

國立交通大學

工業工程與管理學系

博士論文

整合性供應鏈績效評估與最佳化分析



An Analytical Framework for Supply Chain Performance Evaluation and Optimization

研究生：王風帆

指導教授：蘇朝墩、洪瑞雲

中華民國九十四年十一月

整合性供應鏈績效評估與最佳化分析

An Analytical Framework for Supply Chain Performance Evaluation and Optimization

研究生：王風帆

Student : Fong-Fan Wang

指導教授：蘇朝墩

Advisors : Prof. Chao-Ton Su

洪瑞雲

Prof. Ruey-Yun Horng

國立交通大學

工業工程與管理學系



A Dissertation

Submitted to Department of Industrial Engineering and Management
College of Management

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Industrial Engineering and Management

November 2005

Hsinchu, Taiwan, Republic of China

整合性供應鏈績效評估與最佳化分析

研究生：王風帆


指導教授：蘇朝墩

洪瑞雲

國立交通大學

工業工程與管理學系博士班

摘 要



本論文以系統整合角度提出多階物流供應鏈績效評估及最佳化模式。一般來說，多階物流供應鏈可概分為零物料供應、製造、配銷及運輸等供應子系統。以往學者在分析類似問題大多零散不完整，且傾向個別求解。例如已知市場需求，希望求解最適製造策略；或是假設配銷系統供應無限來求解最適庫存水準。本論文假設產品需求及製造環境為隨機分配，各供應子系統產能有限，以矩陣分析方法求出在長期穩定狀態下此供應鏈網路系統績效，即長期穩定作業成本及服務水準。探討各種可能市場需求及供給不確定因素型式對系統績效之影響，提出以馬可夫調適卜瓦松過程建構在此情況下各供應子系統隨機模式，並以矩陣幾何方法求解系統績效。本論文最後運用各種古典及現代尋優理論，配合所提出系統評估模式找出供應最佳庫存及產能水準，並考慮上游供貨不穩定對系統績效之影響，配合實例說明以此整合分析及最佳化模式找出不穩定供需環境下最穩健系統資源規劃。

關鍵詞：整合性供應鏈、矩陣分析方法、擬生死過程、馬可夫調適卜瓦松過程、巨集啟發式解法。

An Analytical Framework for Supply Chain Performance Evaluation and Optimization

Student : Fong-Fan Wang

Advisors : Dr. Chao-Ton Su

Dr. Ruey-Yun Horng

Department of Industrial Engineering and Management
National Chiao Tung University

Abstract

An integrated matrix analytic model is proposed to evaluate a multi-echelon supply chain (SC) from a systematic viewpoint. Generally speaking, a multi-echelon supply chain consists of four basic subsystems: raw material supply, production, distribution and transportation. In the past, most of the analyses of complex supply chain models seemed to be fragmented and tended to pursue local analysis. For example, some studies assumed market demands were known and tried to find the most adequate manufacturing strategies. Others assumed that supplies were unlimited and tried to find optimal distribution strategies to fight against demand uncertainty. In the first phase of this study, the market demand and supply process were assumed stochastic. Each SC contributor was then treated as a single server in a tandem queueing network. We developed an integrated matrix analytic model for the steady state performance evaluation in terms of total operating cost and customer service level for a fabricated SC problem. We also took upstream unavailability into the modeling process. Next, we investigated deeply into the impact of supply/demand uncertainties on SC performance. We modeled various uncertainty scenarios as Markov modulated Poisson processes (MMPP).

Using matrix geometric method we obtained the steady state performance measures and proposed solution measures for a test SC problem with make-to-order mode. Finally, based on the above-developed SC evaluation model, we investigated various optima seeking procedures. Our objective was to solve the optimal buffer and capacity allocation problems simultaneously such that the chain wide operation cost was minimized under pre-specified service level. Especially, we discussed the impact of unreliable supply on system performance.

Keywords: Integrated stochastic supply chain, Quasi-birth-and-death process, Matrix analytical method, Markov modulated Poisson process, Meta-heuristic method.



誌 謝

本論文得以順利完成，首先要感謝我的指導教授，清華大學工業工程與工程管理系蘇朝墩博士。無論在期刊投稿或是最後論文撰寫，蘇老師均給了我最大實質上幫助，謹致上最誠摯謝意。系上洪瑞雲老師的共同指導，亦衷心銘謝。感謝洪老師及本系巫木誠老師，大葉大學工業工程與科技管理系駱景堯老師、台北科技大學工業工程與管理系田方治老師在論文審查及口試期間寶貴意見，斧正了論文草稿中部份缺失。感謝學弟們在我論文投稿期間各方面的幫忙，尤其是楊健炘君在口試期間庶務上的幫助。最後感謝交大工業工程與管理系給我繼續深造機會！



Contents

Abstract	ii
List of tables	vii
List of figures	viii
Notation	ix
Chapter 1 Introduction	1
1.1 Overview	1
1.2 Motivation	6
1.3 Objectives	6
1.4 Scope and limitation	8
1.5 Organization	8
Chapter 2 Related works	9
2.1 Queueing network with planned inventory	9
2.2 Multi-echelon inventory control theories	10
2.3 Models related to uncertainties	12
2.4 An integrated framework for stochastic SC analysis	13
2.5 Optimization of a stochastic SC	15
Chapter 3 An integrated evaluation model	18
3.1 Introduction	18
3.2 Matrix analytic approach	19
3.2.1 The approximation model	21
3.2.2 The proposed approach	23
3.3 Implementation	28
3.3.1 A test problem	28
3.3.2 Numerical results	29
3.4 Discussion and sensitivity analysis	32
3.5 Extension	37
3.6 Concluding remarks	40
Chapter 4 Non-stationary arrival and unreliable service processes	43
4.1 Introduction	43
4.2 QBD process decomposition	44
4.3 MMPP modeling of uncertainties	47

4.4	Illustrative examples.....	54
4.5	Computation issue	63
4.6	Concluding remarks.....	64
Chapter 5	Optimal buffer and capacity design under service constraint.....	66
5.1	Introduction	66
5.2	Methodology.....	69
5.2.1	Flow Equivalent System.....	70
5.2.2	Meta-heuristic methods	70
5.3	Optimization	72
5.3.1	Tandem topology	72
5.3.2	A more general topology	86
5.4	Discussion.....	91
5.4.1	Penalty of BC model	91
5.4.2	Impact of unavailability.....	93
5.4.3	Impact of poor quality	96
5.5	Concluding remarks.....	99
Chapter 6	Conclusions	101
Appendix A	QBD process and simulation model.....	107
Appendix B	Moment derivations and related proofs.....	109
Appendix C	Matlab codes of algorithms	112
References	128

List of tables

Table 3.1 Approximation vs. simulation ($S_5 = 0$)	31
Table 3.2 Approximation vs. simulation ($S_5 \neq 0$)	32
Table 3.3 A two-stage system	34
Table 3.4 A four-stage system.....	34
Table 3.5 Approximation vs. simulation for the case where retailers use (r, q) policies.	38
Table 3.6 Approximation vs. simulation for cases with multi-server and breakdowns.....	39
Table 4.1 Traffic intensity ρ' under different uncertainty cases	58
Table 4.2 $E[O]$ for each design under different uncertainty cases.....	58
Table 4.3 Demand queue length for each design under different uncertainty cases.....	59
Table 4.4 Local and global optimal solutions under different uncertainties.....	60
Table 4.5 Approximation results obtained from GSPN for different uncertainty cases	62
Table 5.1 Optimal buffer allocation using CSA, and ENU (with minimum S_T)	84
Table 5.2 Optimal buffer allocation using CGA, CSA (no S_T restriction)	84
Table 5.3 Time performance of CSA, CGA and ENU.....	84
Table 5.4 Optimal workload allocation using CSA.....	85
Table 5.5 Optimal buffer and workload allocations using iterative method ($J=4$).....	85
Table 5.6 Optimal buffer and workload allocations using CSA.....	85
Table 5.7 Optimal buffer and workload allocations using CSA for larger SC.	86
Table 5.8 DOE setting for approximation validation of more general topology	90
Table 5.9 Approximation validation (1).	90
Table 5.10 Approximation validation (2).	90
Table 5.11 Optimal buffer and workload allocations for more general SC.....	91
Table 5.12 DOE setting for unavailability test.	95
Table 5.13 Comparison of regular and random unavailability mode.	95
Table 5.14 Impact of unavailability by different stages (worst case).....	96
Table 5.15 Impact of unavailability by different stages (best case).	96

List of figures

Fig. 1.1 A supply chain network.....	5
Fig. 1.2 An integrated framework for stochastic supply chain analysis.....	7
Fig. 3.1 A transition diagram of a Hyper-exponential distribution.....	26
Fig. 3.2 A multi-echelon SC with feedback consideration.....	29
Fig. 3.3 Sojourn time of stage 1 as a function of S1 for a two-stage PF with $\rho = 0.5$	35
Fig. 3.4 WIP as a function of S1 for a two-stage PF with $\rho = 0.5$	36
Fig. 3.5 $E[B]$ as a function of S1 for a two-stage PF with $\rho = 0.5$	36
Fig. 4.1 A multi-server supply network of production and distribution echelons.....	45
Fig. 4.2 A simple MMPP process.....	48
Fig. 5.1 Three basic congested supply systems.....	69
Fig. 5.2 Transformation from an assembly system into an $M/M/1$ FES.....	70
Fig. 5.3 Using convexity-based method to search a tandem SC.....	76
Fig. 5.4 A more general SC model.....	87
Fig. 5.6 Backorder level as a function of β under different cost structure.....	93
Fig. 5.7 Impact of δ on TC by different feedback location $S = (4, 4, 4, 1)$	97
Fig. 5.8 Impact of δ on SL by different feedback location $S = (4, 4, 4, 1)$	98
Fig. 5.9 Impact of δ on TC by different feedback location $S = (16, 16, 16, 4)$	98
Fig. 5.10 Impact of δ on SL by different feedback location $S = (16, 16, 16, 4)$	99

Notation

- L_j : Total lead-time of customer order at stage j
 D_i : The waiting time at stage i
 T_j : Processing time at stage j , including waiting and service times at stage j
 K_j : The outstanding customer order at stage j
 B_i : The backorder level of stage j recorded at stage i
 S_j : The stock level at stage j
 \tilde{Q} : Infinitesimal generator for the whole queueing system (customer demand queue)
 $Q^{(i)}$: Infinitesimal generator for case i -based server-repair process (server queue)
 λ : Average aggregate retailer demand arrival rate ($= \sum_{\forall i} \lambda_i$)
 u : Average processing rate for each server
 ξ : Average server up rate (= the inverse of mean-time-to-failure)
 γ : Average server down rate (= the inverse of mean-time-to-repair)
 ρ : Traffic intensity without supply unavailability consideration
 ρ' : Traffic intensity incorporating supply unavailability
 c_o : server operation cost per server per unit time
 c_r : server repair cost per server per unit time
 c_h : Inventory holding cost per unit and unit time
 c_b : Backorder cost per unit and unit time
 π : Stationary probability vector for server-repair queue, $Q^{(i)}$
 x : Stationary probability vector for customer queue, \tilde{Q}
 $E[L]$: Average demand queue length (including the one in service if there is any)
 $E[O]$: Average number of operative servers
 $E[RE]$: Average number of servers under repair
 $E[B]$: Average aggregated backorder level at retailer site
 m : Number of machines
 r : Number of repairman, $r \leq m$
 λ_r : Arrival rate for regular job
 λ_b : Breakdown cycle rate for breakdown job
 J : Total number of stages of the SC
 R : Total number of retailers

- u_j : Average processing rate for each server at stage j
 S_T : Total buffer of the SC
 (\bullet) : Lower bound for the decision variable of interest
 $(\bullet)^*$: Optimal solution for the performance measure of interest
 c_a^2 : Coefficient of variation of arrival process, which is a superposition process of both regular and breakdown jobs
 c_s^2 : Coefficient of variation of aggregated service process
 c_r^2 : Coefficient of variation of arrival of regular job
 c_u^2 : Coefficient of variation of service job
 c_b^2 : Coefficient of variation of arrival of breakdown job
 c_ξ^2 : Coefficient of variation of breakdown job
 c_γ^2 : Coefficient of variation of repair job
 p : probability of occurrences of regular job
 q : probability of occurrences of breakdown job
 L : Waiting line in the system
 L_q : Waiting line in the queue
 W_s : Waiting time in the system
 W_q : Waiting time in the queue
 δ : Feedback (Rework) rate due to imperfect quality
WIP : Intermediate inventory levels including input and output buffers at each stage, excluding the input buffer at the first supply stage and the ending inventory level for the last stage. $WIP =$ the sum of **WIP**
 $E[I]$: Expected Inventory level for aggregated retailer stage
 $E[B_j]$: Expected backorder level for stage/retailer j
 $E[I_j]$: Expected Inventory level for stage/retailer j
 $E[T_s]$: Expected service time
 TC : Total cost of operating the system
 SL_j : Service level for retailer j , defined as probability of customer for waiting time greater than t time units ($SL_j = \Pr(T > t)$, here Pr stands for probability)
 SL : (Average) system service level, defined as the arithmetic average of SL_j or as a single performance measure if retailers are aggregated
 β_j : Pre-specified service level for retailer j

β : Pre-specified service level

h_{wip} : Intermediate inventory cost vector

c_j : Service cost per work-unit and unit time at stage j



Chapter 1 Introduction

1.1 Overview

Supply chain management (SCM) was popular in the past years and seems continuing to be so in the future. The main reason of its popularity may be that it extends the traditional operational management in local shop floor to a global context. Basically there are two approaches to model a supply chain (SC), deterministic and stochastic. The former is adopted when all the operational parameters, such as demand arrival rate, processing rate, etc. are certain while the later is used when most of the parameters are uncertain. Material requirement planning (MRP) and lately enterprise resource planning (ERP) are perhaps the most widely used supply control methods to satisfy market demand under a deterministic operational environment. The performance is often judged by the generated supply plan that can deliver the right product with right quantity to right place in right time. Under stochastic environment, the SC performance can be evaluated through stochastic modeling to obtain the steady state system performance. No matter what modeling type the studied problem is, the performance of an SC is often measured by its integrated operational cost and achieved customer service level. It's well known that in order to save operational cost, two important factors must be addressed: inventory and moving (transportation) costs.

For analytic modeling of a stochastic SC, there are three well-known problem domains for a supply network: infinite buffer, finite-buffer and infinite buffer with planned inventories. The former two problems are suitable for make-to-order (MTO) supply mode, while the last is suitable for make-to-stock (MTS) supply mode. Open or closed queueing network (QN) model is suitable for solving the first kind of problem. Several models are developed to solve the second problem. Notice the blocking effect exists in the second problem and the solution process seems not so straightforward as the first problem. For the third problem, it's not paid special attention until in the past decade or so (Lee & Zipkin,

1992) (we abbreviate it as L & Z hereinafter). However, the MTS supply mode of the third type is a well-known industry application, therefore it shows imperative need for this research direction. It depends on the business process and other factors such as market, capital, physical limitation etc. to choose the suitable supply mode and the accompanied stochastic model for practical study.

In reality, an SC is often shown as a sophisticated supply network with complex operation logic. Under this concern, simulation seems to be the most often used method to analyze a realistic SC, especially when mathematic model is not available. However it becomes time-consuming to build an industry-scaled simulation model. Let's take a look at fig. 1.1, which is an SC network, composed of 4 basic SC functions: inbound logistics, manufacturing, outbound logistics and distribution. Inside each echelon, there may be several processing stages with serial or parallel configuration and the probability distribution for each service may be arbitrary (for ease of exposition, we only show serial case in fig. 1.1, see the square enclosed by the dotted lines). To let the complex interaction between SC players (contributors) become tractable, an adequate control scheme governing order/replenish behavior for each echelon is necessary if we regard inventory as the main concern. Unfortunately, we don't know what the optimal order/replenish policy for SC like fig. 1.1 is. Furthermore, the closed-form solution is usually unavailable (Boyacy and Gallego, 2001) even for the simplest tandem supply system with assumed constant supply lead-time. In this study, we do not try to answer these questions. Instead, we "assume" control policies at each supply stage are known in advance. For example, in "pull-type" control (which is suitable for the situation that the demand is unknown or stochastic), there are base-stock policy, reorder point, order quantity (r, q) policy, reorder and target level (s, S) policy, and KANBAN-card controlled policy etc. Among them, base-stock policy is widely used in industry (L & Z) owing to its simple control logic. It is suitable when economy of sale is not a concern. For example, there is no fixed set-up cost

in each ordering cycle.

