda) = 1.128\lambda^{0.5} e^{-\lambda}$. According to eq. (7.14) we obtain the posterior distribution of parameter λ as $p(\lambda \mid \vec{c}) = Gamma(13.5, 11)$.

Hence, the expect value of parameter λ by Bayesian method is estimated as $\hat{\lambda} = E(\lambda \mid \bar{c}) = \frac{\alpha + MN - \sum_{m=1}^{M} c_m}{\beta + M} = 1.2273$, and the variance is calculated as $\sigma^2(\lambda) = \frac{\alpha + MN - \sum_{m=1}^{M} c_m}{(\beta + M)^2} = 0.1116$.

The total number of detectors of this detection system is N=5, however, number of failed detectors n_f is assumed to follow Poisson distribution where n_f covers all the integers with positive possibilities. We notice that the system reliability will go to 0.7 by setting system output to be 1 when all the detectors fail working. To make the Poisson distribution fit this model, we sum up all the probabilities that n_f exceeds N, that is n<0, and define the reliability of the detection system for all these cases to be 0.

Table 7.3 Probability distribution and reliability estimated at n

n	$p(n)$	R(n)
<0	0.0030	0
0	0.0091	0.7
1	0.0311	0.8060
2	0.0904	0.8290
3	0.2100	0.8995
4	0.3475	0.9207
5	0.3089	0.9428

According to eq. (7.17) and (7.18), the probabilities as well as the reliability estimated of the number of available detectors n at time T_p are shown in Table 7.3. The probability of n being negative is 0.003. This means Poisson distribution describes very well the distribution of the number of available detectors with a negligible error, though the support of Poisson distribution covers all the positive integers while only 7 possibilities in our example are considered.

The expect value of entire system reliability is estimated by eq. (7.19):

$$\hat{R} = E(R(n)) = \sum R(n) \cdot p(n) = 0.9064 \cdot$$

and the variance of reliability is estimated from eq. (7.20):

$$\text{var}(R) = E\left[\left(R(n) - \hat{R}_n\right)^2\right] = \sum \left(\left(R(n) - \hat{R}_n\right)^2 \cdot p(n)\right) = 0.0042 .$$

7.5 Parameter Estimation on Threshold

In the reliability modeling described above, the parameter τ is important to the system reliability. The system may make a totally different decision if the parameters are just changed by a small degree, even though all the detectors send the same decision to the data fusion center.

For the on-line estimation for distributed detection system proposed by Naim *et al.* in IFAC (1991), the threshold factor τ is calculated by the logarithm function of the ratio the probability input I=0 over the probability I=1. The threshold is continuously updated with the input information at each step. The threshold may deviate from the correct value due to the various problems in the on-line updating process.

To evaluate the reliability of the systems with the unknown structure and parameters, we also face the problem of estimating the parameters accurately given the scarce information we have. Our task is to reduce the uncertainty in estimating the system reliability given the uncertainty of these parameters.

Therefore, estimating the parameter with small variation, which is a huge task, is necessary for calculating the reliability of the entire DDS accurately. Ignoring the parameters uncertainty can result in underestimating of the uncertainty in the entire system reliability. The measures of variance, confidence interval, bounds etc can well represent the uncertainty of the reliability, and give a more reasonable and more detailed result for the system reliability than just a point-estimated value. The Bayesian approach provides a coherent setup to obtain these measures.

7.6 Summary

Most reliability models are associated with their own parameters which are typically estimated from the historical data. For the widely used distributed detection system in fault detection, the system reliability depends on the number of normally working detectors and the accuracy of its local detectors. To evaluate the reliability accurately, it is necessary to obtain the system parameters precisely from the test data we hold. However, parameters of the reliability model are subject to random variation as the detection system may be used in different purposes and environments.

In this chapter, a Bayesian approach is presented to estimate the unknown parameters from the scarce data and quantify the uncertainty on the system reliability by measure of variance. A simulation is conducted as well to calculate the effect on the system reliability from the uncertainty of the parameters. An example is applied to illustrate the parameter estimation by Bayesian approach.

CHAPTER 8 PREVENTIVE RESOURCE ALLOCATION

8.1 Apical Dominance

In botany, a famous phenomenon in the growing process of a plant, named as apical dominance, shows how a plant allocates its resource to the most necessary parts and how this strategy works. A kind of plant hormone, called auxin, is the key factor to the growth of the plant cell (the auxin stimulates the plant cell to be elongated), which was found in the actively growing apical bud nearly two centuries ago (Cline, 1997). Auxin causes the lateral buds to remain dormant because the auxin is transported basipetally from the apical bud; so the concentration of auxin at lateral buds is lower than that in apical buds. The auxin stimulates the apical buds to grow and inhibit the lateral buds as well, as shown in Figure 8.1.

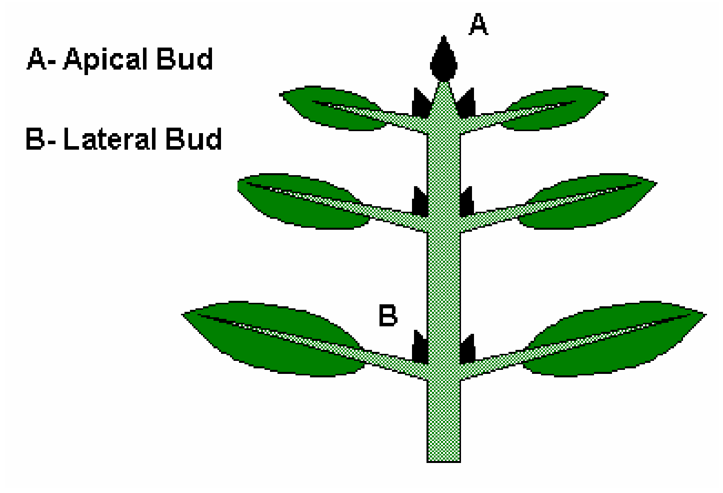

Figure 8.1 Apical dominance for a tree

The natural resource allocation strategy controlled by auxin which is abundant in apical bud can heuristically be applied in finding an efficient and optimal resource allocation strategy for security in reliability systems, especially in systems with complex structures and a large number of components, such as large-scale grid computing systems, and Peer-to-Peer network systems.

We assume system *G* consists of *N* components working independently in different locations with various functions. Each component *i* is measured with its reliability R_i, which represents current reliability of the component *i*.

Among these *N* components, some have more influence on the entire system. Take grid system for example, the reliability of resource management systems (RMS) is the most critical part to the entire grid systems compared with other components. For the pee-to-peer systems with super- peer structure, some peers with stronger ability to deal with data and transfer data to other peers are considered as super-peer nodes.

These peers contribute much more proportion to the performance and reliability of entire P2P system.

In the herbaceous plant and woody plant growth process, this resource allocation is controlled by apical dominance strategy where the apical bud is stimulated to grow with much more resource allocated and the lateral buds are inhibited as the concentration of auxin in apical bud is higher than that in lateral buds. This is the rule in the natural world where the concentration of auxin determines effectively the resource transported. This also heuristically teaches us the potential resource allocation policy for reliability systems, especially large-scale complex systems, such as grid systems and P2P systems. The similarities between apical dominance in a tree and resource allocation strategy for a engineering system (grid system) are elaborated in Table 8.1.

Table 8.1 Comparison of tree and grid system

Tree	Grid System
apical bud	RMS
high concentrate of auxin	high importance
more water and nutriment	more resource

However, the critical questions are: how to define the ‘auxin’ in reliability system and how to measure the ‘auxin’ of each component *i* in a complex system in order to compare them and sort them to decide what amount of resource to be allocated to component *i* with an optimal resource allocation policy that ensures the limited resource is distributed effectively to improve the system performance and reliability as much as possible. In our model, we define α_i as the concentration of auxin in component *i* in the complex system.

This chapter is organized as follows. Section 8.2 introduces the model and discuss 3 important factors for calculating α. In section 8.3, we propose an optimal resource allocation strategy. In section 8.4 a numerical example is presented to illustrate the process of calculating α. Section 8.5 gives a summary.