To model the SC like fig. 1.1, we begin by learning the system dynamic of a base-stock controlled tandem supply system as appeared in (L & Z) and shown below:

$$I_j = [S_j - K_j]^+, B_j = [K_j - S_j]^+, 1 \leq j \leq J, \quad (1.1)$$

$$K_1 = N_1, N_j = K_j - B_{j-1} \text{ for } j > 1, \quad (1.2)$$

where $[x]^+ = \max(x, 0)$. J is the stage number of the tandem system. S_j is base-stock level at each stage. K_j is demand on order, and B_j is backorder level at stage j , and I_j is inventory level at stage j (here assume we have already break the original echelon boundary into a multi-staged tandem form). N_j is the input queue occupancy before each stage j (including the one being served if there is any). (1.2) is due to the property of the underlying system dynamic: $[demand\ on\ order] = [input\ queue\ occupancy] + [backorder\ level\ at\ the\ previous\ stage]$. We assume ample supply before the first stage, and therefore no backorder from external supply: $K_1 = N_1$. The difficult part of analyzing the above system is that the queueing network will not be a $M/M/1$ connected system when the planned inventories are added after each stage. Based on (1.1) and (1.2), several methods for performance evaluation of a base-stock controlled SC have been reported recently for the approximation of such SC like recursive method, squared coefficient variation (SCV) of departure process (which will be abbreviated as SCV method hereinafter), quasi-birth-and-death (QBD) method, and matrix computation method. Recursive method starts at stage 1 with $K_1 = N_1$, the distribution of N_1 is approximated to be that of $M/M/1$. Specifically

$$P(N_j = i) = \begin{cases} 1 - \rho_j, & i = 0, \\ \rho_j^i (1 - \rho_j), & i \geq 1. \end{cases} \quad (1.3)$$

Given K_j , it computes B_j by shift-truncation operation in (1.1). Then, it applies (1.2) to obtain $K_{j+1} = N_{j+1} + B_j$, which is just the convolution of product-form approximation. Motivated by the widely used approximation of the SCV of the departure process from a

standard queue (Buzacott and Shanthikumar, 1993), specifically $c_d^2 = (1 - \rho^2)c_a^2 + \rho^2 c_s^2$, SCV method uses the following approximation for the SCV of the departure process from output buffer of stage j ,

$$c_{dj}^2 = (1 - \rho_j^{2+S_j/2})c_{aj}^2 + \rho_j^{2+S_j/2}c_{sj}^2, \quad (1.4)$$

where c_{aj}^2 and c_{sj}^2 are respective SCV of inter-arrival and service times at stage j , $1 \leq j \leq J - 1$. Notice when $c_{aj}^2 = c_{sj}^2 = 1$, (1.4) becomes $c_{dj}^2 = 1$, which is the Markovian departure process of a standard $M/M/1$ queue. It then approximates the distribution of input queue occupancy (Buzacott and Shanthikumar, 1993, p76)

$$P(N_j = i) = \begin{cases} 1 - \rho_j, & i = 0, \\ \hat{\rho}_j^{i-1}(1 - \hat{\rho}_j)\rho_j, & i \geq 1, \end{cases} \quad (1.5)$$

where $\hat{\rho}_j = \frac{\rho_j(c_{ai}^2 + c_{si}^2)}{\rho_j(c_{ai}^2 + c_{si}^2) + 2(1 - \rho_j)}$ (Note here Allen-Cunneen approximation for a $GI/G/1$ queue is used to obtain the above formula). This method uses the property: $c_{aj}^2 = c_{d,j-1}^2$ to recursively find performance measures as the previous method. Specifically, it lets c_{a1}^2 be the SCV of the inter-arrival time of external demands. It then computes c_{d1}^2 by (1.4) and $K_1 = N_1$ (since $B_0 = 0$) by (1.2), whose distribution is known by (1.5). Given K_1 , it computes B_1 by (1.1), and then it moves to the next stage and recursively call the above procedure until stage J . Notice these two methods only differ in queue occupancy calculation between (1.3) and (1.5). Specifically (1.5) related to the SCV of departure process (1.4). QBD method tries to directly solve the whole SC system by approximating the input buffer to be finite number. Then it uses matrix geometric method (MGM) to solve the finite QBD. However, the computation becomes intractable when J is large, say $J > 4$. Instead, herein we use QBD process to decompose each queueing system in an SC and therefore get tractable results. In this study, our evaluation model belongs to the last

method, matrix computation. It is essentially equal to recursive method. However, it focuses on response times instead of queue occupancies. It implements the calculation with simple matrix-algebraic manipulations. Herein we combine it with our proposed QBD method and give it another name, matrix analytical method.

The optimization of the SC to obtaining strategic parameter setting for efficient material flow can be categorized as the following methods: classical derivative-based method, enumerative method, meta-heuristic method among others. These methods will be fully explored herein. In this work, we show that through adequate “transformation”, similar or more complex SC like fig. 1.1 can be tackled in adequate mathematic models. Specifically, in the first phase, we built the evaluation model for an SC by proposing the QBD modeling procedures for solving non-exponential, parallel processing, and single-server based distribution systems. We also discussed possible extension of the evaluation model with (r, q) controlled system dynamic. In the second phase we detailed the analysis of parallel processing under supply and demand uncertainties with the help of Markov-modulated Poisson process (MMPP) models. Finally, we investigated several optimization methods, classical and modern, which may be used in strategic optimization of an SC.

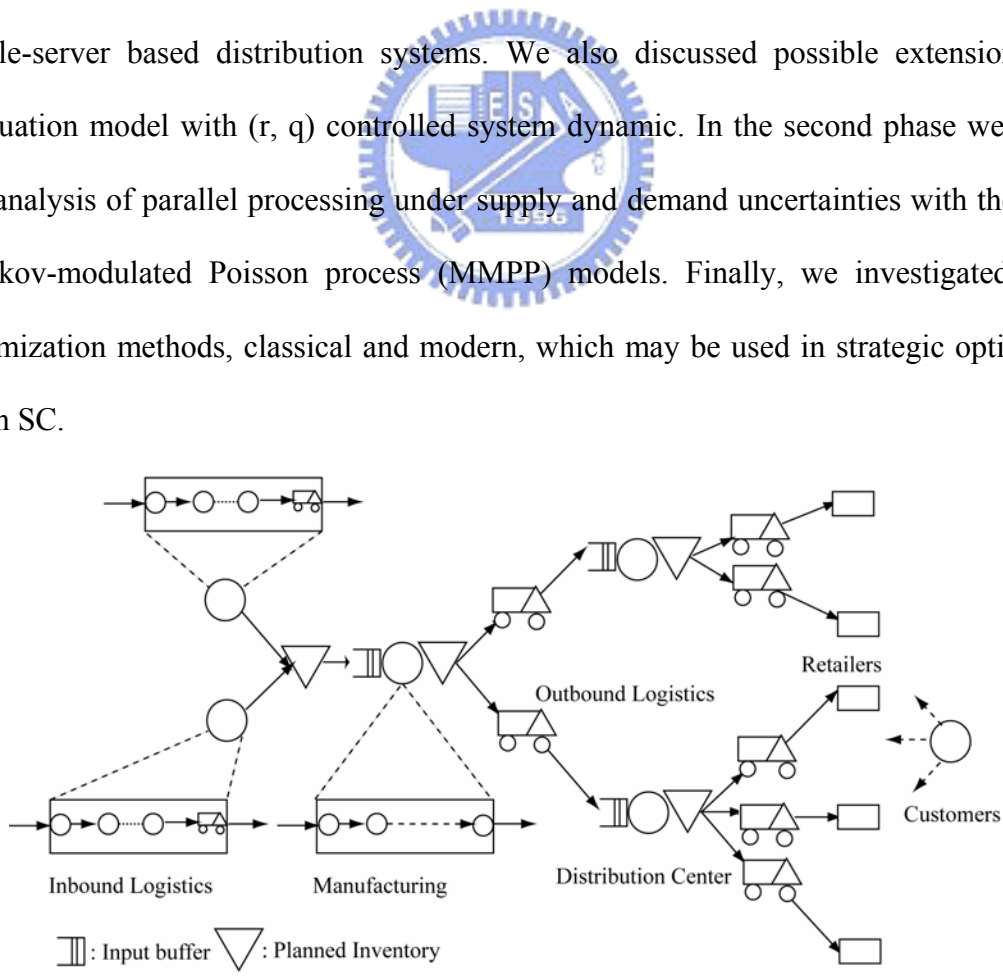


Fig. 1.1 A supply chain network.

1.2 Motivation

As is known, the operational environment is usually full of uncertainty factors. In this respect, the stochastic modeling of an SC in a systematic approach seems natural and necessary. This analytic direction focuses on infinite horizon; therefore it should be categorized as the strategic planning phase of an SC. Though there is already a large body of literature discussing several aspects in quantitative analysis of stochastic SCM (Tayur et al., 1999) such as supply contract, collaboration, operational management etc., several issues remain to be addressed. Among them, an integrated SC model is still lacking. The purpose of studying the behavior of integrated SC models is to quickly identify the pros and cons of an SC design in the long run. Usually the operational goal of an SC is to maintain a quick responding SC with minimum operating cost. However, owing to the sophisticated nature of an SC, most stochastic multi-echelon SC models only focus on an individual phase. Some models focus on the interaction between warehouse and retailer operations. Others focus on the study of the production phase. When the focus is on distribution system, the supply is usually assumed unlimited, and consequently supply time is assumed constant. This assumption neglects the processing variability at the supply system. Likewise, when the focus is on production system, there is no information about the subsequent material flow from the transportation to the end users. System performance changes due to the variation embedded in processing, transportation, and distribution phases is therefore ignored. This motivated us to propose a systematic design and analytic framework for an integrated SC.

1.3 Objectives

As depicted in fig. 1.1, a typical integrated SC model includes several operational phases such as production, distribution and transportation. In the past, simulation model

seems to be the only viable approach to analyze such complex supply chain topology. However, we show it's possible to develop a computation-efficient analytic model. The integrated analytic model is depicted in fig 1.2. Especially we will investigate the feasibility of our proposed QBD model to solve non-Markovian arrival and/or service processes and non-tandem supply structure (such as a distribution system). Based on the integrated system dynamics of such QN, optimization procedure will be able to be conducted based on (1.1) and (1.2). Since the studied problem is highly nonlinear inherent, meta-heuristic methods will be employed on general problem structures such as fig 1.1. Finally we will discuss the impact on the system performance due to the variation(s) of unreliable supply and/or demand in processing, transportation, and distribution stages.

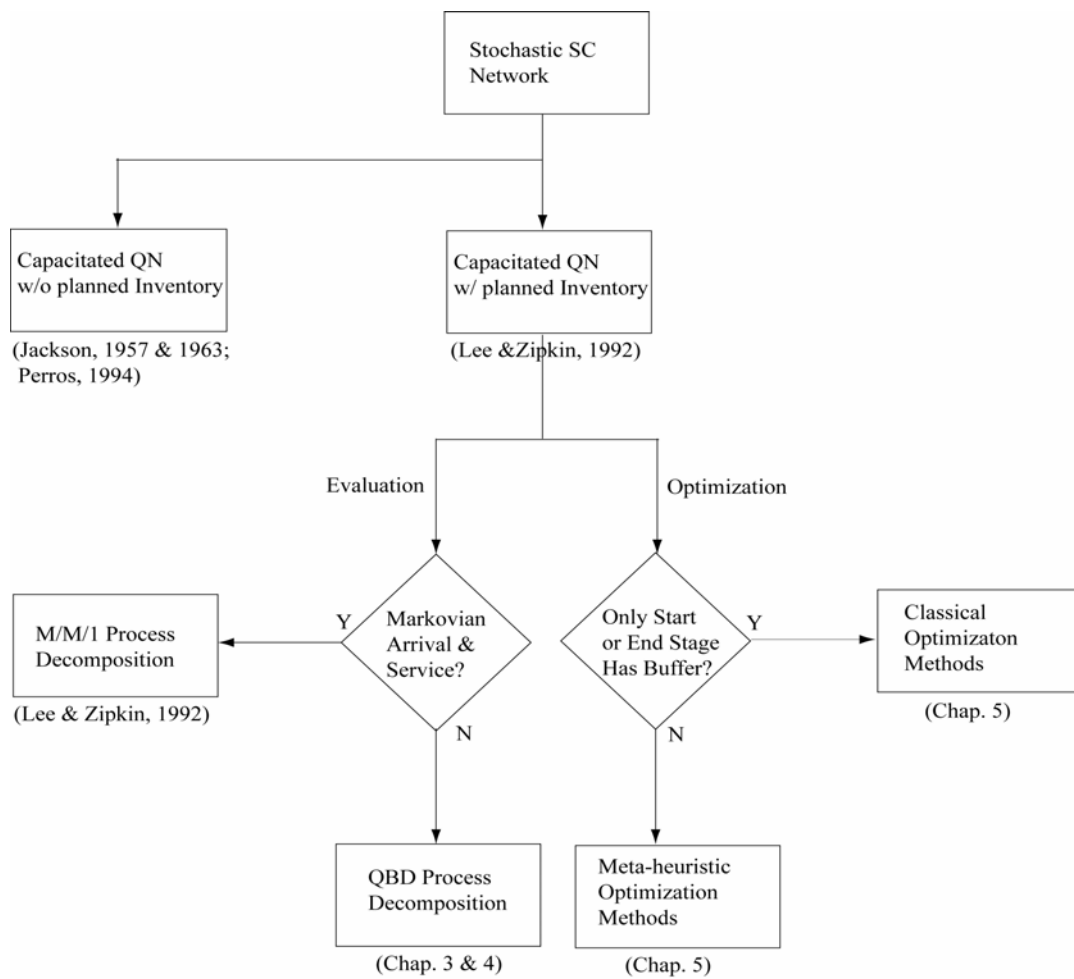


Fig. 1.2 An integrated framework for stochastic supply chain analysis.

1.4 Scope and limitation

This work uses daily operational data and probability distributions as input to the analytic framework. However, it belongs to strategic level of SCM. The major scope of this study is to find adequate steady state performance measures of an SC with tandem or any other tractable topology and manageable system dynamic. We assume the behavior of the studied SC is predictable such that it can be modeled as stochastic process with various probability distributions. Hence we rule out any abrupt behavior of an SC, such as unwarned unavailability caused by natural disaster, strike etc. As compared to traditional single stage inventory analysis, this work studies multi-echelon inventory control in SC domain. It used abstract level of modeling approach, neglecting unnecessary details inside each echelon. However, this limitation can be relaxed by other advanced modeling skills such as QBD process introduced in chapter 4. This study limits to continuous review base-stock inventory control policy. Unless otherwise stated, we assume service distributions are exponential at all stages except for distribution stage where a Hyper-exponential probability distribution is purposefully modeled. For the modeling of inbound logistics, exponential distributions are also assumed for each supplier's processing. We assume there is an infinite input buffer for each stage along the SC, and that each stage uses the base stock control policy to manage its finite output buffer. Also, unit transfer is assumed and the supply discipline is assumed to be first-come-first-served (FCFS).

1.5 Organization

This study is organized as follows. Chapter 2 reviews related work. Chapter 3 proposes our performance evaluation model. Chapter 4 discusses the advanced modeling approach for uncertainty. Chapter 5 shows the optimization process. In chapter 6 we draw our conclusions.

Chapter 2 Related works

2.1 Queueing network with planned inventory

Basic knowledge of probability models is necessary in solving stochastic problems, for which we refer to Cinlar (1975), or Taylor and Karlin (1994), or Kao (1997). From literature survey, we observed that all the evaluation models for solving planned inventory type problems used system dynamic of (1.1) and (1.2) in chapter 1. Svoronos and Zipkin (1991) first proposed the matrix-computation approach to solve multi-echelon distribution/inventory problems. The authors assumed unlimited capacity with stochastic lead-times. In particular, the lead-times are unaffected by demand. The major result of Svoronos and Zipkin (1991) is that the transit-time variances play an important role in system performance. Later L & Z used similar approach to discuss the model of a tandem queue with planned inventories. The model therein assumed finite capacity. It used the model of Svoronos and Zipkin (1991) as an approximation. In order to make the approximation reasonable, it set the parameters of the lead-time to correspond with the average lead-time in a queueing system. Hence the lead-times depend on the demand. The simulation results showed the accuracy of the approximation model. More research, which assumes that only the output buffer at the final stage is positive while the others are zeroes and most involving multiple final products can be found in the literature as reviewed by L & Z therein.

Based on L & Z, several studies developed approximation models of tandem supply systems. Using another recursive method, Zipkin (1995) calculated the same tandem congested problem through the convolution of $M/M/1$ queueing systems. They concluded an important concept, namely that a tandem queue with feedback built-in can be treated as capacity loss. Herein we relate the concept of capacity loss with adjusted traffic intensity as shown in chapter 3. Duri et al. (2000) extended the processing network of L & Z to

allow phase type distribution and they used convolution theory of two phase-type distributions to get the sojourn time at any complex stage and obtain similar accuracy of performance. They then used simple enumerative search to find the optimal buffer allocation assuming searching bounds of buffers were known. Actually we find their approach is very suitable for modeling complex system dynamic inside each echelon as shown in fig. 1.1. For example, the inbound logistics compose of a serial processing following by a transportation process. This may be modeled as a convolution of two phase-type models.

Boyacy and Gallego (2001) used constant lead-time assumption at each supply stage with Poisson arrival. Using SCV method, Liu et al. (2004) first proposed an efficient evaluation model (which we called SCV method herein) with reported percentage error of deviation from simulation being 22.712% under worst case for a congested tandem supply system with $J \leq 4$ (J refers to stage number). Gupta and Selvaraju (2004) investigated the possibility of applying QBD process on SC modeling. Though the computation difficulties increase in stage number, the decomposition approach of applying QBD on individual queueing system is tractable as reported herein. The flexibility of using QBD modeling approach to 'capture' complexity of inner echelon (stage) processing can be found, for example, in Neuts (1994, pp 274-286), where arriving customer order is served by multiple parallel machines. The random unavailabilities of machines are attended by multiple repairmen. The above review completes our survey of tandem supply systems. Except for Boyacy and Gallego (2001), which assume constant supply lead-time, all are modeled as queueing networks with planned inventories.

2.2 Multi-echelon inventory control theories

As for other multi-echelon inventory control theories, which we feel may contribute to our understanding in this area and therefore are listed below. Sherbrooke (1992) used

queueing theory to develop one for one (base stock) inventory control model for low demand repairable (or recoverable) aircraft parts for US Air Force, called METRIC in 1968 where METRIC stands for Multi-Echelon Technique for Recoverable Item Control. Several later stochastic inventory studies followed that approach. Besides, there are many other inventory control policies, which were similar to or based on the above control policies. Svoronos and Zipkin (1988) presented a two-moment approximation technique with reference to distribution/inventory system. Axsäter (1993a) presented approximate and exact evaluation models of batch-ordering policies for two-echelon inventory systems when retailers face unit demand. That investigation compared the proposed approach with the technique presented in Svoronos and Zipkin (1988). Axsäter (1993b) summarized an overview of continuous review policies with reference to multi-echelon inventory models. He showed that METRIC underestimates the performance of a system. Axsäter (2000a) developed an exact model of base stock policy to evaluate the system performance. The above investigations addressed only situations that involved identical retailers. Axsäter (2000b) further proposed an exact analysis of inventory policies in a two-echelon distribution/inventory system when retailers face customers with different compound Poisson demand (non-identical retailers). We found almost all the multi-echelon inventory control models reviewed above assumed unlimited upstream supply, and therefore supply lead-times are assumed constant. Regarding this, the study of congested system seems to be more practical since practically capacity of supply is seldom unlimited. Further, no matter it is related to congested analysis or not, almost no literature surveyed above put other unexpected settings such as machine unavailability, demand fluctuation etc. into its modeling logic. The exception is Liu et al. (2004), where they investigated the impact of processing variation on supply system performance. In our study, we stress the issues of uncertainties and put them into the modeling logic. Concerning this, we refer to other models related to uncertainties.

2.3 Models related to uncertainties

Emre et al. (2002) used simulation and regression to set up rules-of-thumb to decide adequate buffer capacity, which guaranteed high production efficiency of a tandem production system with unreliable machines. Abboud (2001) used the discrete-time Markov model to study the machine breakdown issue of a one-stage production/inventory model. Mohebbi (2003), Kalpakam and Sapna (1997), Mahmut and Perry (1995) used respective Markov models to formulate supply unavailability as two ‘on’ and ‘off’ states, and to study the embedded stochastic process to derive performance measures of interest. For robust and practical reasons, stochastic model should also take demand uncertainties into concern. As criticized by Thomas and Griffin (1996), market demand may not always follow stationary Poisson process. However, the modelers often overlooked these factors. As an example, Tee and Rossetti (2002) used Axsäter (2000b) as a test bed and run simulation to assert that the system behavior under stationary assumption may not behave as expected when the assumption is violated. Under our survey, another assumption usually adopted in stochastic modeling is “single server”, which is usually seen in a congested system analysis. The assumption does not seem to be practical. As is well known, manufacturers often adopt manufacturing cell, which gathers together similar or identical machines at one place to alleviate parts waiting time. The over-simplified assumption and neglect in the impact of uncertainties in all the stochastic models may make them far away from realistic application. MMPP is widely used as telecommunication traffic modeling (Trivedi 2002). Ching (2001) investigated the use of MMPP on several case studies of supply system with machine breakdown. However, he used finite continuous time Markov chain (CTMC) model and adopted “single server” assumption. In chapter 4, we address the issues of uncertainty in supply system. Specifically, we used MMPP to implement our uncertainty modeling for a supply system

concerning parallel (multi-server) processing.

2.4 An integrated framework for stochastic SC analysis

As we can see there were abundant and mature works in the literature related to independent production/inventory or distribution/inventory models as surveyed above. However, very little literature can be found on integrated SC model under stochastic setting. To solve problems like fig. 1.1, any of the above literature does not seem to be able to work if they work alone. As Houtum et al. (1996) pointed out that an integrated model for analyzing a multi-stage, multi-product SC problem, which is theoretically sound and numerically tractable would be recognized as a breakthrough in SCM study. Nevertheless, we do find endeavor in this direction. Research on such models can be found at Cohen and Lee (1988), Pyke and Cohen (1993 & 1994) etc. However they neglected the mutual relationship between the different subsystems. They didn't consider factors of uncertainties from upstream stages such as material unavailability, which influences the behavior of the downstream stage. Raghavan and Viswanadham (2001) is another example for problem with infinite buffer setting. Gurgur (2002) used the decomposition method to separate the whole SC into several two-node subsystems to facilitate the analysis and then used the iterative approach to integrate all the subsystems to obtain the final system-wide performance measures. In her study, transfer is assumed in batches and (r, q) control policies were used. Since under her design there is only one single limited buffer between any two nodes, blocking effect has to be tackled and thus is incorporated into the solution algorithm.

As for queueing network with planned inventory setting, Dong & Chen (2005) proposed an analytic framework for performance modeling and analysis of integrated supply chains. They used queueing theory to model capacitated supply networks, which composed of supplier, manufacturing, assembly, and distribution modules. The inventory

control they used belongs to (r, q) type. The authors employed the classical derivative method as the optimization avenue by differentiating performance measure of interest with respect to decision parameters. However, their approach focuses on mathematic derivation and therefore lacks investigation on possible insights in managerial applications. Also, it's very difficult for industry practitioners to understand the difficult logic of their mathematic model. Herein, we propose the solution framework, which is computation efficient with simple operation logic based on (1.1) and (1.2) of chapter 1. The method of L & Z originated from phase type distribution. Zipkin (1988) first used phase type distribution in inventory study and the main results achieved were that the marginal distribution of lead-time demand has a discrete phase-type distribution with the same number of phases as the lead-time distribution. Under L & Z, the matrix parameters have to be decided first. We propose the QBD processes to model the system logic locally and use MGM to obtain these matrix parameters for global linkage. For using MGM to solving QBD, we refer to Latouche (1999) and Neuts (1994). Actually, QBD process is very flexible in modeling complex queueing processes. Neuts (1979 and 1994) used QBD process to model the traditional machine-interference problem with external customer demand input. The machine interference (repair) problem originates from the fact when a failing machine needs repair and all the repairmen are busy attending the other down machines it has to wait until one repairman is available. With the additional customer demand as input, QBD can handle well. When we delve into the above literature, we recognized that the formulation of an evaluation model for an independent system such as a production/inventory or a distribution/inventory system needs great efforts to achieve satisfactory results, to say nothing of trying to formulate an integrated evaluation model for a whole SC under stochastic setting.

Our own study experience herein also showed that the direct construction of CTMC models becomes cumbersome and error-prone and sometimes even intractable (chapter 4)

if states become enormous. Recent research points to another high-level modeling approach. Generalized Stochastic Petri Net (GSPN) provides a very useful high-level interface for the automatic generation of underlying CTMC. However, since it's beyond the scope of this research. We leave it for the time being and recapture it in chapter 6.

2.5 Optimization of a stochastic SC

Except for the optimization method mentioned earlier, there are other approaches, which may be suitable for solving large-scaled or special purposed SC problems. Meta-heuristic methods are such examples. Among them, genetic (evolution) algorithm, simulated annealing, Tabu search, were largely applied in areas of engineering and science areas as shown in Pham and Karaboga (2000). Michalewicz (1999) introduced meta-heuristic methods, and especially genetic algorithm, in detail. Yokoyama (2002) compared the computational efficiency of a random search and genetic algorithm in an integrated optimization of inventory and the distribution system. In Yokoyama (2002), the SC was first mathematically modeled and, then, optimal decision parameters were sought using random search and genetic algorithm. The computational efficiency was evaluated by comparing the number of computational iterations. Spinellis et al. (2000) used simulated annealing to optimize finite-buffered production lines by using a so-called “expansion method” as the evaluation method. In their study, they state that buffer allocation problem (BAP) is a difficult NP-hard combinatorial optimization problem. It is even more difficult that the performance measures are difficult to be expressed as closed-form based on the decision variable. Exact approaches are appropriate for solving small problem instances or for problems with special structures. For complex problem such as an SC, it seems to imply that heuristic approaches are more appropriate for such problems. Heuristic approaches can be referred to methods such as classical non-linear programming search methods, or meta-heuristic methods. In this study, we think heuristic

approaches may also include any naïve method in simplifying the search method in an efficient and accurate way. See for example Boyaci & Gallego, (2001) as reviewed later. For BAP of finite-buffered systems, several other works have been reported. Ajay & Smith (1997) proposed expansion method to model blocked QN and used the Powell's method, which is based on classical conjugate gradient search to solve the BAP for several test problems of series, split, and merging types. The results were verified through simulation runs. For a finite-buffered production line, Liu and Lin (1994) proposed an approximated evaluation method for an unbalanced production line and even for long production lines. A dynamic programming method is proposed to find the minimum buffer units to provide maximum system throughput. Gurgur (2002) used design of experiment (DOE) to locate the optimal buffer setting.

As for BAP for MTS systems some works adopting base-stock policies can be found. Graves & Willems (2003) developed two models tackling problems of safety stock placement, which relate to service level requirements and assume normal supply lead-time. They employed several industrial cases to illustrate the applicability of the models. Further, the authors formulated a nonlinear mixed-integer optimization program to decide issues of SC configuration such as option selection and the service time decisions. Boyaci & Gallego (2001) used enumerative search to solve the BAP under pre-specified service constraint. Also they compared different heuristic methods with enumerative method. Two-stage heuristic (TSH) restricts that only two stages hold inventories, the last and some other stage. The Majorization heuristic (MH) uses greedy procedure that initially places all the stock at stage J and then moves maximum possible stock to stage $J - 1$ while retaining feasibility, and it repeats the procedure for $J - 1$, $J - 2$, etc. The mixed heuristic (MXH) modifies the sequential push mechanism of MH. Instead of moving maximum possible stock from stage j directly to $j - 1$, it compares the potential cost savings of moving maximum stock to all possible upstream stage $1, \dots, j - 1$ and choose the best location j^* ,

and repeats the procedure for stage $1, \dots, j^* - 1$. Liu et al. (2004) tried to minimize the holding costs of planned inventories of a congested tandem supply system under service constraint. They proposed a so-called “relaxation-recursive approach” to implement their optimization process and apply this optimizer in managerial studies. In addition, they investigated the impact on system performance by factors of output buffer, workload sequence, and service-time variation. Axsäter (2000b) proposed truncation measures in search process, which is to systematically disregard events with very low probabilities.

In addition to the above analytically tractable models, which may depend largely on pre-specified assumptions to work, many of the literature used simulation as the performance evaluation tool accompanied with some optima-searching methods for large-scaled SC problems. For example, Alberto I. et al. (2002) used simulation as the performance evaluation tool and employed evolution strategy (ES) as the optimization method to tackle a goal-programming problem for a European recycling plant project. The outputs of simulation runs serve as fitness functions of optimization process. The outputs of ES serve as input parameters of simulation process. The optima were found after iterative simulation runs. However, the major flaw of simulation is time-consuming. In this study, we developed several optima-searching algorithms. Some of them follow the spirit of Alberto I. et al. (2002). However, the evaluation functions are derived through analytically tractable forms instead of simulation.

Chapter 3 An integrated evaluation model

3.1 Introduction

Our objective in this chapter is two-fold. The first is to develop a flexible modeling approach, which can ‘capture’ realistic activities inside each stage, such as parallel servers, machine unavailability, etc. We used QBD process to achieve this goal. The second is to extend the applicability of L & Z to include into our model not only a tandem-processing network but also other SC subsystems. Unlike L & Z, which assumed single server and exponential distribution at each processing stage, our model relaxes these assumptions and hence allows for more modeling flexibility. Our model is also a variant of classical tandem supply network. Different from previous related works (see literature review below), our work links production/inventory subsystem and distribution/inventory subsystem. Transportation process is considered as well. All three subsystems have limited capacities (actually we use single-server settings in the main part of this study). Through QBD transformation, the original complex topology of an integrated stochastic SC becomes tandem-like and hence tractable. We also used QBD process to model machine unavailability, which makes our model more real than other integrated stochastic SC models such as Cohen and Lee (1988).

In this study we assumed that there is an infinite input buffer at each stage along the SC, and that each stage uses the base stock control policy. A policy of this kind demands that each stage starts operation at its own target inventory level at its output buffer. Under such scheme, the output buffer at each stage is set to be finite, while the input buffer at each stage doesn't have to be set so. However the infinite assumption at the input buffer of each stage releases the difficult analysis of possible blocking effect when units at the upstream stage cannot find any vacancy at the input buffer of downstream stage. Also, unit transfer is assumed and the supply discipline is assumed to be first-come-first-served (FCFS). For

practical reason, there is usually a natural quantity unit for both demand and supply (e.g. truckload or 1 000 tons/ unit load), and in terms of that unit it makes sense to set order quantity to be equal to unity. First we formulated the respective stages as either $M/M/1$ or phase-type queueing model. For the latter type, we then used the QBD model of the Markov process to derive respective sojourn times. Finally, the method of L & Z was applied and then the system-wide performance measures were computed approximately with respect to the base stock levels at all sites. A simulation model was also developed to facilitate the verification study of the accuracy of the proposed approach.

3.2 Matrix analytic approach

The inventory control scheme of our proposed approach is the base stock policy. In practical production/inventory control policies, in contrast to centralized (and push-type) control scheme like MRP, there are other local (and pull-type) control policies like (r, q) , KANBAN and their variants except for base-stock policy. Though we used base-stock policy in this study, the extension of the existing model to other control policies is possible (this will be investigated in section 3.5). Below we illustrate why we select this policy instead of other pull-type control schemes.

The base-stock policy makes sense when economies of scale in the SC are negligible relative to other factors. For example, when each individual unit is very valuable, and hence holding and backorder costs dominate any fixed order (set-up) costs. Likewise, for a slow-moving product (one with a low demand rate where Poisson distribution is adequate to model the arrival process), the economics of the system dynamics clearly rule out batch size (Zipkin, 2000). When the above conditions no longer exist, for example, economies of scale do matter; other control schemes such as (r, q) policy may be more adequate than base-stock policy. In this study, we assumed processing conditions are like those mentioned above so as to use base-stock policy accordingly. On the other hand, since

KANBAN is more restrictive when possible blocking may occur due to no immediately available KANBAN cards at hand when demand arrives, we select base-stock policy as our major control scheme to quickly verify the applicability of the proposed model in the first place. Also, base-stock policy is not uncommon in practical production/inventory control situation. Finally, it's known base-stock policy can be treated as the building block of (r, q) policy and therefore we begin our study from the base-stock policy.