8.2Factors in Preventive Resource Allocation

Similar to how auxin in a plant bud is determined by bud location, type of the plant, and many other factors, α_i of component *i* is evaluated by many important measures. These measures include:

1). Reliability importance of component *I,* denoted by I_{R_i}, which indicates the effect of failure in individual component or subsystem on overall system reliability

2). Cost ratio for improving reliability of component *i* denoted by C_{R_i}. This measures the efficiency of the resource allocation strategy to increase component reliability. 3).Potential attacks the system experiences, which is denoted by T_{R_i}.

Now, we can consider α_i as a function of both I_{R_i} and C_{R_i}: $\alpha_i = f\left(I_{R_i}, C_{R_i}, T_{R_i}\right)$

8.2.1 Reliability Importance Measure

Different importance measures can be classified into two categories: structural importance and reliability importance. Structural importance indicates the topological position of the component in a system, while reliability importance, first introduced by Birnbaum (1969), is used to rank the significance of individual components in a reliability system, with respect to their contribution to the overall reliability of the system. This measure is important in figuring out the weakness in a system and searching for the best strategy for allocating resources to the most important components in a complex system. After Birnbaum's paper, a number of importance measures have been introduced for binary systems, which generally take the following forms (Leemis, 1995):

$$I_{R_i} = \frac{\partial R_s}{\partial R_i} \tag{8.1}$$

where I_{R_i} is reliability importance of the i^{th} component, R_s is the system reliability and R_i is the component reliability.

In this definition, a system is defined as binary system when the system is either functional or non-functional. Such a system has very wide applications in engineering and many other fields; research on various importance measures of binary system has also received increasing attention. Research of reliability importance for binary system have also been extended to multistate system of which the component or system have more than 2 states. Xie and Shen (1989) suggest using different rankings for the corresponding improvement actions. Chang *et al*. (2004) use OBDD based algorithms to calculate system failure frequencies and reliability importance measures, where the reliability importance measures include Birnbaum importance, the Criticality importance and other indices. Meng (2004) presents some simple criteria to compare Birnbaum reliability importance measure of components in a general binary coherent system.

Another importance index is joint importance measure, which scales the interaction of two components in contribution to the overall system reliability. Wu (2005) introduces joint structural importance measure and joint reliability importance measure for multistate system. Zio and Podofillini (2006) present a differential importance measure (DIM) and a second order extension of DIM to account of the interactions between two components when evaluating the effect of changes in the reliability parameters of components on the system reliability.

The concept of auxin in this chapter can be interpreted in nature as a special kind of reliability importance measure. This term differentiates from the original representation of reliability importance measure in the literature by taking into account resource allocation efficiency to component and potential threat or intentional attack from outsiders (e.g., terrorists attacks).

8.2.2 Cost Factor

Reliability importance in the preceding subsection is mainly about how the changes in component reliability will improve the overall system reliability. However, so far, we do not know how to improve the reliability of individual components most efficiently under the resource restrictions. In following subsection, we try to depict the possible relationships between resource allocated in components and corresponding reliability improvement.

Some components in the system may be even harder or more costly to improve, that is, more preventive resource need to be allocated to the components to increase their reliability. However, there is no clear conclusion on the relationship for the cost of each component as a function of its reliability. Mettas (2000) provides a general empirical function, which is derived from past experiences or data to overcome the problem. The cost function takes the following form

$$c_i\left(R_i; f_i, R_{i,\min}, R_{i,\max}\right) = e^{\left[(1-f_i)\cdot\frac{R_i - R_{i,\min}}{R_{i,\max} - R_i}\right]} \quad (8.2)$$

where $R_{i,min}$ is minimum reliability of component/subsystem i, $R_{i,max}$ is maximum achievable reliability of component/subsystem i, and f_i is feasibility of increasing the reliability of component/subsystem i, which is assumed between 0 and 1. f_i represents the relative difficulty in increasing a component's reliability.

Charles Elegbede *et al*. (2003) also provide 3 reasonable conditions to find the cost function c_i. c_i is positive definite, non decreasing and it increases rapidly when reliability approach to 1.

Different resource allocation strategies for the same component incur as well different cost. Xie and Shen (1989) enumerate three alternative improvement actions to increase the system reliability: equal improvement, replacement by a perfect component, and active redundancy. Take 'active redundancy' policy for example. Suppose the reliability of i^{th} computing node (such as a computer, or memory) in a grid computing system is R_i with cost c_i. If we use parallel redundancy by adding one active computing node to avoid system failure caused by accidents in component i, the cost will be doubled. In this way the reliability of this component increases from R_i to $1-(1-R_i)^2$. Similarly, if we use 3 active computing nodes in parallel to improve the system performance, the reliability of this subsystem of components will be $1-(1-R_i)^3$ with cost $3c_i$. The following figure depicts the relationship between reliability improvement and the number of units in parallel redundancy.

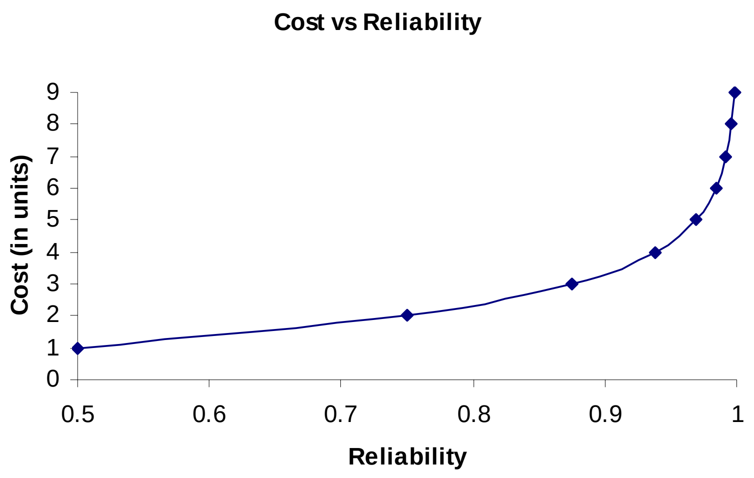

Figure 8.2 Cost vs Reliability

The cost curve in Figure 8.2 coincides with eq. (8.2) very well. They both indicate that cost of a high/low reliability component is high/low.

This poses a new problem: which component should we choose to improve with minimum resource? A new index, cost coefficient, is introduced here to solve this problem: derivative of cost with respect to reliability obtained from eq. (8.2):

$$C_{R_i} = \frac{\mathrm{d}c_i}{\mathrm{d}R_i} = (1-f_i)\cdot\frac{R_{i,\max}-R_{i,\min}}{(R_{i,\max}-R_i)^2}e^{\left[(1-f_i)\frac{R_i-R_{i,\min}}{R_{i,\max}-R_i}\right]} \tag{8.3}$$

This index measures the efficiency of allocating resource in component i to increase its reliability: Higher C_{R_i} indicates that it is more costly to improve i^{th} component compare to improving a component with lower cost coefficient.

8.2.3 Attack Factor

Besides the two factors we have discussed previously, the third important factor we account for in the strategy of resource allocation for improving system reliability is the potential threat or attack each component or subsystem experiences. Especially for large scale network systems such as grid and P2P computing systems, where the attackers can easily access the loosely distributed computing nodes which may be performing safety critical computing tasks. The intentional attacks from terrorists may cause the computing system to fail. To improve the reliability of such safety critical systems, we must consider outside factors when we allocate the preventive resources, such as guards, security devices, and so on. The following subsection focuses on strategies for preventive resource allocation to protect the components and systems from possible threat and intentional attacks or at least to mitigate the destructive effects from these outside attacks.

From the previous research on the optimal defensive strategy in response to outside attacks, we can easily see that the attacker (terrorist) is presumed to have different preference on choosing locations to attack. They may have reliable or unreliable information of the defender's resource allocation strategy against the potential threat which may be collected from public channel or private channel. The outside attacks are carried on under a certain level of cost limitation and the payoff of every possible attack is also expected. So analyzing the attackers' decision is very complicated, and to some extent, impossible. However, it is crucial for safety critical system, such as power transmission system and power station, where an outside attack may cause destructive effect.

Bier *et al*. (2005b) study a strategic model by using game theory in which two players are involved: a defender and an attacker. The defender must choose a collect of locations to allocate defensive resources and the attacker must a component to attack. They reach some conclusions as follows: it is sometimes an optimal strategy that the defender leaves a location undefended. The defender has a preference of allocating resource in a centralized way. And in the value of the attacker's outside option, the optimal allocation could be non-monotonic. Bier et al. (2007) propose a simple but practical Max Line method based on a greedy algorithm to assess the vulnerability of complex systems to intentional attacks and study the effectiveness of the method. Levitin (2007) extends the research of optimal defense strategy to multi-state series-parallel system. The author presents a model in which separation and protection of system elements are considered.