Next we discuss how the base stock control policy works. This policy is also called $(S-1, S)$ policy. Where S represents base stocking level. This policy means that whenever demand reaches one unit, the inventory is immediately replenished. Under our proposed model, each stage along the SC has its own input queue (N_j) and output buffer (I_j) physically or imaginarily, where semi-finished or finished products are kept. Assume infinite N_j and finite I_j . Aggregate customer demands at the retailers trigger the delivery from the distribution center (DC). This demand information propagates to the production facility initiating a production order at each stage. For a specific production, transportation or distribution stage j , a material flow comes from the output buffer of the immediate upper-stage $j-1$. If the inventory at the buffer is available, one item is immediately deducted from the output buffer of $j-1$ and sent to the input queue of j . If the inventory at the buffer is not available, one item is backordered and recorded at $j-1$. When there is one part/product finished at stage j and there is recorded backorder, then the item will be sent immediately to the input queue of the next stage. Otherwise it will just stay at that stage as a base stock item. Under base-stock control, the stage adopting MTS policy will maintain its own stock level and reduce customer-waiting times downstream as compared to an MTO policy. Next, the models developed by Svoronos and Zipkin (1991) and L & Z are briefly discussed. In subsection 3.2.2, our proposed approach is presented.

3.2.1 The approximation model

Consider stage j and its immediate predecessor stage i . Then, Svoronos and Zipkin (1991) defined the following: $L_j = D_i + T_j$ and assume T_j and L_j had continuous phase-type distributions (CPH) as follows: $T_j \sim CPH(\alpha_j, A_j)$ and

$$L_j \sim CPH(\psi_j, G_j^*) \quad (3.1)$$

Let I denote an identity matrix and $\mathbf{1}$ a column vector of ones whose dimension is chosen to fit the content of the context. Then, they indicated that K_j has the same distribution as the lead-time demand. This property combined with Neuts (1981) theorem 2.2.8 implies that K_j has a discrete phase type distribution (DPH): $K_j \sim DPH(\pi_j, P_j)$ where

$$P_j = \lambda(\lambda I - G_j^*)^{-1} \quad (3.2)$$

$$\pi_j = \psi_j \cdot P_j. \quad (3.3)$$

Since $B_i = [K_i - S_i]^+$, where $[x]^+ = \max\{x, 0\}$ is a shifted phase-type distribution, it follows that $B_i \sim DPH(\pi_i P_i^{S_i}, P_i)$ (Neuts 1981, p. 47). According to Svoronos and Zipkin (1991), B_i has the same distribution as the waiting-time demand. Again this property combined with Neuts (1994) theorem 2.2.8 implies that $D_i \sim CPH(\psi_i P_i^{S_i}, G_i^*)$. From the definition of L_j , (3.1) is the convolution of two phase-type distributions: D_i and T_j . According to Neuts (1981) theorem 2.2.2, since $L_j = D_i * T_j \sim CPH(\psi_j, G_j^*)$, where $*$ represents convolution operation, then

$$\psi_j = [\psi_i P_i^{S_i}, (1 - \psi_i P_i^{S_i} \mathbf{1}) \alpha_j], \quad (3.4)$$

$$G_j^* = \begin{bmatrix} G_i^* & -G_i^* \mathbf{1} \alpha_j \\ 0 & A_j \end{bmatrix}. \quad (3.5)$$

As L & Z assumed each processing stage to be exponential with one single server, then

(3.5) can be expressed as (3.6) (see the following) after some recursive algebraic operations starting from stage 1:

$$\mathbf{G}_j^* = \begin{bmatrix} -v_1 & v_1 & & & & \\ & -v_2 & v_2 & 0 & & \\ & & \ddots & \ddots & & \\ & & & & -v_{j-1} & v_{j-1} \\ & 0 & & & & -v_j \end{bmatrix}, \quad (3.6)$$

where $v_k, k \leq j$ represents the inverse of the sojourn time of customer order at stage k . In our approximation approach, we relax the assumptions of exponential and single server. The inverse of sojourn time: v_k is obtained through QBD modeling. Under this approach, the processing activity at each stage can be modeled as complex as possible theoretically. This approach largely enhances the flexibility of the model.

Since there is no waiting time before the first stage, the distribution of L_j is the same as T_j , which is already known. Starting at $\boldsymbol{\psi}_j = [1]$, L & Z recursively solved (3.4) by using (3.2) and (3.6) and let $\alpha_j = 1$. From the property of DPH, they finally derived

$$\Pr\{K_j > S_j\} = \boldsymbol{\pi}_j \mathbf{P}_j^{S_j} \mathbf{1},$$

and

$$E[B_j] = \boldsymbol{\pi}_j \mathbf{P}_j^{S_j} (\mathbf{I} - \mathbf{P}_j)^{-1} \mathbf{1}, \quad (3.7)$$

where $\boldsymbol{\pi}_j$ is obtained from (3.3). Alternatively we find it's simpler to derive (3.7) as follows

$$\begin{aligned} E[B_j] &= \sum_{K_j > S_j} (K_j - S_j) \Pr\{K_j\} \\ &= \sum_{y_j \geq S_j} \Pr\{K_j > y_j\} \\ &= \sum_{y_j \geq S_j} \boldsymbol{\pi}_j \mathbf{P}_j^{y_j} \mathbf{1} \\ &= \boldsymbol{\pi}_j \mathbf{P}_j^{S_j} (\mathbf{I} - \mathbf{P}_j)^{-1} \mathbf{1} \end{aligned}$$

Here we use the tail probability to derive the second equality. Since $S_j = I_j + K_j - B_j$, where I_j represents on hand inventory at stage j , L & Z gave

$$\begin{aligned} E[I_j] &= S_j - E[K_j] + E[B_j] \\ &= S_j - \boldsymbol{\pi}_j(\mathbf{I} - \mathbf{P}_j)^{-1}\mathbf{1} + E[B_j] \end{aligned} \quad (3.8)$$

Notice that the first moment of DPH was used to derive the last equality of (3.8). For $j < J$, this quantity together with $E[N_{j+1}]$, gives the total intermediate inventory between stages j and $j + 1$. When all the S_j equal to zero, the initial probability vector of the Markov chain is $(1, 0, \dots, 0)$, so that the sojourn time in the queue, if it's a pure tandem one involving no feedback or breakdown issues, is the sum of independent J random variables with mean $\frac{1}{v_j}$, where $\frac{1}{v_j}$ is the respective sojourn time at stage j . Consequently the approximation

can be verified to be exact. However, in our test model, which we will discuss shortly, the queue involves feedback and breakdown. Under this situation, the respective sojourn time, except for the first stage in the tandem queue is still that of a $M/M/1$ queue. However we have to modify the sojourn time at the first stage to improve the accuracy of the approximation model as discussed in section 3.4. For now, we will only focus on the build-up of our approximation model as described below.

3.2.2 The proposed approach

Now we discuss how to use the QBD process combined with the approach of L & Z to derive the performance measures of more complex SC. First we discuss how to break the original queueing problem into many smaller queues along the chain. Then we use the matrix computation approach developed by L & Z and plug in all the decomposed sub-queues sojourn time information as the matrix parameter to derive the final performance measures that are of interest to us.

Our sub-queues include two types: the $M/M/1$ queue and the phase type queue. For the $M/M/1$ queue, the derivation of sojourn time is well known by simply applying the well known Little's formula. For the phase type queue, our model demands that each job arriving at stage j may have to go through several physical processing phases before it finishes the processing work and releases the occupied resource to the next arriving job waiting in the queue. Under this stochastic process, the infinitesimal generator matrix will have a tri-diagonal block form. Markov chain with this form is a QBD process. Applying the theory of QBD, we can derive the expected sojourn time at this processing stage. As for the distribution subsystem, we can also treat it as an $M/PH/1$ queue and apply the above QBD process derivation procedure. Alternatively we can accumulate all the retailers as a single stocking site and treat it as an $M/M/1$ queue, which will later be shown to be equal to the $M/PH/1$ queue under some specific conditions. And we then calculate each individual retailer separately and finally we obtain aggregate performance measures for retailer site. In the following, we use the steady-state probability derivation procedure as illustrated in Feldman (1995, see Appendix A.1) to derive the sojourn times in an unreliable production stage and a distribution stage respectively.

First, we derive the sojourn time for an unreliable processing stage. Assume 0 and 1 phases represent the breakdown and operating states respectively. And, assume all stochastic processes are Markovian with parameters $\lambda, \mu, \zeta, \gamma$, representing mean arrival, processing, and the up and down rates respectively. We can then formulate the phase type generator as

$$\mathbf{G} = \left[\begin{array}{cc|c} -\gamma & \gamma & 0 \\ \zeta & -(\mu + \zeta) & \mu \\ \hline 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{c|c} \mathbf{G}_* & \mathbf{G}_d \\ \hline \mathbf{0} & 0 \end{array} \right].$$

(Note that the bold character form represents vector or matrix.) Assume the initial probability in the phase stage as $\alpha_* = (0, 1)$, apply (A.1.5), and after some matrix

algebraic operations, we get

$$\mathbf{R} = \begin{bmatrix} \frac{\lambda}{\lambda + \gamma} (1 + \frac{\zeta}{\mu}) & \frac{\lambda}{\mu} \\ \frac{\lambda}{\lambda + \gamma} \cdot \frac{\zeta}{\mu} & \frac{\lambda}{\mu} \end{bmatrix},$$

which is consistent with Buzacott and Shanthikumar (1993, p122). Applying the expectation formula ($L = \sum_{n=1}^{\infty} n \cdot p^n$) and (A.1.4), we can easily obtain the expected number of orders in the system

$$L = (1 - \rho) \alpha_* \mathbf{R} (\mathbf{I} - \mathbf{R})^{-2} \mathbf{1} \quad (3.9)$$

where the traffic intensity rate is

$$\rho = \lambda E[T] = -\lambda \alpha_* \mathbf{G}_*^{-1} \mathbf{1}. \quad (3.10)$$

The last equality of (3.10) is from the CPH distribution. Then the sojourn time in the processing stage can be obtained by applying the Little's formula $W_s = \frac{L}{\lambda}$.

Since every distribution can be approximated as closely as desired by phase type distribution (Svoronos and Zipkin, 1991), it seems that we can formulate any stage in the SC as a QBD process in a very flexible way. For now, we will now apply the same approach to a distribution subsystem and show that the end result is the same as treating all the retailers as a single stocking unit under some conditions. Basically the random process of a distribution system can be modeled as a Hyper-exponential process. Recall a Hyper-exponential distribution as shown in fig. 3.1. We can treat the start node as the input queue to each retailer route. α_i is the probability of which route the transportation will take.

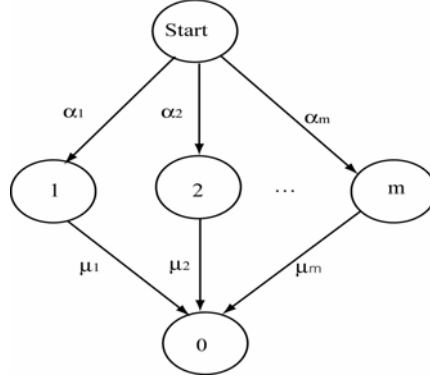


Fig. 3.1 A transition diagram of a Hyper-exponential distribution.

In the long run, under normal conditions, α_i can be approximated as $\frac{\lambda_i}{\sum_i \lambda_i}$, where

λ_i represents the mean order rate for retailer i , and the denominator is just the average aggregate demand rate. Node 0 can be thought of as the location of the collective single stock-place. For ease of derivation, assume that the expected delivery rates for all routes are identical, that is $\mu_1 = \mu_2 = \dots = \mu_m = \mu$. Also assume that there are m retailers, and that all customer demands are identical, that is $\lambda_1 = \lambda_2 = \dots = \lambda_m = \lambda$ and thus

$\alpha_1 = \alpha_2 = \dots = \alpha_m = \frac{1}{m}$. Then we have the following phase-type representation:

$$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_m) = \boldsymbol{\alpha}_*, \quad (3.11)$$

$$\mathbf{G} = \left[\begin{array}{ccc|c} -\mu & & & \mu \\ & \ddots & & \vdots \\ & & -\mu & \mu \\ \hline \mathbf{0} & & & 0 \end{array} \right] = \left[\begin{array}{c|c} \mathbf{G}_* & \mathbf{G}_\Delta \\ \hline \mathbf{0} & 0 \end{array} \right] \quad (3.12)$$

Applying (A.1.5), we get

$$\mathbf{R} = \left[\begin{array}{cc} \frac{(\lambda + m\mu)\lambda}{m\mu(\lambda + \mu)} & \frac{\lambda^2}{m\mu(\lambda + \mu)} \\ & \ddots \\ \frac{\lambda^2}{m\mu(\lambda + \mu)} & \frac{(\lambda + m\mu)\lambda}{m\mu(\lambda + \mu)} \end{array} \right].$$

Applying (3.10), after some algebraic operations, we get $\rho = \frac{\lambda}{\mu}$ and using (3.9), we

derive

$$\begin{aligned}
 L &= (1 - \rho)\alpha_*\mathbf{R}(\mathbf{I} - \mathbf{R})^{-2}\mathbf{1} \\
 &= (1 - \frac{\lambda}{\mu})[\frac{1}{m}, \dots, \frac{1}{m}]\mathbf{R}(\mathbf{I} - \mathbf{R})^{-2}\mathbf{1} \\
 &= \frac{\lambda}{\mu - \lambda}.
 \end{aligned} \tag{3.13}$$

We used the symbolic math toolbox of MATLAB to derive the last equality of (3.13) by plugging in any number of m greater than or equal to 1, otherwise it becomes too laborious to derive manually. Actually we found that by using the Pollaczek-Khintchine formula, the result is the same as the above. The square of coefficient of variation of the

service time of the above $M/H_m/1$ queue is $C_s^2 = \frac{Var[T]}{E^2[T]} = \frac{2\alpha_*\mathbf{G}_*^{-2}\mathbf{1} - (-\alpha_*\mathbf{G}_*^{-1}\mathbf{1})^2}{(-\alpha_*\mathbf{G}_*^{-1}\mathbf{1})^2}$, which is

equal to unity by plugging in (3.11) and (3.12) and after some algebraic operations. Notice here that we use the first and the second moments of CPH. So

$$W_q = \frac{1}{2}(1 + C_s^2)\tilde{W}_q = \frac{\lambda}{\mu(\mu - \lambda)}$$

where \tilde{W}_q is the waiting time in queue of an $M/M/1$ queue with arrival rate λ and service rate μ .

We have just shown that if all the initial probability and service rate at each phase of an $M/H_m/1$ queue are identical, then its performance is the same as an $M/M/1$ queue. Though the above result can be easily identified on the probability density function of Hyper-exponential distribution, our purpose here is to illustrate how QBD can handle such distribution structure usually seen in an SC study. Here we use a special case to illustrate the derivation process. However the application is not limited to such special distribution form as assumed above. For the sake of brevity, we omit the details here. However, we show the general case in chapter 5. We have indicated how to derive all the sojourn times inside each stage for a realistic SC. Now we can use L & Z to derive the performance

measures of a tandem queue. We test the accuracy of our proposed model by employing it on a tentative multi-echelon production, transportation and distribution system as described below.

3.3 Implementation

3.3.1 A test problem

A multi-echelon production, transportation and distribution model as shown in fig. 3.2 is employed as a test bed for our method. To keep the study manageable, we restrict our attention to a very basic model. The production facility (PF) produces finished goods to downstream retailers. The retailers face a stationary Poisson demand process with mean inter-arrival time of $1/\lambda$. Machining process is as introduced in section 3.2. Successfully finished goods will leave the machine and go to the next stage for final inspection before shipping to a remote DC. After inspection, any imperfect product has to go back to the processing stage for reworking. Assume that the feedback rate is constant with probability δ . For the sake of simplicity we assume that the second (inspection) stage will never fail. Products passing inspection will wait at the shipping area, ready for transportation to DC. Upon arrival at the DC, the product will immediately be transported to the assigned retailer whenever a transporter is available. Again, for the sake of simplicity, we restrict all transporting vehicles between any two sites to one. Assume that all the transportation times are stochastic. Applying the method as described in section 3.2, we formulate this problem as a tandem queue with 5 independent stages. The first stage is the production stage with the unreliable machine being formulated as two “on” and “off” phases. The second stage is the inspection stage. The transportation from PF to DC and the DC itself are formulated as respective $M/M/1$ queueing systems. Finally, the distribution stage is formulated as a phase type, even though it is easier to accumulate all the retailers as one single stocking site, and

treat it as an $M/M/1$ alike, as shown in section 3.2.

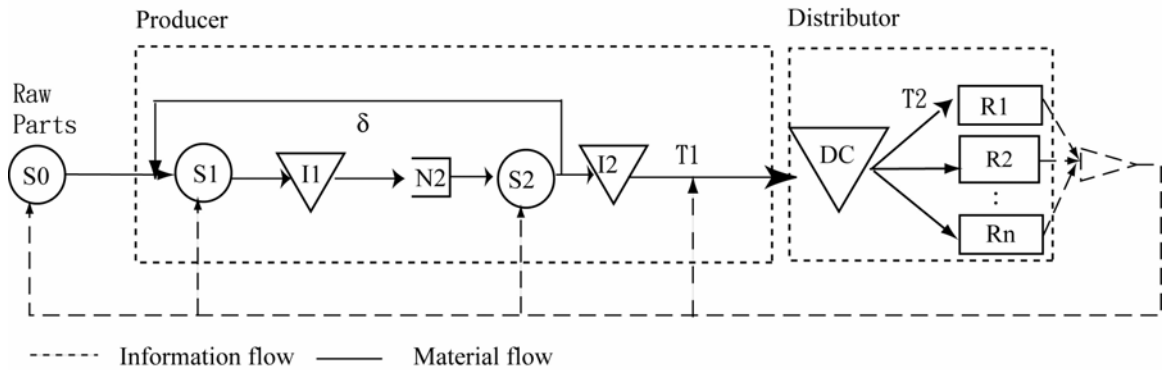


Fig. 3.2 A multi-echelon SC with feedback consideration.

Please note that we omitted the other N_j and I_j except those of PF in fig. 3.2. Specifically, N_3 , the input queue of the transit from PF to DC; I_3 , the output buffer of the transit from PF to DC; N_4 , the input queue of DC; I_4 , the output buffer of DC, N_5 , the input queue of the transit from DC to retailer; I_5 , the output buffer of the transit from PF to DC, which is set to the accumulative retailer inventory level in this design. Further, assume there is an infinite supply at the first stage.

I_3 is always zero, assuming the MTO policy is adopted by this service. At DC, it's reasonable to adopt the MTS policy to lessen the customer order waiting time. Assume that the DC processes its inventory with high efficiency at near zero operation time. This means that each arriving good will be put into stock immediately if there is no backorder recorded. When there is a backorder, the arriving unit will be shipped to the waiting retailer. N_4 is always zero as well. If the customer order arrives, and the stock is out, a situation, which the MTO-type control is sure to encounter, unfilled orders are backlogged and will be satisfied when replenishing goods arrive on a FCFS basis.

3.3.2 Numerical results

Here reports our tests of the approximation of the model illustrated above, and compares its predictions to estimates derived from computer simulation, as illustrated in

Appendix A.2. Basically we follow the same test approach as reported in Zipkin (1995) with some modifications. The queueing system at PF is just like an open Jackson network. Thus the λ_i are all identical to $\lambda/(1 - \delta)$, where δ is the feedback rate. All ρ_i are equal to $\rho = \lambda/[u(1 - \delta)]$. To test the taxing condition on the performance of the approximation, we fix $\delta = 0.5$. ρ is determined by λ/u . Assume that the mean demand rate for each retailer is 0.25 and that there are four retailers. The combined demand rate is 1. We fix u to be either 2.5 or 4, and thus ρ is 0.8 or 0.5 respectively. Assume mean failure and repair rate to be 0.25 and 2.5 respectively. Assume that the average transportation time is 1/4. We adopt a similar simulation stopping criteria as reported in L & Z and Zipkin (1995). Each run simulates thirty replications of 10 000 time units. Assume there is a holding cost of 0.5 for working-in-process per unit and per unit time, a holding cost of 1 for the end retailer inventory per unit and per unit time, a backorder cost of 10 for unfilled retailer orders per unit and per unit time. Five key performance measures are measured, TC (the total incurred cost of operating the chain, which is equal to $0.5 \cdot \text{WIP} + E[I] + 10 \cdot E[B]$, see below), SL (average service level measured in no stock-out probability at the retailer site), WIP (the total intermediate inventory, which is defined as all the working-in-process, inventory level at DC, and all the queues of transit, $I_1 + N_2 + I_2 + N_3 + I_4 + N_5$, in this case), $E[I]$ (average retailer inventory, which is omitted for space consideration), $E[B]$ (average retailer backorder). Note that in calculating WIP, I_3 and N_4 are always zeroes, as described above. Tables 3.1 and 3.2 summarize the results. Note that the parameter setting of table 3.1 is the same as in Zipkin (1995).

Also notice that the ‘SL’ column is not listed in table 3.1 since they are all zeros. The column labeled S_j is the initial base stock level at the respective stages. The column labeled ‘Sim’ represents the simulation estimates; ‘App’ stands for the approximation, and ‘%Err’ is the absolute percentage error of the approximation compared to the simulation value,

which is defined as $|App - Sim| / Sim \times 100\%$. It is evident that the approximation is quite accurate for table 3.1 with all retailers adopting base stock policies with $S_5 = 0$. Table 3.2 shows the results when all retailers adopt base stock policies with $S_5 \neq 0$. Also, we adjusted the stock levels for all the other stages according to base stock levels of table 3.1. From table 3.2, we see that when $S_5 \neq 0$, the accuracy of the matrix approximation method is also satisfactory. From table 3.2 several useful observations can be made. For example, in the case of $S_5 \neq 0$ with $\rho = 0.5$, an increasing stock level at different stages, except at the last stage, seems to have the same effect of performance influence. The total cost and WIP levels increase and the service levels increase very limitedly while backorder levels decrease slightly. On the other hand, an increasing stock level at the last stage, i.e., retailer inventory level, does increase the service levels and decreases the backorder level, however it does so at the price of higher total cost, which is due to higher retailer inventory levels.

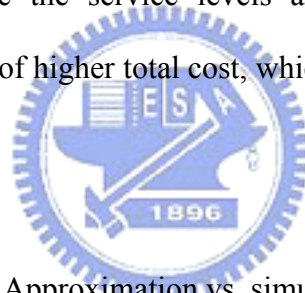


Table 3.1 Approximation vs. simulation ($S_5 = 0$)

ρ	S_1	S_2	S_3	S_4	TC			WIP			E[B]		
					Sim	App	%Err	Sim	App	%Err	Sim	App	%Err
0.5	0	0	0	0	30.605	30.177	1.40	1.684	1.668	0.95	2.949	2.93	0.64
0.5	1	1	0	1	13.237	13.019	1.65	2.825	2.8908	2.33	1.182	1.1574	2.08
0.5	3	1	0	1	11.388	10.930	4.02	4.359	4.597	5.46	0.921	0.863	6.30
0.5	1	3	0	1	9.029	9.011	0.20	4.322	4.414	2.13	0.687	0.680	1.02
0.5	1	1	0	3	8.494	8.425	0.81	4.28	4.358	1.82	0.635	0.625	1.57
0.5	1	1	0	5	7.345	7.420	1.02	6.08	6.167	1.43	0.43	0.434	0.93
0.8	0	0	0	0	118.03	125.01	5.91	4.625	4.668	0.93	11.825	12.268	3.75
0.8	1	1	0	1	93.182	97.455	4.59	4.821	4.900	1.64	9.077	9.500	4.66
0.8	3	1	0	1	84.620	83.820	0.95	5.436	5.507	1.31	8.19	8.107	1.01
0.8	1	3	0	1	75.606	81.049	7.20	5.126	5.243	2.28	7.304	7.843	7.38
0.8	1	1	0	3	76.335	80.931	6.02	5.154	5.232	1.51	7.376	7.832	6.18
0.8	1	1	0	5	65.185	66.975	2.75	5.828	5.807	0.36	6.227	6.407	2.89

Table 3.2 Approximation vs. simulation ($S_5 \neq 0$)

ρ	S_1	S_2	S_3	S_4	S_5	TC			SL			WIP			E[B]		
						Sim	App	%Err	Sim	App	%Err	Sim	App	%Err	Sim	App	%Err
0.5	4	4	0	4	4	9.309	9.512	2.18	0.919	0.994	8.16	10.636	11.079	4.17	0.031	0.029	6.45
0.5	12	4	0	4	4	13.167	13.497	2.51	0.922	0.994	7.81	18.396	19.074	3.69	0.028	0.027	3.57
0.5	4	12	0	4	4	13.248	13.484	1.78	0.923	0.994	7.69	18.596	19.068	2.54	0.026	0.026	0
0.5	4	4	0	12	4	13.252	13.482	1.74	0.923	0.994	7.69	18.607	19.067	2.47	0.026	0.026	0
0.5	4	4	0	4	12	16.969	17.198	1.35	0.999	1	0.10	10.624	11.079	4.28	0	0	N/A
0.5	4	4	0	4	20	24.964	25.194	0.92	1	1	0	10.624	11.079	4.28	0	0	N/A
0.8	4	4	0	4	4	29.772	29.923	0.51	0.649	0.702	8.17	8.125	8.331	2.53	2.347	2.335	0.50
0.8	12	4	0	4	4	23.469	20.958	10.70	0.742	0.836	12.67	13.379	14.532	8.62	1.416	1.075	24.09
0.8	4	12	0	4	4	19.066	19.118	0.27	0.816	0.882	8.09	13.947	14.115	1.21	0.930	0.889	4.44
0.8	4	4	0	12	4	22.175	19.119	13.78	0.813	0.882	8.49	13.744	14.114	2.69	1.026	0.889	13.38
0.8	4	4	0	4	12	23.176	21.999	5.08	0.866	0.885	2.19	8.125	8.331	2.53	0.976	0.888	9.05
0.8	4	4	0	4	20	25.047	23.902	4.57	0.944	0.956	1.27	8.125	8.331	2.53	0.407	0.333	18.08

To conclude, the approximation does seem to work well for all the retailers adopting either MTO or MTS operational strategies with one-for-one replenishment policies. When we incorporate all the stochastic features, including imperfect quality, machine breakdown, random transportation, and random distribution in the system, the degradation of the accuracy is only slight, and is often within the tolerance limits of industrial use. The feedback factor can be treated as capacity loss as concluded in Zipkin (1995).

3.4 Discussion and sensitivity analysis

For a tandem queue without feedback, every stage behaves just like an independent $M/M/1$ service system. The sojourn time is exact by applying Little's formula $W_s = \frac{1}{\mu - \lambda}$ in each stage, which is not influenced by base-stock setting at each stage. The matrix-algebraic solution of the performance evaluation is approximately correct as reported in L & Z. However, the sojourn time varies in the first stage when there is feedback. We compared our findings with the numerical results of Zipkin (1995), which are shown in Tables 3.3 and 3.4. Looking at Table 3.3, which is a two-stage system, it's apparent that the sojourn time (ST) at stage 1 increases when the stock level at stage 1 (S_1)

increases for both traffic intensity rates (0.5 and 0.8). Fig. 3.3 shows this tendency for $\rho = 0.5$. We can see that ST starts from 0.5, when $S1=0$, and then increases when $S1$ increases until it finally converges at near 0.7 when $S1$ is near 20. After modifying the sojourn time at stage 1, which is obtained by simulation, and applying it to the matrix approximation procedure, we get a closer match between approximation value and simulation value for both performance values of WIP and $E[B]$. Here Sim(1) is the simulation values adopted from Zipkin (1995) for comparison. Sim(2) represents the results from our own simulation model. It shows great agreement when compared with that of Zipkin (1995). For comparison, in Table 3.3 we show the absolute percentage error between App and Sim (1) as indicated in the %Err (1) column as defined in section 3.3. The %Err (2) is the absolute percentage error between App and Sim (2). The %Err (3) is the absolute error before adjusting the sojourn time at stage 1, which is reported by Zipkin (1995). It's clear that the sojourn time at stage 1 does influence the accuracy of the theoretical approximation value.

The WIP and $E[B]$ of stage 2 as a function of $S1$ for a two-stage PF with $\rho = 0.5$ are also shown in fig. 3.4 and 3.5. App (adj) means the performance by applying adjusted sojourn time to the matrix solution. App ('adj) is the performance by not plugging in adjusted sojourn time. We can observe minor differences between the approximation and the simulation results regarding the base-stock level at stage 1. Table 3.4 shows the comparison of simulation and approximation of a four-stage system. It seems that the accuracy does not improve as expected for a system composed of more stages. However, the accuracy does not degrade either for longer line. In conclusion we find that the impact of the above analysis on the accuracy of the matrix algebraic method is limited. In the worst case, the absolute error between Sim and App of WIP is only near 0.5. Therefore, there is no adjustment made in the computing code of the implementation section.

As for our tentative SC model of section 3.3 we also tested the case when there is only

feedback and no machine break down issue incorporated. Basically the difference between App and Sim is also small, when compared to the numeric results of section 3.3. In addition we investigated when there is only the influence of machine breakdown, and it behaved as expected when compared to the simulation results.

Table 3.3 A two-stage system

ρ	S1	ST	WIP						$E[B]$ (S2=0)					
			Sim (1)	Sim (2)	App	%Err (1)	%Err (2)	%Err (3)	Sim (1)	Sim (2)	App	%Err (1)	%Err (2)	%Err (3)
0.5	0	0.501	N/A	1.005	1	N/A	0.5	N/A	N/A	2.01	2	N/A	0.5	N/A
0.5	1	0.548	1.475	1.487	1.477	0.1	0.7	1.7	1.56	1.57	1.573	0.8	0.2	3.8
0.5	3	0.626	2.97	2.959	2.963	0.2	0.1	5.2	1.237	1.213	1.215	1.8	0.2	9.1
0.5	5	0.668	4.753	4.748	4.746	0.2	0	5.9	1.095	1.089	1.082	1.2	0.6	5.9
0.8	0	2	N/A	3.997	4	N/A	0.1	N/A	N/A	7.998	8	N/A	0	N/A
0.8	1	2.089	4.145	4.233	4.2	1.3	0.8	1.3	7.253	7.421	7.371	1.6	0.7	0.7
0.8	3	2.176	4.894	4.895	4.988	1.9	1.9	3.2	6.212	6.253	6.34	2.1	1.4	2.6
0.8	5	2.299	5.983	6.014	6.121	2.3	1.8	6	5.575	5.622	5.719	2.6	1.7	4.7

Table 3.4 A four-stage system

ρ	(S1, S2, S3)	S. T.	WIP						$E[B]$ (S4=0)					
			Sim (1)	Sim (2)	App	%Err (1)	%Err (2)	%Err (3)	Sim (1)	Sim (2)	App	%Err (1)	%Err (2)	%Err(3)
0.5	(0,0,0)	0.498	N/A	2.979	3	N/A	0.7	N/A	N/A	3.971	4	N/A	0.7	N/A
0.5	(1,1,1)	0.549	4.213	4.229	4.1446	1.6	2	0.6	2.308	2.333	2.2426	2.8	3.9	5.2
0.5	(3,1,1)	0.568	5.908	5.92	5.8492	1	1.2	0.8	2.036	2.057	1.9852	2.5	3.5	4.1
0.5	(1,3,1)	0.575	5.695	5.685	5.6131	1.4	1.3	0.4	1.841	1.832	1.763	4.2	3.8	6.6
0.5	(1,1,3)	0.591	5.497	5.478	5.328	3.1	2.7	0.9	1.7	1.658	1.51	11.2	8.9	15
0.5	(1,1,5)	0.621	7.104	7.093	6.9596	2.0	1.9	0.7	1.371	1.327	1.2016	12.4	9.5	15.6
0.8	(0,0,0)	2.028	N/A	12.066	12	N/A	0.6	N/A	N/A	16.126	16.056	N/A	0.4	N/A
0.8	(1,1, 1)	1.998	12.16	12.262	12.3033	1.2	0.3	1.2	13.213	13.256	13.2993	0.7	0.3	0.7
0.8	(3,1,1)	2.032	13.064	12.901	13.199	1.0	2.3	1.1	12.212	11.957	12.263	0.4	2.6	0.4
0.8	(1,3,1)	2.041	12.538	12.723	12.725	1.5	0	1.6	11.735	11.809	11.807	0.6	0	1.2
0.8	(1,1,3)	2.033	12.352	12.513	12.4949	1.12	0.1	1.2	11.501	11.584	11.5609	0.5	0.2	1.1
0.8	(1,1,5)	2.095	12.939	12.951	12.8978	0.3	0.4	0.1	9.924	10.136	10.0878	1.7	0.5	1.8

As a final remark, from the open Jackson network, the feedback impact on traffic intensity rate $\rho = \lambda/[u(1 - \delta)]$ can be explained in two different ways, by either increasing the input arriving rate from λ to $\lambda/(1 - \delta)$ or by losing capacity from u to $u(1 - \delta)$. From

our computing experience with the performance of a tandem queue with feedback, both methods achieve the same results. Actually the equivalence can be easily verified through simple matrix algebraic operation on (3.2) and shown that both P_j are the same under these two approaches. Since P_j are the same the succeeding calculations of performance measures obtain the same results. In analyzing the impact of ST of tables 3.3 and 3.4, we used the arrival increase method. However, it's better to use the method of capacity loss when there is also a machine breakdown issue, otherwise the outcome will differ largely from the simulation results. This can be seen from (3.9) and (3.10), the calculation of ST is affected by R and G^* and G^* is affected by capacity, not arriving rate.