8.3 Optimal Strategy

As we stated previously, we can implement the strategy of apical dominance, which has proven to be efficient and effective in biology, to simulate resource allocation decision process in complex system such as grid or p2p computing systems to improve optimally the reliability of the system under the limited budget. In apical dominance, concentration of auxin in a bud determines the amount of resources, such as water and nutriment, that can be transported to the bud and controls (stimulates or inhibits) its growth. Concentration of auxin is affected by many factors such as the location, plant species, and sunlight. The case of apical dominance is similar to resource allocation for engineering systems: the strategies are highly dependent on reliability/structure importance measure, cost coefficient, and outsider threat or intentional attack. We use $\alpha_i = f\left(I_{R_i}, C_{R_i}, T_{R_i}\right)$ to represent the relationships.

The relationships might not be so clear and unique for different systems. However, following rules are implied:

(1). more preventive resource should be allocated to the components with higher reliability/structure importance

(2). resource should be allocated in priority to the component with low cost-coefficient. This ensures that resources are allocated in the most efficient manners.

Table 8.2 Auxin and α

Auxin	α
Location	*Reliability importance*
Different species	*Cost Coefficient*
Sunlight	*Outside Threat*

(3). the component under more potential threat should be allocated more resources.

Rule (3) is not valid for the complicated case in Bier et al. (2005). But in this chapter, for the sake of simplicity, we assume this rule holds.

Considering the 3 rules, we propose the following form to calculate α_i for component *i*:

$$\alpha_i = \frac{I_{R_i}}{C_{R_i}} \cdot T_{R_i} \tag{8.4}$$

As we know from the earlier analysis, the cost coefficient of component will increase exponentially over the reliability improvement of the component. In this sense, α_i is a decreasing function of reliability of component *i*. With continuously resource supplement to component *i*, α_i will approach a certain level until the optimal allocation is reached.

***Theorem**: for a coherent system, the resource allocation strategy is optimal if and only if*

$$\alpha_1 = \alpha_2 = \cdots = \alpha_i = \cdots = \alpha_N \tag{8.5}$$

***Proof*:**

Assume reliability of the two components are R_i and R_j with accumulative cost C_i and C_j respectively. Hence the sum of cost to maintain the reliability of these two components at level R_i and R_j is $C = C_i + C_j$. And the contribution to the overall system reliability can be obtained as $Cont = I_{R_i} T_{R_i} R_i + I_{R_j} T_{R_j} R_j$.

Therefore, we obtain a maximization problem:

$$\begin{aligned} &Max: \quad Cont = I_{R_i} T_{R_i} R_i + I_{R_j} T_{R_j} R_j \\ &Sub: \quad C_i + C_j \le C \end{aligned} \tag{8.6}$$

The fact that the system is coherent ensures that the optimal solution is obtained at boundary condition. We turn the original problem into:

$$\begin{aligned} &Max: \quad Cont = I_{R_i} T_{R_i} R_i + I_{R_j} T_{R_j} R_j \\ &Sub: \quad C_i + C_j = C \end{aligned} \tag{8.7}$$

From eq. (8.2), we have

$$R_i = \frac{(1 - f_i)(R_{i,\max} - R_{i,\min})}{f_i - 1 - \ln C_i} + R_{i,\max} \tag{8.8}$$

Substitute $C_j = C - C_i$ and eq. (8.8) into the objective function, we have

$$\begin{aligned} Cont &= I_{R_i} T_{R_i} R_i + I_{R_j} T_{R_j} R_j \\ &= I_{R_i} T_{R_i} \left(\frac{(1 - f_i)(R_{i,\max} - R_{i,\min})}{f_i - 1 - \ln C_i} + R_{i,\max} \right) + I_{R_j} T_{R_j} \left(\frac{(1 - f_j)(R_{j,\max} - R_{j,\min})}{f_j - 1 - \ln(C - C_i)} + R_{j,\max} \right) \end{aligned} \tag{8.9}$$

Let $A_i = I_{R_i} T_{R_i} (1 - f_i)(R_{i,\max} - R_{i,\min})$

From eq. (8.9), we know the total contribution from component *i* and *j* is a function of C_i. The optimal value may lie at the point where the derivative of *Cont* with respect to C_i is 0:

$$\frac{\mathrm{d}Cont(C_i)}{\mathrm{d}C_i} = 0 \tag{8.10}$$

We have

$$\frac{A_i}{(f_i - 1 - \ln C_i)^2}\frac{1}{C_i} = \frac{A_j}{(f_j - 1 - \ln C_j)^2}\frac{1}{C_j} \tag{8.11}$$

Substitute eq. (8.2) into eq. (8.11),

$$\frac{I_{R_i}T_{R_i}}{\frac{(1-f_i)(R_i - R_{i,\min})}{(R_{i,\max} - R_i)^2}}\frac{1}{e^{\left[(1-f_i)\frac{R_i - R_{i,\min}}{R_{i,\max} - R_i}\right]}} = \frac{I_{R_j}T_{R_j}}{\frac{(1-f_j)(R_j - R_{j,\min})}{(R_{j,\max} - R_j)^2}}\frac{1}{e^{\left[(1-f_j)\frac{R_j - R_{j,\min}}{R_{j,\max} - R_j}\right]}} \tag{8.12}$$

Compare with eq. (8.3), we obtain

$$\frac{I_{R_i}T_{R_i}}{C_{R_i}} = \frac{I_{R_j}T_{R_j}}{C_{R_j}} \tag{8.13}$$

that is

$$\alpha_i = \alpha_j \tag{8.14}$$

In order to verify that this is the optimal strategy, it is necessary to verify the concavity of *Cont* function, that is to verify the sign of the second order of derivative of *Cont* with respect to C_i: $\frac{\mathrm{d}^2 Cont(C_i)}{\mathrm{d}C_i^{\,2}}$.

$$\frac{\mathrm{d}^2 Cont(C_i)}{\mathrm{d}C_i^{\,2}} = -\frac{A_i(3 + \ln C_i - f_i)}{(\ln C_i + 1 - f_i)^3} - \frac{A_j(3 + \ln C_j - f_j)}{(\ln C_j + 1 - f_j)^3} \tag{8.15}$$

Since $\ln C_i > 0$ and f_i is valid from 0 to 1, the second order of derivative of *Cont* is negative. Concavity of the function is proved.

Based on the concavity, we conclude that the optimal strategy is reached when eq. (8.14) is satisfied. The validation to an *N*-component system is easy to verify as well.

8.4Numerical Example

Suppose in a grid network system, some routers connect to each other in a bridge structure network to exchange data between two grid computing nodes, to perform the desired tasks as shown in Figure 8.3.

The routers in our example are assumed to be vulnerable under different levels of intentional attacks T_i but the connections between the routers are assumed to be invulnerable which are depicted in bold.

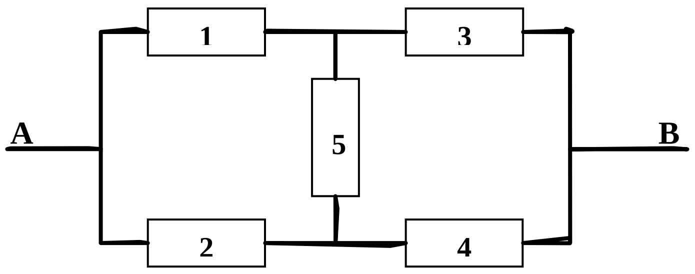

Figure 8.3 Bridge network in a grid computing system

From existing knowledge, the overall system reliability is a function of all components as follows:

$$R = R_1R_3 + R_2R_4 + R_1R_4R_5 + R_2R_3R_5 - R_1R_2R_3R_4 - R_1R_2R_3R_5 \\ - R_1R_2R_4R_5 - R_1R_3R_4R_5 - R_2R_3R_4R_5 + R_1R_2R_3R_4R_5$$

Following eq. (8.1) to take derivatives, we obtain the reliability importance measures of each component:

$$I_1 = R_3 + R_4R_5 - R_2R_3R_4 - R_2R_3R_5 - R_2R_4R_5 - R_3R_4R_5 + R_2R_3R_4R_5$$

$$I_2 = R_4 + R_3R_5 - R_1R_3R_4 - R_1R_3R_5 - R_1R_4R_5 - R_3R_4R_5 + R_1R_3R_4R_5$$

$$I_3 = R_1 + R_2R_5 - R_1R_2R_4 - R_1R_2R_5 - R_1R_4R_5 - R_2R_4R_5 + R_1R_2R_4R_5$$

$$I_4 = R_2 + R_1R_5 - R_1R_2R_3 - R_1R_2R_5 - R_1R_3R_5 - R_2R_3R_5 + R_1R_2R_3R_5$$

$$I_5 = R_1R_4 + R_2R_3 - R_1R_2R_3 - R_1R_2R_4 - R_1R_3R_4 - R_2R_3R_4 + R_1R_2R_3R_4$$

Before we analyze the resource allocation strategy, we must collect some useful information of the system as the following table shows:

Table 8.3 Calculation of Alpha

	1	2	3	4	5
Attack	0.8	0.88	0.85	0.9	0.85
Cost	12	11	13	10	10
Rmax	0.95	0.99	0.9	0.99	0.95
Rmin	0.5	0.4	0.3	0.5	0.4
Feasibility	0.3	0.2	0.4	0.3	0.2
Reliability	0.8511	0.8424	0.7862	0.8758	0.8082
Importance	0.3052	0.3009	0.3666	0.2334	0.4563
CostCoefficient	386.42	238.33	361.72	262.84	218.77
Alpha	**0.0006320**	**0.001111**	**0.000861**	**0.000799**	**0.001773**

The information includes the attack level that each component is exposed to, the amount of resource has been used to build up the components, $R_{i,\max}$, $R_{i,\min}$ and relative feasibility.

Based on this, reliability of each component is calculated by eq. (8.2) and reliability importance is obtained by eq. (8.1). We use eq. (8.3) to calculate cost coefficient of each component. We can then obtain α by using our definition and compare them in Figure 8.4 to determine which component is the weakest (with highest value of α). In our example, we shall choose component 5 to improve first to use the resource most efficiently.

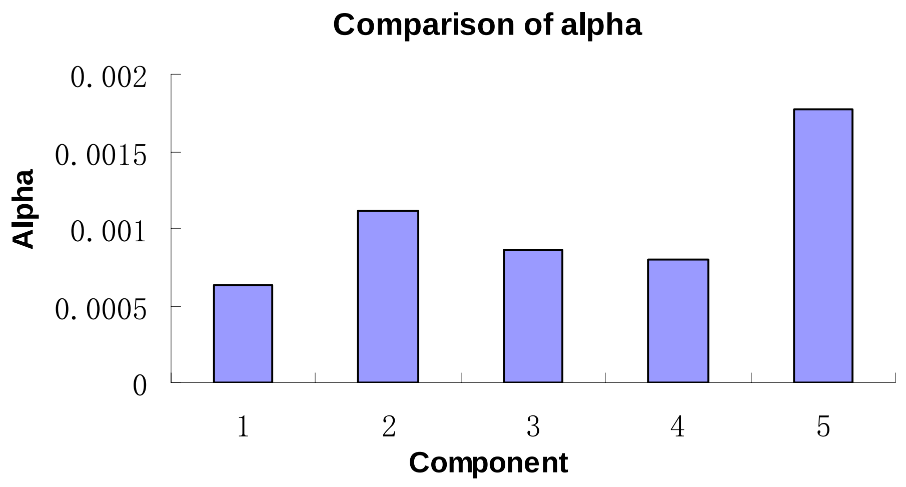

Figure 8.4 Comparison of alpha

8.5 Summary

In this chapter, we have introduced an important phenomenon of apical dominance in plant growth process in which the concentration of auxin in buds controls the amount of resource to allocate. This phenomenon shows great similarities to the process of allocating resource in complex engineering systems such as grid systems and p2p systems. The similarities have also been analyzed to get the important factors, which include reliability importance measures, cost coefficients, and intentional attacks they may suffer.

As analyzed, the cost coefficient of a component varied when its reliability was improved. α is a function of the cost coefficient which changes as well. This makes the resource allocation strategy more complicated. In this chapter, we provide a sufficient and necessary condition to judge whether the optimal strategy is reached. A numerical example is also presented to illustrate the process of calculating α for each component.

Although the factor of attack from outsiders is considered as an important part in calculating α , it is still a long way to understand clearly how intentional attack will affect the resource allocation strategy, especially to complex systems such as grid computing systems. The real situation of attacking and defending is much more complicated and many researchers use game theory and other technique to model this. Further analysis on the effect from attacker's decision on resource allocation strategy is critical and necessary to carry on.

CHAPTER 9 CONCLUSIONS AND FUTURE WORK

The concept of reliability of computing systems has attracted more and more attention from many practical areas, due to explosive development of information technology, which brings to us as well the exponentially increasing in size and complexity of the computing systems. This thesis mainly focused on building reliability models for different common used computing systems, such as weighted voting systems, peer-to-peer systems and software systems, conducting comprehensive analysis of the models we built, and designing optimal resource allocation strategies to improve system reliability based on the information we had.

This chapter summarizes the results and the contributions of the research work in this dissertation and discusses their limitations and implications. Suggestions on the possible future research are indicated as well.

9.1Summary

Computing systems are a kind of system with one or more computers/processors and associated software with common storage, which perform computing tasks in meaningful ways. Faults in either hardware or software may cause failures in the entire computing systems. Hence, in this thesis, research on reliability modeling and analysis of computing systems covers both of hardware systems and software systems.

Computing tasks that different computing systems execute lie in numerous forms in many practical areas. Weighted voting system is a computing system that is widely used in pattern recognition, human organization systems, fault detection and technical decision making systems. Chapter 3, and chapter 4 both focus on some important issues related to reliability modeling and analysis of weighted voting systems (distributed detection system is a special case). Chapter 3 studied a new reliability model of WVS by taking into account continuous state input which is represented by measures such as temperature and pressure. To reduce the computing complexity, some necessary assumptions were made. The distribution of output of each voting unit was assumed to be normally distributed with the mean value being the input. The output of entire system was the weighted sum of the outputs of units composing the system. The definitions of 'correct decision' and 'reliability of entire system' were modified correspondingly as the inputs were continuous. A reliability optimization problem with cost constraints was formulated as well under the consideration that different voting units have different accuracies and cost. The different allocations of these voting units could make the reliability of the entire voting system different. GA was applied to obtain the optimal strategy of allocating the voting units. A detailed GA was introduced in this chapter with an illustrative example.

Extensions on the reliability models built in chapter 3 are presented in chapter 4, in which the output of voting units are considered to be biased to the input and the accuracy of the units are assumed to depend on the input. Three different cases of the weighted voting systems, accounting for different assumptions and application backgrounds, were discussed. To illustrate the three cases, three numerical examples were conducted respectively. Reliability of the weighted voting system was calculated both by Monte Carlo and by analytical method for each example. Comparing the first two cases, we find that the reliability of the biased voting system is lower than the unbiased voting system, given the same accuracy of the system. A brief comparison of the two methods was conducted and we find that both methods have their own advantages and disadvantages.

As a newly developed distributed computing system, peer-to-peer has been widely accepted by users for network services such as distributed computing, file sharing, distributed storage, communication, and real time media streaming. Issues on reliability modeling and analysis are becoming more and more important. Chapter 5 formulated a reliability model to estimate the service reliability as a measure of the system performance with service quality considerations. With this reliability model, the performance of service provided by the P2P media streaming system could be obtained easily with information on the internet conditions and user profiles, which can be collected from survey or database of some public agencies.

As the condition of internet is highly dynamic over different times of the day, further analysis on the reliability modeling of the P2P media streaming system is proposed to account for the time effects. The universal generating function (UGF) is then used as the method to calculate the reliability. Two examples are given to illustrate the two models and analysis proposed.

Buffer techniques are commonly applied to store a few segments of media data ahead to hide transient extra delays in packet arrivals, improving the performance of the P2P media streaming systems (Hefeeda and Bhargava, 2003, and Zhang *et al.*, 2005). In chapter 5, we further build a reliability model to take into account the effects of the buffer technique on P2P system reliability. The real performance of the P2P media streaming system are better than what we concluded in the earlier part of this dissertation because of the application of the replication scheme in the real systems. A numerical example is used to illustrate the calculating process.