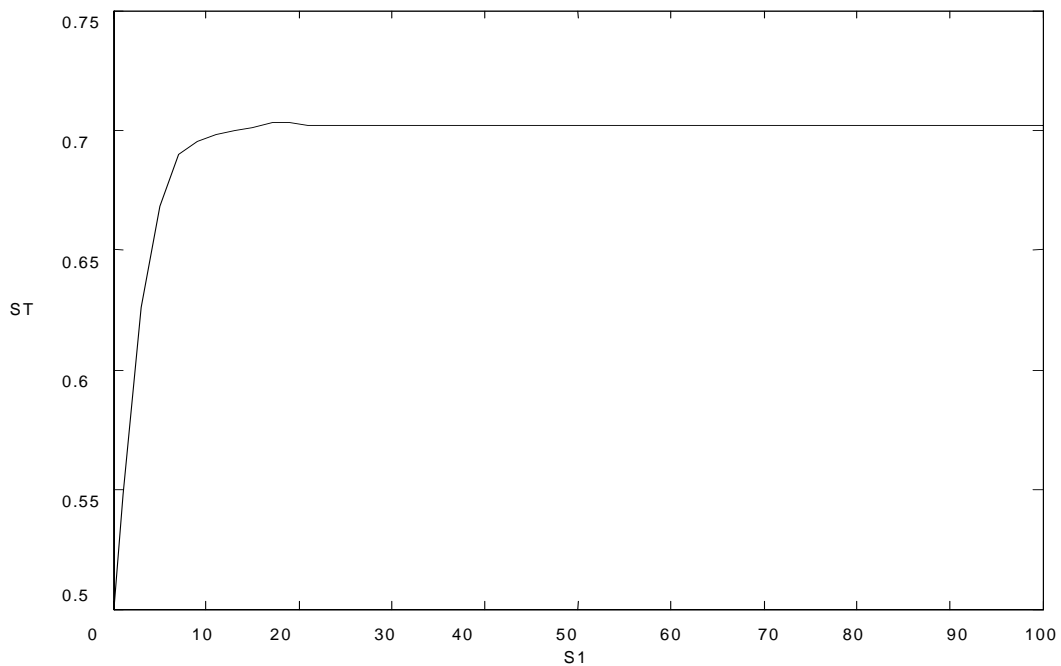


Fig. 3.3 Sojourn time of stage 1 as a function of S1 for a two-stage PF with $\rho = 0.5$.

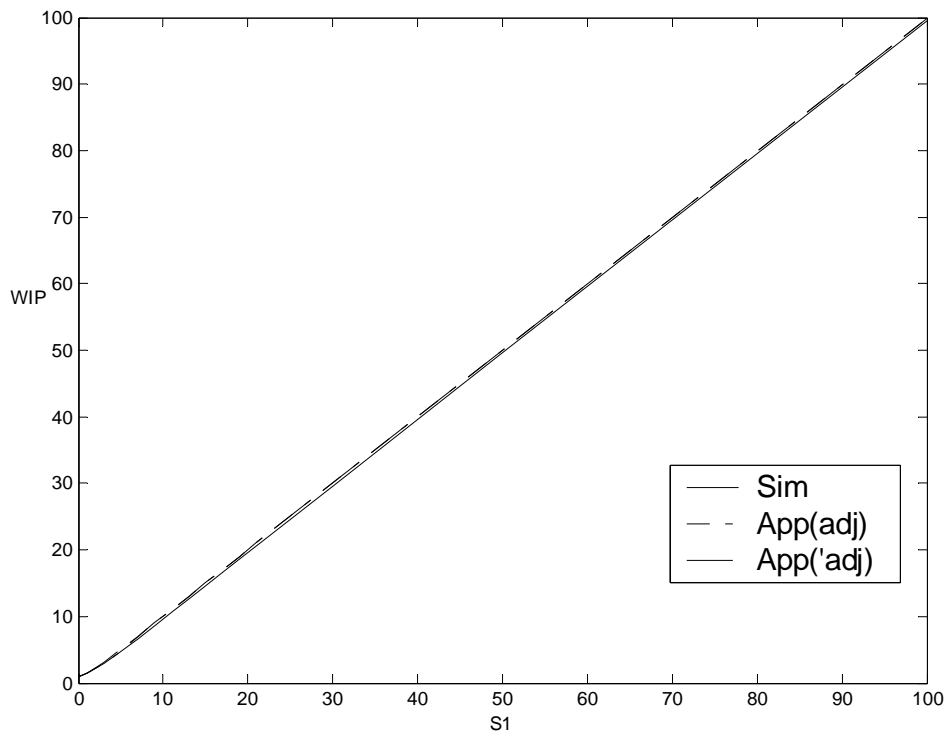


Fig. 3.4 WIP as a function of S1 for a two-stage PF with $\rho = 0.5$.

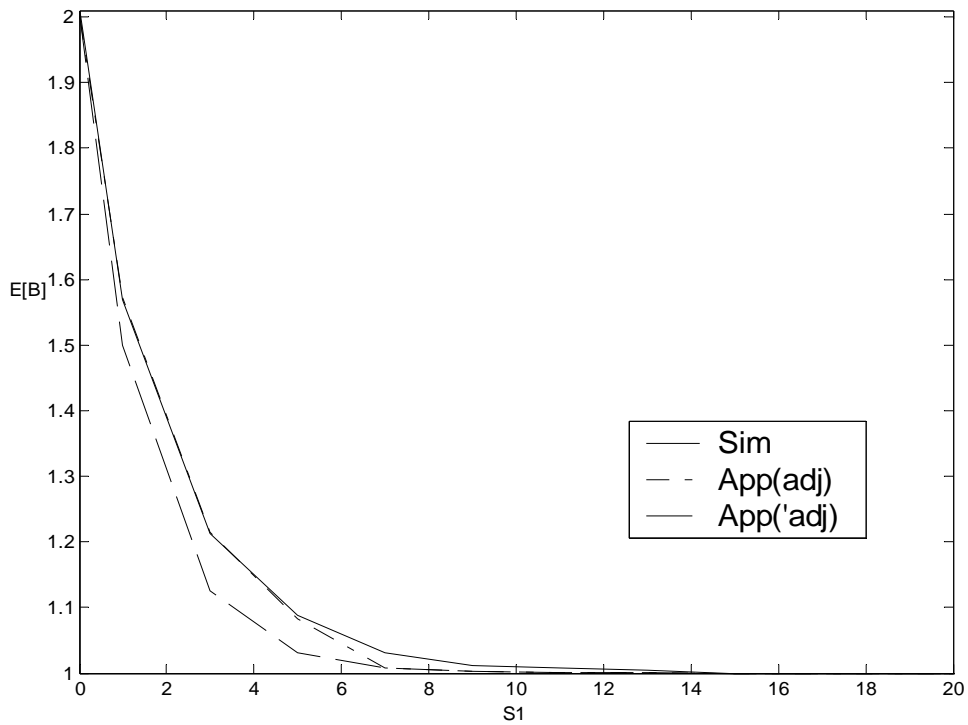


Fig. 3.5 $E[B]$ as a function of S1 for a two-stage PF with $\rho = 0.5$.

3.5 Extension

For the derivation of (r, q) policy, it's natural by using the fact that it is built upon base-stock policy. The key performance measures such as the steady-state backorder level can therefore be represented as the equal weighted sum of respective performance measures at different levels of inventory positions (Axsäter, 2000 or Zipkin, 2000). Specifically, after some algebraic operations, we may express the above argument as:

$$\begin{aligned}
 E[B] &= \sum_{S=r+1}^{r+q} E[B(S)] \\
 &= \left(\frac{1}{q}\right) \pi \mathbf{P} (\mathbf{I} - \mathbf{P})^{-2} (\mathbf{I} - \mathbf{P}^q) \mathbf{P}^r \mathbf{e}
 \end{aligned} \tag{3.14}$$

Note when $q = 1$, (3.14) becomes (3.7). To illustrate our argument, assume we have a two-echelon SC: a production facility (PF) directly serves 4 identical retailers. The PF uses base stock policy to control its inventory while the retailers use (r, q) policies to control their stocks. The demand process at the PF is not Poisson but it is a superposition of several independent renewal processes, which under suitable conditions resembles a Poisson process (Svoronos and Zipkin, 1988). Assume the PF produces in units of retailer batches and each retailer has its dedicated transporter. Here we follow Svoronos and Zipkin (1988) and assume the arrival process at PF as Poisson processes. We then express the aggregated arrival rate at the PF as $N\lambda_R/q$, where λ_R is the arrival rate for each retailer and N is the number of the retailers. We also approximate the arrival process at the respective transit stage as Poisson process with mean λ_R/q . Finally we use the same modeling approach for DC as shown in subsection 3.3.1 to model retailer activity, assuming that the retailer processes its inventory with high efficiency at near zero operation time. Alternatively, we can formulate this problem as a 2-stage SC, with respective retailer-stocks representing planned inventories at the second stage. Using (3.14) and the fact: $WIP = E[I_I] + E[\text{inventory in transit}]$, we obtain performance measures for different combinations of

inventory control parameters at each stage as listed in table 3.5.

Table 3.5 Approximation vs. simulation for the case where retailers use (r, q) policies.

ρ_1	ρ_2	S	r	q	WIP			$E[B]$		
					Sim	App	%err	Sim	App	%err
0.5	0.125	0	0	2	0.53	0.571	7.74	0.659	0.83	25.95
0.25	0.063	1	0	4	1.001	1.016	1.50	0.091	0.12	31.87
0.167	0.042	3	0	6	2.978	2.974	0.13	0.041	0.045	9.76
0.125	0.031	5	0	8	4.988	4.986	0.04	0.031	0.033	6.45
0.8	0.200	0	0	2	0.915	1	9.29	4.287	5.777	34.76
0.4	0.100	1	0	4	1.003	1.044	4.09	0.309	0.48	55.34
0.267	0.067	3	0	6	2.939	2.929	0.34	0.109	0.129	18.35
0.2	0.050	5	0	8	4.967	4.96	0.14	0.081	0.088	8.64

Here S is the base stock level at PF and r and q represent reorder point and fixed order quantity at the retailers respectively. Under the arrival assumptions at respective echelons, ρ_1 and ρ_2 are calculated traffic intensities by changing different level of q and letting λ_R fixed at either 0.5 or 0.8. u is fixed at 2 for the server at respective echelons (No feedback concern in this case). Also for simplicity we don't consider breakdown issue. From the table we see acceptable accuracy exists when ρ_1 is low. We also test other cases when ρ_1 is high and q is large by varying λ_R . Unfortunately the approximation is not satisfactory for $E[B]$ on most of the test cases. Some tests show Erlang distribution may be more appropriate than the proposed Poisson distribution for the arrival process at respective echelons. However such conjecture is related to phase type arrival and needs further analytic efforts and numerical verifications.

As stated in section 3.1, we may use QBD process to achieve the goal of more modeling flexibility. For example if we want to model multi-server at each subsystem with each server suffering random breakdowns, we have an $M/PH/m$ queueing system at each subsystem. To model such queueing system by using the QBD approach, we may express

the state space as $x(1)\cdots x(i)\cdots x(m)$ where $n =$ customer number, $x(i) = 0$ (down) or 1 (on), $1 \leq i \leq m$, and have 2^m states for $n \geq m$. We conjecture that the QBD modeling approach for each subsystem may be treated independently from the linkage of the whole SC. To justify our argument, we employed the same 2-stage example (Also no feedback concern in this case) as in section 3.4 with some modifications that there are multiple parallel machines at the PF and so are there at the second stage. Specifically we assume 2 servers for each stage. Assume the parameters are the same as in section 3.4. We form a QBD process for the decomposed server queue at each stage. Table 3.6 lists the results for different combinations of base-stock level at each stage.

Table 3.6 Approximation vs. simulation for cases with multi-server and breakdowns

ρ	S1	S2	WIP			E[B]		
			App	Sim	%err	App	Sim	%err
0.5	0	0	1.597	1.620	1.42	3.594	3.224	11.48
0.5	1	0	1.982	1.883	5.26	2.579	2.491	3.53
0.5	3	0	3.371	3.256	3.53	1.968	1.853	6.21
0.5	5	0	5.140	5.078	1.22	1.737	1.676	3.64
0.8	0	0	8.214	7.076	16.08	16.428	13.846	18.65
0.8	1	0	8.322	7.176	15.97	15.536	13.049	19.06
0.8	3	0	8.819	7.355	19.90	14.033	10.981	27.79
0.8	5	0	9.624	8.336	15.45	12.839	10.237	25.42

Clearly the accuracy is degraded when traffic intensity is high, but not significantly serious, as compared to all the previous examples.

In short, it is possible a more general framework to accommodate for versatile control policies may be developed by combining the QBD technique and L & Z. Since the basic assumption for the approximation model of L & Z is that the queueing system at each subsystem is independent. Under this assumption, we use QBD to model individual queueing systems. However, the QBD approach often faces the problem of largeness, i. e.,

too many states may make the solution intractable (For example, for the above mentioned $M/PH/m$ queueing system, if we have 20 parallel machines, the states become more than one million). The challenge lies in how large and how sophisticated the QBD modeling approach can allow as well as how accuracy this combining process can provide. All these need further study as well as thorough numerical verifications.

3.6 Concluding remarks

We have demonstrated that by using the matrix analytical approach, the evaluation of a complex SC where all the participants, including PF, transporters, DC and retailers use base-stock control policies, performs as expected through simulation verification. The relative errors between App and Sim are all below 10% for retailers adopting MTO policies. When all the retailers adopt the MTS policy, numerical studies also show that the approximation is accurate for medium traffic intensity and acceptable for high traffic intensity. In this chapter the results are somehow similar to those of Zipkin (1995) where the base stock level at the end stage is set to zero. The present study shows that the matrix analytical approach is very accurate, not just for the application of tandem processing queue as reported by L & Z and Zipkin (1995) but also for the application of tandem SC where the end stage can be of a distribution system. In the literature on the stochastic production-distribution system, most models are developed and analyzed separately. Unlike our model, these evaluation models are usually difficult to integrate as one single model.