Besides hardware systems, faults in software systems may also cause computing systems to fail. It is important to estimate software reliability accurately in assessing computing system reliability. In order to apply the models to predict the reliability of the component, the parameters of the models need to be known or estimated. Chapter 6 studies the uncertainty problems in reliability modeling at both component-level and system-level. It not only addresses the uncertainty problem using the Bayesian Approach (BA), but more importantly solves the challenges for the dearth of data by embedding the Maximum-Entropy Principle (MEP) into the BA. By using MEP with BA, the expert knowledge, historical data from similar experiments and developmental environments can thus be involved in analyzing the uncertainty and for compensating insufficient failure data.

After exploring the uncertainty for software component, chapter 6 further extends the uncertainty analysis to more complicated systems that contain numerous components, each with its own respective distributions and uncertain parameters. A Monte Carlo approach is proposed to solve it. This method is broadly applicable for many systems that can be modeled with different modeling tools. The approach is then illustrated with a case of three-module software on a Markov model, and another case of a grid service reliability (a type of large-scale system) based on Graph theory. These examples with distinguished characteristics exhibit the generality and effectiveness of the MC approach to analyze not only simple module-based systems but also complicated systems with numerous uncertain parameters.

This approach is further illustrated with a recently published model (Dai & Levitin, 2006) for large-scale system reliability. Adopting this approach allows the relaxation of the assumptions of constant parameters in communication speed and processing speed, thus improving the practicality of the model. This demonstrates another novel application of the proposed uncertainty analysis. The results show that the improved model accounting for the uncertainty factors is more realistic and more reasonable than the original model. Moreover, the improved model can further provide the percentiles and variance that exhibit the confidence range to the practitioners about the model's output.

Bayesian approach can also be applied to reliability models of WVS for uncertainty analysis and parameter estimation. For distributed detection system, as a special type of weighted voting system, its reliability is defined as the probability the system provides correct output given corresponding input, which is determined by the accuracy and number of the local detectors, the threshold factor, and etc. These uncertain parameters are subject random variation as the performance of distributed detection system degrades and the detectors fail at certain rate. To calculate the system reliability, we have to estimate these system parameters precisely. In chapter 7, a Bayesian approach was presented to estimate the unknown parameters from the scarce data and quantify the uncertainty on the system reliability by measure of variance. A simulation was conducted as well to calculate the effect on the system reliability from the uncertainty of the parameters. An example was applied to illustrate the parameter estimation by Bayesian approach. Monte Carlo simulation was applied to estimate the effect on the system reliability on the uncertainty of parameters as the system reliability is hard to estimate by analytical method due to the combinatorial complexity of the problem.

So far, this thesis mainly discussed reliability modeling and analysis of different computing systems. Another important issue on resource allocation strategy to computing systems is studied in chapter 8. In the strategy, three important factors were considered: reliability importance measures, cost coefficients, and intentional attacks they may suffer. By providing the proof, we found a sufficient and necessary condition to judge whether the optimal strategy is reached. A numerical example is also presented to illustrate the process of calculating α for each component.

9.2 Future Work

Research on computing system reliability is very active nowadays. Following parts illustrate some possible directions which have great potential to develop based my current work introduced in proceeding sections.

The future research on the reliability analysis of weighted voting systems can follow two possible directions. Firstly, one important assumption in all the models above is the independence between voting units. However, it is very common that the decision of one voting unit affects the decision making of another unit. An area for further research is to formulate the voting system problems with dependent voting units. Secondly, the assumption that the output of each unit satisfies normal distribution makes the computation manageable as it is very easy to obtain the distribution of the sum of output of independent voting units. In practice, however, the distribution may not be normal, and hence it maybe hard to find the distribution function by analytical way. The future research can focus on finding an approximation function to approximate the probability of the entire system or the reliability of the entire system.

Possible future research on peer-to-peer computing system reliability can be summarized in the following. Firstly, the media streaming data is transmitted in packages of a certain volume each and not continuously as assumed in this dissertation. The media data comes in packages and can also be lost in packages. The reliability function hence is a discrete problem. This can be studied in our further research on the reliability analysis of the P2P media streaming systems. Secondly, in this model, we did not consider the stochastic characteristics of the individual peers. In practice, every peer is prone to connect and disconnect to the P2P networks when the networks condition is excellent and bad respectively, so an arriving user may follow stochastic process in connecting to the P2P networks. The stochastic theory applied to analyze the state of each individual peer will take into account the dynamic property, and this can make the models more realistic as well as more complicated. Many network techniques are applied in the real time media streaming network systems to make the services efficient and reliable, such as cache technique, replication of data and advanced search algorithms. These techniques are critical to the reliability and performance of the P2P services. Much more work can be done on the reliability analysis when we consider one or some techniques are considered.

In chapter 3 and 4, Monte Carlo simulation method is applied to derive the reliability of weighted voting systems as the structure of this kind of systems is very complex and it is very hard to find the analytical solution. Although the Monte Carlo simulation method is accurate, further work can be done to study the efficiency of the estimators used in our model. Moreover, to extend the work in chapter 7, research on the parameter estimation on threshold factor can be done. However, the effect on the system reliability of uncertainty from the threshold factor is much more complex to estimate. This is a possible direction to conduct future research.

Although the factor of attack from outsiders is considered in chapter 8 as an important part in calculating α , it is still a long way to understand clearly how intentional attack will affect the resource allocation strategy, especially to complex systems such as grid computing systems. The real situation of attacking and defending is much more complicated even though many researchers use game theory and other technique to model. Further analysis on the effect from attacker's decision on resource allocation strategy is critical and necessary to carry on.

Among all the chapters in this thesis, model validation have not been done in details because real world data of the practical systems is expensive to obtain. However, model validation is important for building reliability models. In future, some research can be potentially done on model validation to see how our models fit the real situation. Some improvement on the models can possibly be done based the observation from model validation.

REFERENCES

[1]. Abbas, A.E., 2006. Entropy methods for joint distributions in decision analysis, *IEEE Transactions on Engineering Management*, vol. 53, no. 1, pp. 146-159.

[2]. Adams, T., 1996. Total variance approach to software reliability estimation, *IEEE Transactions on Software Engineering,* vol. 22, no. 9, pp. 687-688.

[3]. Agarwal, H. and Renaud, J., 2004. Reliability based design optimization using response surfaces in application to multidisciplinary systems, *Engineering Optimization*, vol. 36, no. 3, 291-311.

[4]. Anderson, D.P., Cobb, J., Korpela, E., Matt, L. and Werthimer, D., 2002. Seti@home: an experiment in public-resource computing. Communications of the ACM, 45(11), 56-61.

[5]. Aneja, Y.P., Chandrasekaran, R., and Nair, K. P. K., 2004. Minimal-cost system reliability with discrete-choice sets for components. IEEE Transactions on Reliability, 53, 1, 71-76.

[6]. Azaiez, N. and Bier, V., 2007. Optimal resource allocation for security in reliability systems. *European Journal of Operational Research*, 181(2), 773-783.

[7]. Berger, J., 2006. The case for objective Bayesian analysis, *Bayesian Analysis*, vol. 1, no. 3, pp. 385-402.

[8]. Berger, J., 1985. *Statistical Decision Theory and Bayesian Analysis*, Springer-Verlag.

[9]. Berger, A.L., Della, S.A., Della, V.J., 1996. A maximum entropy approach to natural language processing, *Computational Linguistics*, vol. 22, no. 1, pp. 39-71.

[10]. Bernardo, J. and Smith, A., 1994. *Bayesian Theory*. Chichester, U.K.: Wiley.

[11]. Bier, V.M., Nagaraj, A., and Abhichandani, V., 2005. Protection of simple series and parallel systems with components of different values, Reliability Engineering and System Safety, 87, 315-323.

[12]. Bier, V., Oliveros, S., and Samuelson, L., 2005. Choosing what to protect: Strategic defensive allocation against an unknown attacker. *Working paper*.

[13]. Bier, V., Gratz, E. R., Haphuriwat, N. J., Magua, W., and Wierzbicki, K. R., 2007. Methodology for identifying near-optimal interdiction strategies for a power transmission system. *Reliability Engineering and System Safety*, 92, 1155-1161.