The most significant contribution of this chapter is that we proposed an originaive and useful system design and analysis tool for evaluating the performance of an integrated stochastic SC. Although a rich body of multi-echelon inventories systems in the literature, which use the same base-stock policies as we used herein, our idea is to provide a viable scheme for solving integrated stochastic supply network in a flexible and realistic way. We

used the simplest inventory control scheme of base-stock as the first step towards more involved inventory control technique. Under the matrix analytical approach, decision makers can easily formulate stochastic and/or factors of uncertainties, which are often encountered in real life, as adequate queueing form and later integrate them together as a single tandem queue. The performance measures are then readily available by simple matrix-manipulated computation. In this chapter we also found that, the Hyper-exponential queue $M/H_m/I$ can be used adequately to model a distribution subsystem of a supply chain. The phase-type structure can then be handled as a usual QBD process. We illustrate how it works by proposing a special structure, under which the distribution subsystem behaves just like an $M/M/I$ queue. However numerical studies show it is not limited to such special form by adequate modification of L & Z (we omit the numerical details here and recapture it in chapter 5). We believe this modeling approach introduced herein is new in supply chain study. The other finding is that the sojourn time of an order at the beginning stage of a tandem queue may differ from the other stages. This seems to violate the inherent theory of an open Jackson network. However, sensitivity study shows that the matrix analytical approach still approximates well. In the extension section we test the applicability of the proposed approach for another control scheme as well as for multi-server setting. We employed two 2-echelon problems. Numerical studies of (r, q) policy are satisfactory for low to medium traffic intensities when arrival rates of individual retailers are fixed at either 0.5 or 0.8. In the multi-server case the approximated results are more satisfactory with medium traffic intensities than with heavy traffic intensities. Generally speaking, we see the promising future of the proposed model as a quick and accurate SC evaluation tool not just for base stock inventory control schemes but also for (r, q) policy employed at the retailer site if traffic intensity of the studied queueing system is medium or low. Another advantages of the study in this chapter is that the proposed method herein seems more tractable when compared with existing multi-echelon stochastic models in the literature,

which often used more involved stochastic process to derive performance measures of interests. Finally, the closed-form solutions of the current model may be used as later SC optimization applications.



Chapter 4 Non-stationary arrival and unreliable service processes

4.1 Introduction

As is well known the real supply system is seldom fault-free. Unfortunately operation-without-error is usually assumed in most stochastic models of SCM. The server may breakdown sometimes. The operator may not normally work from time to time. The over-simplified assumptions in all the stochastic models made them far away from practical application. The cause of non-stationary demand may be due to seasonal product, short product life cycle time, etc. Another assumption, which is usually used, is single server. The assumption of single-server setting often seen in existing literature does not seem to be practical. All these supply and/or demand uncertainties and inadequate assumptions should be concerned and built into the system dynamic of an SC to make the developed model more robust and practical to use. The objective of this chapter is to find an efficient analytic modeling approach. This approach should be able to capture as many uncertainty factors as possible embedded in a supply system under random environment. A Markov-modulated Poisson process (MMPP) is a stochastic process whose arrival rate is “modulated” by an irreducible CTMC and is widely used as telecommunication traffic modeling (Trivedi, 2002). In this chapter we tried to use MMPP to model supply and demand uncertainties and used MGM to solve the proposed QBD model. The application of the proposed model was illustrated by numerical studies. The results provide managerial insights regarding adequate supply resource design under a MTO supply policy.

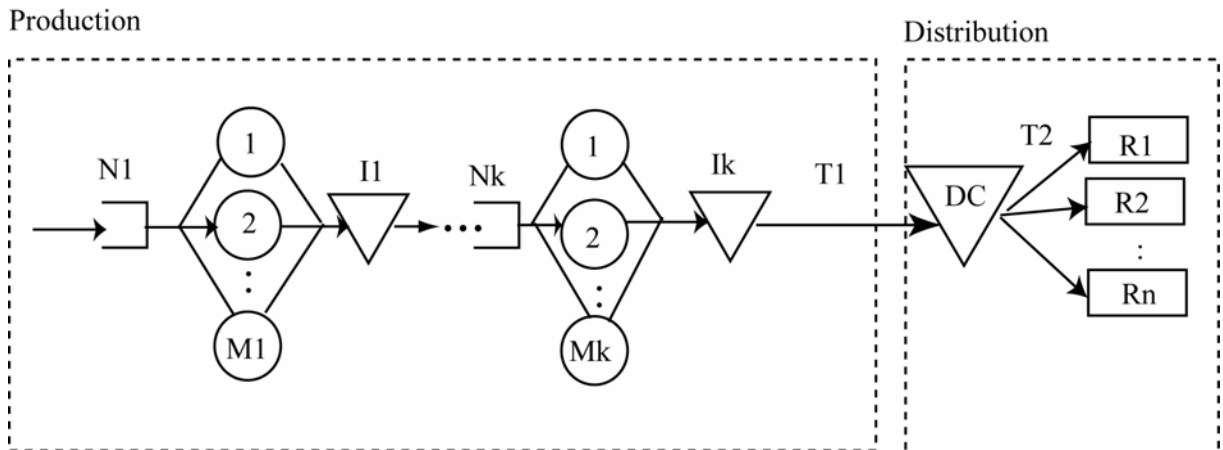
In Kendall notation our studied problem is of $M/M/m/\infty/\infty/FCFS$ type with failure-prone server. Hereinafter we will refer to this queueing system as $M/M^*/m$. According to Jackson’s theorem, QN composed of $M/M/1(m)/\infty/\infty/FCFS$ nodes has

product-form solution. The product-form solution even exists for tandem QN with feedback or for arbitrarily linked network with Markovian routing (Trivedi, 2002). For generalized Jackson network such as the one we studied herein where the renewal arrival process need not be Poisson and i.i.d. service times that need not follow exponential distribution, the stationary distribution usually does not have explicit analytic form. For such generalized Jackson network, approximation method such as fluid or diffusion is usually sought for. However state-dependent service times is more restrictive and therefore is not allowed for generalized Jackson network (Chen and Yao, 2001). Since our service times are state-dependent we abandon the effort on deriving approximation model. Instead, we try to use direct approach, i.e. drawing state transition diagram of the CTMC under study, writing down the balance equations, and proceeding to solve them through an algorithmic approach. Though the process is cumbersome, this modeling approach can represent system dynamics of complex supply systems more faithfully. It generates an exact solution.

4.2 QBD process decomposition

A Markov process is called a QBD process when its infinitesimal generator matrix has a tri-diagonal block form. In this chapter we used MMPP to model demand and repairman uncertainty. MMPP combined with existing machine-repair model can be easily formulated as QBD processes to adequately represent versatile uncertainty environment in a supply-demand system. Assume the SC model under study is depicted in fig. 4.1. There are two echelons: production and distribution. We used decomposition approach to treat each supply stage as isolated queueing system. First we apply Jackson's rule to find traffic process. Then we derive analytic model for each stage under possible resource allocation combinations subject to available capacity restriction. Optimal resource allocation for the whole supply network was then solved as the trade-off among costs of inventory holding, customer waiting, machine repair and normal operation. Below is detail problem

description. Assume k workstations arranged in tandem to produce one single product type. Each workstation is composed of m parallel serving machines with r repairmen to attend the occasional machine breakdown. Actually the supply system under study is a classical machine-repair problem with modification of considering repairman on and off. Assume the machine is subject to random malfunction whether it's active or not.



Legend: N_i : Input buffer; I_i : Output buffer; T_i : Transporter; DC : Distribution Center; R_i : Retailer

Fig. 4.1 A multi-server supply network of production and distribution echelons.

Note that since the work in this chapter is a matrix-oriented study, we neglect the bold face identification for a vector or a matrix for most of the cases when it can be easily identified from the context. Also, server may be referred to as machine or transporting vehicle.

The objective is to decide the optimal server and repairman deployment in a tandem setting such that the total operation cost is minimized under capacity restriction. Below we introduce QBD process, which forms the backbone structure of our analytic model. The form of the infinitesimal generator of a QBD process representing $M/M/m$ system with server breakdown and repair can be expressed as the following (Neuts and Lucanton, 1979; Neuts, 1994):

- Step 4. Compute ρ' for each possible solution. If the queue is stable (< 1), the solution is feasible. Then compute average number of operative and under-repaired machines. Otherwise it is abandoned. Continue this way until all possible solutions have been enumerated.
- Step 5. Construct \tilde{Q} .
- Step 6. Compute matrix R (see later definition).
- Step 7. Compute stationary probability vector x for \tilde{Q} .
- Step 8. Compute respective performance measure, specifically average queue length.
- Step 9. Continue the above procedure until all queues have been decomposed and analyzed.
- Step 10. Solve the optimization problem for the integrated system

4.3 MMPP modeling of uncertainties

To understand MMPP we explained with a state transition diagram of a simple two-state Markov process as illustrated in fig. 4.2. σ_i is state transition rate. When system is in state 0 the Poisson arrival rate is λ_0 . When system is in state 1 the arrival rate is λ_1 . This explains what “modulated” arrival process means. In the following we show how to derive the sub-matrix of (4.1) considering other uncertainty factors embedded in an SC. Specifically, we assume the uncertain factor of supply is mainly from server breakdown with additional repairman on-off states. The demand process is assumed non-stationary with two modes, low and high, which is often seen in seasonal products. Assume the server is unreliable with reliable (stationary) or unreliable (non-stationary) repairman (demand). We model all the (repairman/demand) uncertainties as MMPP models as shown in cases 4.2 to 4.4. First we show the typical machine-repair problem without considering any uncertainty factors embedded for benchmark purpose. Also notice that there are two

queues, customer queue and server-repair queue embedded in all the analytic models. The kronecker algebra is used in our solution procedure to conquer the problem of largeness.

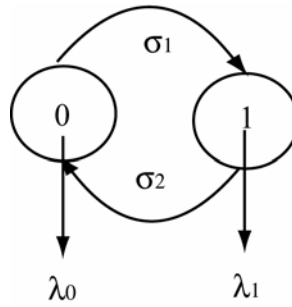


Fig. 4.2 A simple MMPP process.

Case 4.1 Stationary demand, unreliable server with reliable repairman

Basically this is the machine-repair problem. From Neuts (1994) (pp. 274-286) and also from (4.1), we get the following customer queue:

$$\begin{aligned}
 A_{00} &= Q^{(1)} - \text{diag}(\lambda_0, \lambda_1, \dots, \lambda_m) = Q^{(1)} - \Delta(\lambda), \\
 A_{01} &= A_{12} = \dots = A_{m-1,2} = A_0 = \Delta(\lambda), \\
 A_{i0} &= \text{diag}\{u \min(i, j), 0 \leq j \leq m\}, \text{ for } 1 \leq i \leq m-1, \\
 A_{i1} &= Q^{(1)} - \Delta(\lambda) - A_{i0}, 1 \leq i \leq m-1, \\
 A_1 &= Q^{(1)} - (\Delta(\lambda) + A_2), \\
 A_2 &= \text{diag}(0, u, 2u, \dots, mu) = \Delta(u),
 \end{aligned}$$

where $\text{diag}(\cdot)$ represents diagonal matrix with diagonal element specified by its parameters. $\Delta(\lambda)$ here represents state dependent arrival rate. Since in this case we model stationary demand process and there is $m+1$ machine states we have $m+1$ arrival states: $\lambda_0, \lambda_1, \dots, \lambda_m$ with identical values. $Q^{(1)}$ is the generator of the CTMC model of the embedded server-repair queue. Let states represent number of operative machines. It's easy to construct the following generator matrix:

$$\begin{aligned}
 Q_{j,j-1}^{(1)} &= j\zeta, 1 \leq j \leq m, \\
 Q_{j,j+1}^{(1)} &= \gamma \min(r, m-j), 0 \leq j \leq m-1, \\
 Q_{j,j}^{(1)} &= -j\zeta - \gamma \min(r, m-j), 0 \leq j \leq m.
 \end{aligned}$$

From $Q^{(1)}$ we can solve it recursively and obtain the following well-known stationary solution vector π .

$$\begin{aligned}\pi_0 &= \left[\sum_{j=0}^{m-r+1} \frac{1}{j!} \left(\frac{r\gamma}{\zeta}\right)^j + \sum_{j=m-r+2}^m \frac{1}{j!} \left(\frac{m\gamma}{\zeta}\right)^j \prod_{\tau=1}^{j-m+r-1} \left(1 - \frac{\tau}{r}\right) \right]^{-1} \\ \pi_j &= \frac{1}{j!} \left(\frac{r\gamma}{\zeta}\right)^j \pi_0, & 0 \leq j \leq m-r+1, \\ \pi_j &= \frac{1}{j!} \left(\frac{r\gamma}{\zeta}\right)^j \prod_{\tau=1}^{j-m+r-1} \left(1 - \frac{\tau}{r}\right) \pi_0, & m-r+2 \leq j \leq m.\end{aligned}$$

It's known input rate must be less than output rate to maintain a stable queue from the basic queueing theory. Thus the following equation must hold for demand queue under study:

$$\rho = \left(\sum_{j=0}^m \pi_j \lambda_j \right) \left(u \sum_{j=1}^m j \pi_j \right)^{-1} < 1.$$

And for $m \geq 2$ and customer number larger or equal to machine number the following repetitive equation must hold, which constitutes the main structure of a MGM method:

$$\begin{aligned}\Delta(\lambda) + R[Q^{(1)} - (\Delta(\lambda) + \Delta(u))] + R^2 \Delta(u) &= 0, \\ x_i &= x_{m-1} R^{i-m+1}, \\ i &\geq m.\end{aligned}$$

The objective is to find matrix R , which satisfies the above equations. Here x represent stationary probability vector for solving (4.1). Starting from initial guess of $R = 0$ we use successive substitution to obtain R as follows.

$$R_{k+1} = -[\Delta(\lambda) + R_k^2 \Delta(u)] * [Q^{(1)} - (\Delta(\lambda) + \Delta(u))]^{-1}.$$

When $m = 1$ it becomes a well-known $M/M/1$ queue with machine breakdown and repair problem, which can be dealt with algorithmically as in the above approach. In order to obtain x_i , $i \leq m - 1$ we use the same successive substitution method for R as suggested in Neuts (1994).

$$\begin{aligned}
\mathbf{x}_0 &= [\mathbf{x}_0(A_{00} + \Delta_0) + \mathbf{x}_1 A_{10}] \Delta_0^{-1}, \\
\mathbf{x}_i &= [\mathbf{x}_{i-1} \Delta(\lambda) + \mathbf{x}_i(A_{i1} + \Delta_i) + \mathbf{x}_{i+1} A_{i+1,0}] \Delta_i^{-1}, 1 \leq i \leq N-2, \\
\mathbf{x}_{N-1} &= \{\mathbf{x}_{N-2} \Delta(\lambda) + \mathbf{x}_{N-1}[A_{N-1,1} + \Delta_{N-1} + R\Delta(u)]\} \Delta_{N-1}^{-1},
\end{aligned}$$

where Δ_0 and $\Delta_i, 1 \leq i \leq N-1$ represent $-\text{diag}(A_{00})$ and $-\text{diag}(A_{i,1})$ respectively, and $\text{diag}(A_{\bullet})$ is a matrix composing of diagonal elements of original matrix: A_{\bullet} . The successive method will continue until the difference between the previous solution and the current one is within some acceptable level. After x is obtained through numerical method the queue length, which is defined as average number in the system can be obtained through expectation formula. The standard deviation can also be calculated. See Neuts and Lucanton (1979) for more details. Appendix B.1 shows related proofs.

Case 4.2 Stationary demand, unreliable server with unreliable repairman

Here we modify the well-known machine-repair problem as in case 4.1 with additional construction that the repairman can also take on and off states. This consideration is more realistic since operator may ill or take breaks from time to time. We model the individual two-state Markov process for each repairman as follows. Assume state 0 represents off and state 1 on. We have the following infinitesimal generator and rate matrix (here the matrix represents available number of repairmen) for this MMPP:

$$\begin{aligned}
G_j &= \begin{bmatrix} -\sigma_{j1} & \sigma_{j1} \\ \sigma_{j2} & -\sigma_{j2} \end{bmatrix}, \text{ where } 1 \leq j \leq r, \text{ with arrival rate} \\
\Lambda_j &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.
\end{aligned}$$

(Note here the arrival rates are constants, which is a special case of MMPP. However when we combine available repairman capacity into server-repair queue, we derive the “true” repair rate). And we get

$$\begin{aligned}
G &= G_1 \oplus G_2 \oplus \cdots \oplus G_r, \text{ and} \\
\Lambda &= \Lambda_1 \oplus \Lambda_2 \oplus \cdots \oplus \Lambda_r,
\end{aligned} \tag{4.2}$$

where \oplus represents Kronecker sum. Note σ_{j1} represents the transition rate from 0 to 1 and σ_{j2} represents the transition rate from 1 to 0 for repairman j . $A \oplus B$ of two matrices A and B of sizes $m \times m$ and $n \times n$ is defined as follows:

$$A \oplus B = A \otimes I_n + I_m \otimes B,$$

here we use I_x to represent identity matrix of size x and \otimes to represent Kronecker product.