[14]. Bier, V., Nagaraj A., and Abhichandani, V., 2005. Protection of simple series and parallel systems with components of different values. *Reliability Engineering and System Safety* 87, 315-323.

[15]. Bischofs,L., Giesecke, S., Gottschalk,M., Hasselbring,W., Warns, T., and Willer, S., 2006. Comparative evaluation of dependability characteristics for peer-to-peer architectural styles by simulation. *Journal of Systems and Software*, 79, 1419–1432.

[16]. Chang, Y. R., Amari, S. V., and Kuo, S. Y., 2004. Computing failure frequencies and reliability importance measures using OBDD. *IEEE Transactions on Computers*, 53, 1, 54-68.

[17]. Charles Elgbede, O., Chu, C. B., Adjallah, K. H., and Yalaoui, F, 2003. Reliability allocation through cost minimization, *IEEE Transactions on Reliability*, 52 (1), 106-111.

[18]. Cline, M. G., 1997. Concepts and terminology of apical dominance. *American Journal of Botany*, 84 (9), 1064-1069.

[19]. Coit, D.W., Jin, T.D., and Wattanapongsakorn, N., 2004. System optimization with component reliability estimation uncertainty: a multi-criteria approach*,* *IEEE Transactions on Reliability*, Vol 53, No. 3, 369-380.

[20]. Coit, D.W., and Smith, A.E., 1996. Penalty guided genetic search for reliability design optimization. *Computers and Industrial Engineering*, 30(4), 895—904.

[21]. Cremonini, M., and Nizovtsev, D., 2006. Understanding and influencing attackers' decisions: implications for security investment strategies.

[22]. Cui, L.R., Kuo, W., Loh, H.T. and Xie, M., 2004. Optimal allocation of minimal & perfect repairs under resource constraints, *IEEE Transactions on Reliability*, VOL. 53, No. 2, 193-199.

[23]. Cunha, J.C., Rana, O.F, and Medeiros, P.D., 2005. Future trends in distributed applications and problem-solving environments. *Future Generation Computer Systems*, 21, 843-855.

[24]. Dai, Y.S. and Levitin, G., 2006. Reliability and performance of tree-structured grid services. *IEEE Transactions on reliability*, 55 (2), 337-349.

[25]. Dai, Y.S., Xie, M., and Poh, K.L., 2006. Reliability of grid service systems. *Computers & Industrial Engineering*, 50, 130-147.

[26]. Dai, Y.S., and Levitin, G., 2006. Reliability and performance of tree-structured grid services. *IEEE Transactions on Reliability*, vol. 55, no. 2, pp. 337-349.

[27]. Dai, Y.S., Xie, M., Poh, K.L., and Liu, G.Q., 2003. A study of service reliability and availability for distributed systems, *Reliability Engineering and System Safety*, vol. 79, pp. 103-112.

[28]. Dai, Y.S., Xie, M., Poh, K.L., and Yang, B., 2003. Optimal testing-resource allocation with genetic algorithm for modular software systems. *Journal of Systems and Software,* vol. 66, pp. 47-55.

[29]. Dai, Y.S., Xie, M., Long, Q., and S.H. Ng, 2005. Uncertainty analysis in reliability modeling based on Bayesian analysis. The 10th Annual International Conference on Industrial Engineering Theory, Applications and Practice, Florida, 468-473.

[30]. DeGroot, M.H., Schervish, M.J., 2002. *Probability and Statistics*, Boston: Addison-Wesley.

[31]. Dorigo, M., 2001. Ant Algorithms solves difficult optimization problems, *Advances in Artificial Life, Proceedings of the Sixth European Conference on Artificial Life*, LNAI 2159, Springer-Verlag, pp. 11-22.

[32]. Eyink, G.L., Kim, S., 2005. A maximum entropy method for particle filtering. *Journal of Statistical Physics*, vol. 123, no. 5, pp. 1071-1128.

[33]. Foster, I., and Imanitchi, A., 2003. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. Proceeding. Peer-to-Peer Systems II: 2nd Int'l Workshop (IPTPS '03), LNCS 2735, Springer-Verlag.

[34]. Freixas, J., and Puente, M. A., 2002. Reliability importance measure of the components in a system based on semivalues and probabilistic values. *Annals of Operations Research*, 109, 331-342.

[35]. Gen M. and Cheng, R.W., 2000. Genetic Algorithms and Engineering Optimization. *John Wiley & Sons, Inc.*

[36]. Goel, A.L., Okumoto, K., 1979. Time dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability,* vol. 28, pp. 206-211.

[37]. Goldstein, M., 2006. Subjective Bayesian analysis: principles and practice. *Bayesian Analysis*, vol. 1, no. 3, pp. 403-420.

[38]. Gong, L., 2002. Peer-to-peer networks in action. *IEEE Internet Computing*, 5 (1), 37-39.

[39]. Guo, T.H., Merrill, W., and Duyar, A., 1991. A distributed fault-detection and diagnosis system using on-line parameter estimation. *IFAC Distributed Intelligence Systems*, Virginia, USA, 221-226.

[40]. Haverkort, B.R., Meeuwissen, A.M.H., 1995. Sensitivity and uncertainty analysis of Markov-reward models. *IEEE Transactions on Reliability,* vol. 44, no. 1, pp. 147-154.

[41]. Hefeeda, M., and Bhargava, B.K., 2003. On-demand media streaming over the internet. *Future Trends of Distributed Computer Systems*.

[42]. Hefeeda, M., Habib, A., Botev, B., Xu, D., and Bhargava, B. 2003. PROMISE: Peer-to-peer media streaming using CollectCast. *In Proc. of ACM Multimedia 2003, 45–54, Berkeley, CA, USA*.

[43]. Holmes, D.E., 2006. Toward a generalized Bayesian network. *American Institute of Physics Conference Proceedings on Bayesian Inference and Maximum Entropy Methods In Science and Engineering*, vol. 872, pp. 195-202.

[44]. Hsieh, C., 2003. Optimal task allocation and hardware redundancy policies in distributed computing systems, Computing, *Artificial Intelligence and Information Technology*, 147, 430-447.

[45]. Hsieh, C. and Hsieh, Y., 2003. Reliability and cost optimization in distributed computing systems. *Computers & Operations Research* 30, 1103-1119.

[46]. Isada, Y. James, R., and Nakagawa, Y., 2005. An approach for solving nonlinear multi-objective separable discrete optimization problem with one constraint. *European Journal of Operational Research*, 160, 503-513.

[47]. Jaynes, E. T., 1963. Information theory and statistical mechanics. *Statistical Physics,* pp. *181-218*.

[48]. Jelinski, Z., Moranda, P.B., 1972. Software reliability research, In: Freiberger W. (ed), *Statistical Computer Performance Evaluation*, New York: Academic Press, pp. 465-497.

[49]. Jewell, W.S., 1985. Bayesian extensions to a basic model of software reliability, *IEEE Transactions on Software Engineering,* vol. SE-11, no. 12, pp. 1465-1471.

[50]. Johnston, W., 2006. Optimal allocation of reliability tasks to mitigate faults during system development. IMA Journal of Management Mathematics, 17, 159-169.

[51]. Kapur, J., 1989. *Maximum-Entropy Models in Science and Engineering*, John Wiley & Sons.

[52]. Kim, S., Bastani, F.B., Yen, I.L., Chen, I.R., 2004. Systematic reliability analysis of a class of application-specific embedded software framework, *IEEE Transactions on Software Engineering*, vol. 30, no. 4, pp. 218-230.

[53]. Korczak, E., Levitin, G., Haim, H.B., 2005. Survivability of series-parallel systems with multilevel protection. *Reliability Engineering & System Safety*, 90 (1), 45-54.

[54]. Kuo, W., and Prasad, V.R., 2000. An annotated overview of system reliability optimization, *IEEE Transactions on Reliability*, Vol. 49, No. 2, 176-187.

[55]. Kuo, W., Prasad, V.R., Tillman, F.A. and Hwang, C.L., 2000. *Optimal Reliability Design: Fundamentals and Applications*: Cambridge University Press, 2000.

[56]. Kuo, W. and Zuo, M.J., 2003. *Optimal Reliability Modeling: Principles and Applications*. John Wiley & Sons, Inc.

[57]. Kurowicka, D., Cooke, R., 2006. *Uncertainty Analysis with High Dimensional Dependence Modeling*. Wiley.

[58]. Latif-Shabgahi, G., Bass, J.M., Bennett, S., 2004. Taxonomy for software voting algorithms used in safety-critical systems. *IEEE Transactions on Reliability* 53 (3): 319-328.

[59]. Leemis, L.M., 1995. *Reliability - Probabilistic Models and Statistical Methods*, Prentice Hall, Inc., Englewood Cliffs, New Jersey.

[60]. Leuf, B., 2002. *Peer-to-peer Collaboration and Sharing over the Internet*. Addison-Wesly.

[61]. Levitin, G., 2000. Multistate Series-parallel system expansion scheduling subject to availability constraints, *IEEE Transactions on Reliability*, Vol. 49, NO. 1, 71-79.

[62]. Levitin, G., 2001a. Redundancy optimization for multi-state system with fixed resource-requirements and unreliable sources, *IEEE Transactions on Reliability*, vol 50, no. 1, 52-59,

[63]. Levitin, G., 2001b. Analysis and optimization of weighted voting systems consisting of voting units with limited availability. Reliability Engineering and System Safety 73 91-100.

[64]. Levitin, G., 2002a. Evaluating correct classification probability for weighted

voting classifiers with plurality voting. European Journal of Operational Research 141, 596-607.

[65]. Levitin, G., 2002b. Asymmetric weighted voting systems. Reliability Engineering and System Safety 76, 205-12.

[66]. Levitin, G., 2003. Threshold optimization for weighted voting classifiers. Naval Research Logistics 50, 322-44.

[67]. Levitin, G., 2004. Maximizing survivability of vulnerable weighted voting system. Reliability Engineering and System Safety 83, 17-26.

[68]. Levitin, G., 2005a. The Universal Generating Function in Reliability Analysis and Optimization. Springer.

[69]. Levitin, G., 2005b. Universal generating function in reliability analysis and optimization*,* Springer-Verlag, London.

[70]. Levitin, G., 2005c. Weighted voting systems: reliability versus rapidity. Reliability Engineering and System Safety, 89, 177-184.

[71]. Levitin, G., 2007. Optimal defense strategy against intentional attacks. IEEE Transactions on Reliability, 56, 1, 148-157.

[72]. Levitin, G., and Lisnianski, A., 2001a. A new approach to solving problems of multi-state system reliability optimization, *Quality and Reliability Engineering International*, 17, 93-104.

[73]. Levitin, G. and Lisnianski, A., 2001b. Reliability optimization for weighted voting system. Reliability Engineering and System Safety 71, 131-8.

[74]. Levitin, G., Xie, M., and Zhang, T.L., 2007. Reliability of fault-tolerant systems with parallel task processing. European Journal of Operational Research, 117 (1), 420-430.

[75]. Levitin, G., and Dai, Y.S., 2007. Service reliability and performance in grid system with star topology. Reliability Engineering and System Safety, 92 (1), 40-46.

[76]. Li, D., Sun, X., and Mckinnon, K., 2005. An exact solution method for reliability optimization in complex systems, *Annals of Operations Research*, 133, 129-148.

[77]. Liang, Y.C., and Smith, A.E., 1999. An ant system approach to redundancy allocation, *In Proceeding of the 1999 congress on evolutionary computation*, 1478-1484.

[78]. Liang Y.C. and Smith, L. E., 2004. An ant colony optimization algorithm for the redundancy allocation problem (RAP), *IEEE Transactions on Reliability*, VOL. 53, No. 3, 417-423.

[79]. Lisnianski, A. and Levitin. G., 2003. *Multi-state System Reliability. Assessment, Optimization and Applications*. World Scientific, Singapore.

[80]. Liu, X., Wang, J. and Vuong, S.T., 2006. A peer-to-peer framework for cost-effective on-demand media streaming. *Consumer Communications and Networking Conference*, 2006. CCNC 2006. 2006 3rd IEEE, 1, 314-318.

[81]. Lo, J.H., and Huang, C.Y., 2006. An integration of fault detection and correction process in software reliability analysis. *Journal of Systems and Software*, 79, 1312-1323.

[82]. Lyu, M., Rangarajan S., and Moorsel, A., 2002. Optimal allocation of test resources for software reliability growth modeling in software development. *IEEE Transactions on Reliability*, Vol. 51, No. 2, 183-192.

[83]. Ma, H., Yen, I.L., Zhou, J., and Cooper, K., 2006. QoS analysis for component-based embedded software: Model and methodology. *Journal of Systems and Software*, 79, 859-870.

[84]. Maniezzo, V., and Carbonaro, A., 2001. Ant colony optimization: an overview. *Essays and Surveys in Metaheuristics,* Kluwer Academic Publishers.

[85]. Marseguerra, M and Zio, E., 2005. Basics of genetic algorithms with application to system reliability and availability optimization. *Working Paper*.

[86]. Marseguerra, M., Zio, E. and Podofillini, L., 2004. Optimal reliability/availability of uncertain systems via multi-objective genetic algorithms. *IEEE Transactions on Reliability*, VOL. 53, No. 3, 424-434.

[87]. Masera, M., 1987. Uncertainty propagation in fault tree analyses using lognormal distributions. *IEEE Transactions on Reliability,* vol. R-36, no. 1, pp. 145-149.

[88]. Meng, F. C., 2004. Comparing Birnbaum importance measure of system components. *Probability in the Engineering and Informational Sciences*, 18, 237-245.

[89]. Mettas, A., 2000. Reliability allocation and optimization for complex systems. *PROCEEDINGS Annual RELIABILITY and MAINTAINABILITY Symposium*, Los Angeles, USA.

[90]. Michalewicz, Z., 1992. *Genetic Algorithm+Data Structure= Evolution Programs*, Springer-Verlag, New York.

[91]. Miller, K.W., Morell, L.J., Noonan, R.E., Park, S.K., Nicol, D.M., Murrill, B.W., Voas, J.M., 1992. Estimating the probability of failure when testing reveals no failures, *IEEE Transactions on Software Engineering*, vol. 18, no. 1, pp. 33-43.

[92]. Myrtveit, I., Stensrud, E., Shepperd, M., 2005. Reliability and validity in comparative studies of software prediction models. *IEEE Transactions on Software Engineering*, vol. 31, no. 5, pp. 380-391.

[93]. Nahas, N. and Nourelfath, M., 2005. Ant system for reliability optimization of a series system with multiple choice and budget constraints. *Reliability Engineering and System Safety,* 87(1) 1-12.

[94]. Naim, A., Kam, M., and Farsaie, A., 1991. On-line estimation of probabilities for Bayesian distributed detection. *IFAC Distributed Intelligence Systems*, Virginia, USA, 207-213.

[95]. Nordmann, L., and Pham, H., 1998. Weighted voting systems. *IEEE Transaction on Reliability* 48 (1), 42-9.

[96]. Nourelfath, M, and Dutuit, Y., 2004. A combined approach to solve the redundancy optimization problem for multi-state systems under repair policies, *Reliability Engineering and System Safety*, 86, 205-213.

[97]. O'Connor, P.D.T., 1995. Quantifying uncertainty in reliability and safety studies, *Microelectronics and Reliability,* vol. 35, no. 9-10, pp. 1347-1356.

[98]. Painton, L., and Campbell, J., 1995. Genetic Algorithms in the Optimization of System Reliability. *IEEE Transactions on Reliability, Special Issue on Design*, 44 (2), 172-178.

[99]. Parhami, B., 1994. Threshold voting is fundamentally simpler than plurality voting. International Journal of Reliability*, Quality & Safety Engineering* 1 (1) 95-105.

[100]. Pham, H., 2000. *Software Reliability*, Singapore: Springer-Verlag.

[101]. Piotrowski, T., Banerjee, S., Bhatnagar, S., Ganguly, S., and Izmailov, R., 2006. Peer-to-peer streaming of stored media: the indirect approach. *SIGMetrics/Performance'06*, June 26–30, 2006, Saint Malo, France.

[102]. Prasad, V., and Kuo, W., 2000. Reliability optimization of coherent systems, *IEEE Transactions on Reliability*, VOL. 49, No. 3, 323-320.

[103]. Ramirez-Marquez, J., Coit, D.W. and Konak, A., 2004. Redundancy allocation for series-parallel systems using a max-min approach. *IIE Transactions*, 36, 891-898.

[104]. Ramirez-Marquez, J., and Coit, D.W., 2004. A heuristic for solving the redundancy allocation problem for multi-state series-parallel systems, *Reliability Engineering and System Safety*, 83, 341-349.

[105]. Ravi, V., Reddy, P.J. and Zimmermann, H. J., 2000. Fuzzy global optimization of complex system reliability. *IEEE Transactions on Fuzzy Systems*, 8(3), 241-248.

[106]. Robert, C., 1994. *The Bayesian Choice: A Decision Theoretic Motivation*. Springer-Verlag.

[107]. Romera, R., Valdes, J. and Zequeira, R., 2004. Active redundancy allocation in Systems, *IEEE Transactions on Reliability*, 53(3), 313-318.

[108]. Rowstron, A., and Druschel, P., 2001. Storage management in past, a large-scale, persistent peer-to-peer storage utility. *In Proceeding of 18th ACM Symposiumon Operating Systems Principles*, Chateau Lake Louise, Banff, Canada.

[109]. Ryoo, H.S., 2005. Robust metaheuristic algorithm for redundancy optimization in large-scale complex systems, *Annals of Operations Research*, 133, 209-228.

[110]. Saroiu, S., Gummadi, P., and Gribble, S.D., 2002. A measurement study of peer-to-peer file sharing systems. *Proceeding of MMCN*' 02, January.

[111]. Savsar, M., and Al-Anzi, F. S., 2005. Reliability of data allocation on a centralized service configuration with distributed servers. *The Computer Journal*, 49(3), 258-267.

[112]. Schnier, T. and Yao, X., 2003. Using negative correlation to evolve fault-tolerant circuits, *In Proc. Of the 5th International Conference on Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science*, Vol. 2606, Springer-Verlag, pp. 35-46.

[113]. Selby, R.W., 2005. Enabling reuse-based software development of large-scale systems, *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 495-510.

[114]. Shannon, C.E., 1948. A mathematical theory of communication, *The Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656.

[115]. Shi, W.S., and Mao, Y.G., 2006. Performance evaluation of peer-to-peer Web caching systems. *Journal of Systems and Software*, 79, 714–726

[116]. Soundappan, P, Nikolaidis, E, Haftka, R.T,, Grandhi, R, Canfield, R, 2004. Comparison of evidence theory and Bayesian theory for uncertainty modeling, *Reliability Engineering & System Safety*, vol. 85, no. 1-3, pp. 295-311.

[117]. Srinivas, M. and Patnaik, L.M., 1994. Genetic algorithms: A Survey. *Computer*, 27 (6), 17-26.

[118]. Steinmetz, R., and Wehrle, K., 2005. *Peer-to-Peer Systems and Applications*. Springer-Verlag, Berlin, Germany.

[119]. Taylor, T., Marshall, M., and Neumann, E., 2001. Developing a reliability improvement strategy for utility distribution systems, *Transmission and Distribution Conference and Exposition*, 1, 444-449.

[120]. Tenney, R.R., and Sandell, N.R., 1981. Distributed detection networks. *IEEE Transactions on Aerospace and Electronics Systems, AES-17*, 501-510.

[121]. Thierens, D., 2002. Adaptive Mutation Rate Control Schemes in Genetic Algorithms. *Proceedings of the 2002 IEEE World Congress on Computational Intelligence: Congress on Evolutionary Computation,* 980-985.

[122]. Tian, H.R., Zou, S.H., Wang, W.D., and Cheng, S.D., 2006. Constructing efficient peer-to-peer overlay topologies by adaptive connection establishment. *Computer Communications*, 29 (17), 3567-3579.

[123]. Tseng, C.Y., 2005. Entropic criterion for model selection, *Physica A: Statistical and Theoretical Physics*, vol. 370, no. 2, pp. 530-538.

[124]. Trivedi, K.S., 1982. *Probability and Statistics with Reliability, Queuing, and Computer Applications*, Englewood, NJ: Prentice-Hall.

[125]. Tu, Y.C., Sun, J.Z., Hefeeda, M., and Prabhakar, S., 2005. An Analytical Study of Peer-to-Peer Media Streaming Systems. *ACM Transactions on Multimedia Computing, Communications and Applications,* 1(4), 354-376.

[126]. Wattanapongsakorn N. and Levitan, S.P., 2004. Reliability optimization models for embedded systems with multiple applications, *IEEE Transactions on Reliability*, VOL 53, No. 3, 406-416.

[127]. Wen, J.H., Huang, K.T., Yang, C.Y., and Tsai, T.C, 2006. Timeslot-sharing algorithm with a dynamic grouping for WDM broadcast-and-select star networks. *Journal of Systems and Software*, 79, 1110-1119.

[128]. Wooff, D.A., Goldstein, M., Coolen, F.P.A., 2002. Bayesian graphical models for software testing, *IEEE Transactions on Software Engineering,* vol. 28, no. 5, pp.510-525.

[129]. Wu, S., 2005. Joint importance of multistate systems. Computers *& Industrial Engineering*, 49, 63-75.

[130]. Xie, M., Dai, Y.S., Poh, K.L., 2004. *Computing System Reliability: Models and Analysis*, Kluwer Academic Publishers: New York, NY, U.S.A.

[131]. Xie, M., Hong, G.Y., Wohlin, C., 1997. A study of the exponential smoothing technique in software reliability growth prediction, *Quality and Reliability Engineering International*, vol. 13, no. 6, pp. 347-353.

[132]. Xie, M.G., Pham, H., 2005. Modeling the reliability of threshold weighted voting systems. *Reliability Engineering and System Safety* 87, 53-63.

[133]. Xu, D., Hefeeda, M., Hambrusch, S., and Bhargava, B., 2002. On peer-to-peer media streaming. *In Proc. of IEEE ICDCS'02*, Vienna, Austria.

[134]. Yacoub, S., 2003. Analyzing the behavior and reliability of voting systems comprising tri-state units using enumerated simulation. *Reliability Engineering and System Safety*, 81 (3.2): 133-145.

[135]. Yang, B., Xie, M., 2000. A study of operational and testing reliability in software reliability analysis, *Reliability Engineering & System Safety*, vol. 70, pp. 323-329.

[136]. Yao, X. and Liu, Y., 1998. Making use of population information in evolutionary artificial neural networks, *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 28 (3), 417-425.

[137]. Yeh, C.C., and Pui, L.S., 2005. On the frame forwarding in Peer-to-Peer multimedia streaming. *Proceedings of the ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming*, 1-10.

[138]. Yin, L., Smith, M.A.J., Trivedi, K.S., 2001. Uncertainty analysis in reliability modeling, *Proceedings of the Annual Reliability and Maintainability Symposium,* pp. 229-234.

[139]. Yin, L., Trivedi, K.S., 1999. Confidence interval estimation of NHPP-based software reliability models, *The International Symposium on Software Reliability Engineering,* pp. 6-11.

[140]. You, P. and Chen, T., 2005. An efficient heuristic for series-parallel redundant reliability problems, *Computers & Operations Research,* 32 2117-2127.

[141]. Yun, W.Y. and Kim, J.W., 2004. Multi-level redundancy optimization in series systems, *Computers & Industrial Engineering*, 46, 337-346.

[142]. Zhang, J.J., Liu, L., Pu, C. and Ammar, M., 2004. Reliable peer-to-peer end system multicasting through replication. *IEEE Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, 235-242.

[143]. Zhang, X., Liu, J.C., Li, B., and Yum. P., 2005. Coolstreaming/donet: A data-driven overlay network for efficient live media streaming. *In Proceedings of IEEE INFOCOM*, March 2005.

[144]. Zhao, J.H., Liu, Z., and Dao, M., 2007. Reliability optimization using multiobjective ant colony system approaches, *Reliability Engineering and System Safety,*92 (1), 109-120.

[145]. Zhao, R. Q. and Liu, B. D., 2003. Stochastic programming models for general redundancy optimization problems, *IEEE Transactions on Reliability*, 52(2), 181-191.

[146]. Zio, E., and Podofillini, L., 2006. Accounting for components interactions in the differential importance measure. *Reliability Engineering and System Safety*, 91, 1163-1174.