$A \otimes B$ of two matrices A and B of sizes $m \times m$ and $n \times n$ is defined as follows:

$$\begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \cdots & a_{2,m}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1}B & a_{m,2}B & \cdots & a_{m,m}B \end{bmatrix}.$$

Therefore (4.2) can be rewritten as

$$\begin{aligned} G &= G_1 \oplus G_2 \oplus \cdots \oplus G_r \\ &= (G_1 \otimes I_2 + I_2 \otimes G_2) \oplus \cdots \oplus G_r \\ &= G_1 \otimes I_2 \otimes I_2 \otimes \cdots + I_2 \otimes G_2 \otimes I_2 \otimes \cdots + I_2 \otimes I_2 \otimes \cdots \otimes G_r, \end{aligned}$$

and,

$$\begin{aligned} \Lambda &= \Lambda_1 \oplus \Lambda_2 \oplus \cdots \oplus \Lambda_r \\ &= \Lambda_1 \otimes I_2 \otimes I_2 \otimes \cdots + I_2 \otimes \Lambda_2 \otimes I_2 \otimes \cdots + I_2 \otimes I_2 \otimes \cdots \otimes \Lambda_r. \end{aligned}$$

Finally, we obtain all the sub-matrices of (4.1) in this case as follows:

$$\begin{aligned} A_{00} &= Q^{(2)} - \lambda I_{(m+1) \times 2^r}, \\ A_{01} &= A_{12} = \cdots = A_{m-1,2} = A_0 = \lambda I_{(m+1) \times 2^r} = \Delta(\lambda), \\ A_{i0} &= \text{diag}\{u \min(i, j), 0 \leq j \leq m\} \otimes I_{2^r}, 1 \leq i \leq m-1, \\ A_{i1} &= Q^{(2)} - A_{i0} - \Delta(\lambda), \quad 1 \leq i \leq m-1, \\ A_1 &= Q^{(2)} - \Delta(\lambda) - A_2, \\ A_2 &= \text{diag}(0, u, 2u, \dots, mu) \otimes I_{2^r} = \Delta(u). \end{aligned}$$

Since here we incorporate repairman on-off factor. Q is the joint process of $Q^{(1)}$ and G .

Applying kronecker operation we get $Q^{(2)} = Q^{(1)} \oplus G = Q^{(1)} \otimes I_{2^r} + I_{(m+1)} \otimes G$. However since

the availability of repairman will affect the joint operative capacity, we have to make some

modification. Further, we consider the repairman who cannot affect the failure rate of machine. Thus we only have to make modification on upper diagonal sub-matrix and diagonal sub-matrix as follows. Define

$$J_{j+1} = \min((m-j)I_{2^r}, \Lambda), 0 \leq j \leq m-1.$$

Then

$$\begin{aligned} Q_{j,j+1}^{(2)} &= \gamma J, 0 \leq j \leq m-1, \\ Q_{j,j}^{(2)} &= -j\zeta \times I_{2^r} - \gamma \min((m-j)I_{2^r}, J_{j+1}) + G, 0 \leq j \leq m, J_{m+1} = 0. \end{aligned}$$

Notice the matrix element $Q_{i,j}^{(2)}$ is of the same size as G , in this case 2^r . For general case m

≥ 2 the queue is stable if and only if $\rho' = \left(\sum_{j=0}^m \sum_{k=1}^{2^r} \pi_{jk} \lambda_j \right) \left(u \sum_{j=1}^m \sum_{k=1}^{2^r} j \pi_{jk} \right)^{-1} < 1$. The stability

test of the following cases 3 and 4 are similar as the one done. The findings of $x_i, i \leq m-1$ and R here and those for cases 3 and 4 are the same as in case 4.1.

Case 4.3 Non-stationary demand, unreliable server with reliable repairman

The infinitesimal generator of a two-mode non-stationary demand process is as follows.

$$\begin{aligned} D &= \begin{bmatrix} -d_1 & d_1 \\ d_2 & -d_2 \end{bmatrix}, \text{ with arrival rate} \\ \Lambda &= \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \end{aligned}$$

where d_1 represents transition rate from “low” season to “high” season and vice versa for d_2 and λ_i is respective demand for this MMPP. Applying kronecker algebra we obtain the server-repair process $Q^{(3)} = Q^{(1)} \oplus D = Q^{(1)} \otimes I_2 + I_{m+1} \otimes D$.

The sub-matrices of (4.1) in this case are as follows:

$$\begin{aligned}
A_{00} &= Q^{(3)} - \Delta(\lambda), \\
A_{01} &= A_{12} = \dots = A_0 = I_{m+1} \otimes \Lambda = \Delta(\lambda), \\
A_{i0} &= \text{diag}\{u \min(i, j), 0 \leq j \leq m\} \otimes I_2, 1 \leq i \leq m, \\
A_{i1} &= Q^{(3)} - \Delta(\lambda) - A_{i0}, \quad 1 \leq i \leq m-1, \\
A_1 &= Q^{(3)} - \Delta(\lambda) - A_2, \\
A_2 &= \text{diag}(0, u, 2u, \dots, mu) \otimes I_2 = \Delta(u).
\end{aligned}$$

Case 4.4 Non-stationary demand, unreliable server with unreliable repairman

In this case we combine cases 2 and 3. By using Kronecker algebra we obtain the machine queue as $Q^{(4)} = Q^{(2)} \oplus D = Q^{(2)} \otimes I_2 + I_{(m+1) \cdot 2^r} \otimes D$. The sub-matrices of (4.1) in this case are as follows:

$$\begin{aligned}
A_{00} &= Q^{(4)} - \Delta(\lambda), \\
A_{01} &= A_{12} = \dots = A_0 = I_{m+1} \otimes I_2 \otimes \Lambda = \Delta(\lambda), \\
A_{i0} &= \text{diag}\{u \min(i, j), 0 \leq j \leq m\} \otimes I_{2^r} \otimes I_2, 1 \leq i \leq m, \\
A_{i1} &= Q^{(4)} - \Delta(\lambda) - A_{i0}, \quad 1 \leq i \leq m-1, \\
A_1 &= Q^{(4)} - \Delta(\lambda) - A_2, \\
A_2 &= \text{diag}(0, u, 2u, \dots, mu) \otimes I_{2^r} \otimes I_2 = \Delta(u).
\end{aligned}$$

Next we add cost parameters and see the impact of uncertainties on system performance. Intuitively we may think if the effect of uncertainty is significant, it seems better to invest more resources (server, repairman) to fight against uncertainty. However, except for the fact that the investment cost may diminish the benefit obtained from such policy. There is other side effect we may want to explore. For simplicity we omit the investigation of incorporating investment analysis here. Other related discussion will be explored in section 4.4. The objective in the current study is to minimize the total cost as follows

$$TC = c_h E[L] + c_o E[O] + c_r E[RE] + c_b E[B]. \quad (4.3)$$

Since we adopt an MTO production policy, the first term is the work-in-process (WIP) inventory holding cost in terms of the cost of average queue length in the system, the middle term is the operation cost, the third term is the repair cost, and the final term is the

backorder cost. Under MTO, $E[L]$ should be equal to $E[B]$. $E[O]$ is equal to $\sum_{i=1}^m i \cdot \pi_i \cdot \mathbf{I}$,

where \mathbf{I} is a unity vector whose dimension is chosen to fit the context. For example, we may have i operative machine with 2 repairmen. There are 4 possible combined repairman states. Then the dimension of \mathbf{I} is 4. And if $\gamma = \zeta$, it can be shown: $E[O] = E[RE]$. In general cases, the ratio of the average available machine to the average machine to be repaired is equal to $\frac{\gamma}{\zeta}$ as illustrated in the following statements. Notice this property is not explicitly related to the repairmen. This is because the available repairman capacity has been modified and incorporated in calculating expected number of under-repaired machines. The repairman state has been accounted for in calculating π . The proofs are listed in appendix B.2 and B.3.

PROPOSITION 1: If $\gamma = \zeta$, then $E[O] = E[RE]$.

COROLLARY 1: If $\gamma \neq \zeta$, then $E[O] = \frac{\gamma}{\zeta} E[RE]$.

The above statement reveals that if γ is greater than ζ then the average number of operative machines is greater than the average number of machines under repair. Decision managers can use this fact and adjust γ and/or ζ accordingly to see its impact on TC.

4.4 Illustrative examples

In this section we show how to derive the minimum trade-off cost. Assume the respective parameters as set as follows:

$$\sigma_{j1} = 0.5, \sigma_{j2} = 0.05, \gamma = 2.5, \zeta = 0.25, d_1 = d_2 = 0.01, \lambda_1 = 0.5, \lambda_2 = 1.5, \lambda = 1, u = 1, \\ c_h = 0.5, c_b = 60, c_0 = 1, c_r = 20.$$

First we start the derivation process of the generator matrix $Q^{(i)}$. Since our study focuses on how to use MMPP to model various uncertainty cases, we skip case 4.1. For

case 4.2, assume $m = 3, r = 2$. We arrange states lexicographically as $(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), \dots, (3, 1, 1)$, where the first parameter denotes the number of available machines, the second parameter denotes the state of the first repairman, and the third parameter denotes the state of the second repairman. Then the generator of repairman is

$$\begin{aligned}
 G &= G_1 \otimes I_2 + I_2 \otimes G_2 \\
 &= \begin{bmatrix} -\sigma_{11} & & \sigma_{11} & & & \\ & -\sigma_{11} & & \sigma_{11} & & \\ \sigma_{12} & & -\sigma_{12} & & & \\ & \sigma_{12} & & -\sigma_{12} & & \end{bmatrix} + \begin{bmatrix} -\sigma_{21} & \sigma_{21} & & & & \\ \sigma_{22} & -\sigma_{22} & & & & \\ & & -\sigma_{21} & \sigma_{21} & & \\ & & \sigma_{22} & -\sigma_{22} & & \end{bmatrix} \\
 &= \begin{bmatrix} -\sigma_{11} - \sigma_{21} & \sigma_{21} & \sigma_{11} & & & \\ \sigma_{22} & -\sigma_{11} - \sigma_{22} & & \sigma_{11} & & \\ \sigma_{12} & & -\sigma_{12} - \sigma_{21} & \sigma_{21} & & \\ & \sigma_{12} & \sigma_{22} & -\sigma_{12} - \sigma_{22} & & \end{bmatrix}, \text{ and} \\
 \Lambda &= \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} \\
 J_1 &= \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix}, J_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix}, J_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}.
 \end{aligned}$$

Applying kronecker algebra and making necessary modification as illustrated in section 4.3, we get

$$Q = \begin{bmatrix} \Xi_1 & \Gamma_1 & & & \\ \Psi_2 & \Xi_2 & \Gamma_2 & & \\ & \Psi_3 & \Xi_3 & \Gamma_3 & \\ & & \Psi_4 & \Xi_4 & \end{bmatrix}. \tag{4.4}$$

Here all sub-matrices are of size 4×4 and are defined below:

$$\Gamma_i = \lambda J_i, \Psi_j = (j-1)\mathcal{I}, \Xi_1 = G - \Gamma_1, \Xi_4 = G - \Psi_4, \Xi_k = G - \Psi_k - \Gamma_k, \\
 1 \leq i \leq 3, 2 \leq j \leq 4, k = 2, 3.$$

Similar approach applies to case 4.3 with simpler derivation. So we omit the detail. Next we investigate case 4.4. Assume $m = 2, r = 1$. $Q^{(4)}$ has the size $2 \times (m + 1) \times 2^r = 12$. If we use $\{x(t), y(t), z(t)\}$ to represent the states, $x(t)$ is the state of operative machine, $y(t)$ is the state of the repairman and $z(t)$ is the oscillating state of market demand. Then the embedded generator of machine queue is as follows:

$$Q = \begin{matrix} \begin{matrix} (0,0,0) \\ (0,0,1) \\ (0,1,0) \\ (0,1,1) \\ (1,0,0) \\ (1,0,1) \\ (1,1,0) \\ (1,1,1) \\ (2,0,0) \\ (2,0,1) \\ (2,1,0) \\ (2,1,1) \end{matrix} & \begin{bmatrix} -(\sigma_1+d_1) & d_1 & \sigma_1 & & & & & & & & & \\ d_2 & -(\sigma_1+d_2) & & \sigma_1 & & & & & & & & \\ \sigma_2 & & -(\sigma_2+d_1+\gamma) & d_1 & & & \gamma & & & & & \\ & \sigma_2 & & d_2 & -(\sigma_2+d_2+\gamma) & & & & \gamma & & & \\ \zeta & & & & -(\sigma_1+d_1+\zeta) & d_1 & \sigma_1 & & & & & \\ & \zeta & & & d_2 & -(\sigma_1+d_2+\zeta) & & \sigma_1 & & & & \\ & & \zeta & & \sigma_2 & -(\sigma_2+d_1+\zeta+\gamma) & d_1 & & & \gamma & & \\ & & & \zeta & & \sigma_2 & d_2 & -(\sigma_2+d_2+\zeta+\gamma) & & & & \gamma \\ & & & & 2\zeta & & & -(\sigma_1+d_1+2\zeta) & d_1 & \sigma_1 & & \\ & & & & & 2\zeta & & d_2 & -(\sigma_1+d_2+2\zeta) & & \sigma_1 & \\ & & & & & & 2\zeta & \sigma_2 & & -(\sigma_2+d_1+2\zeta) & d_1 & \\ & & & & & & & & 2\zeta & \sigma_2 & d_2 & -(\sigma_2+d_2+2\zeta) \end{bmatrix} \end{matrix}$$

Here we show the complete matrix to demonstrate the problem of largeness even for a problem of this small size. From the above illustration it is clear that MMPP suffers the effect of “curse of dimensionality”, which is often encountered in CTMC modeling. However MMPP can model the behavior of individual repairman precisely. Thus it is more flexible. Similar approach can be applied to model individual machine. Here we use kronecker algebra to generate the above matrix. Without kronecker algebra the presentation of Q becomes more and more intractable as can be seen in this example.

Refer to fig. 4.1 our study of a tandem supply network governed by MTO policy treats all I_j 's therein as zeroes. Since no DC is necessary we treat the single transportation (distribution) activities as $M/M^*/m$. Under this concept we analyze a 2-echelon integrated system of a production echelon followed by a distribution echelon. Assume dual stages for production echelon and single stage for distribution echelon. Further assume the service rate at the three stages as $u_1 = u_2 = 2, u_3 = 1$. The other parameter settings are the same as outlined in the beginning of this section. Assume our budget limits the available machine

investment for each stage to be 5. The scenario of our employed example is a little different from a pure tandem queue and will be reiterated as follows for clarity purpose. The aggregated retailer faces a stationary (non-stationary) Poisson demand process with mean inter-arrival time ($1/\lambda$) of 1. The unreliable servicing process is as stated in section 4.3. After production at two consecutive stages, finished goods will be briefly inspected (inspection time is neglected). After inspection, any imperfect product has to go back to the first production stage for reworking. Assume that the feedback rate is constant with probability $\delta = 0.5$. Products passing inspection will immediately be transported to the assigned retailer on a FCFS basis whenever a vehicle is available. Assume there is always an infinite supply at the first stage when order comes. Also, for practical consideration we can think of the pure Poisson demand process as a unit-load, such as 1000 tons per order. Now we can apply the algorithmic approach as stated in section 4.2.

In step 0, the supply queueing network is just like an open Jackson network. Thus we can use Jackson's rule: $\lambda_i = \lambda_{i0} + \sum_j \gamma_{ji} \lambda_j$, where i, j means stage, $\lambda_{i,0}$ is external arrival and γ_{ji} is Markovian routing, to derive underlying traffic process. The traffic intensities at production echelon are all identical to $\rho = \lambda/u(1-\delta)$, where the denominator shows capacity loss when feedback is incorporated. The traffic intensities at distribution echelon remain unchanged, $\rho = \lambda/u$. From the traffic intensities calculated at each stage, it shows single server setting at each queueing system is not stable. Here multi-server setting is adequate. Now we can use the adjusted capacity, $u(1-\delta)$ in steps 1-9 to derive performance measures of interest. Tables 4.1-4.3 give the numerical results for decomposed single-queue under different uncertainty cases. Table 4.1 shows traffic intensity $\bar{\rho}$ of all possible resource combinations for single stage under different uncertainty cases. Accidentally all resource designs are stable. Table 4.2 lists $E[O]$ for each design under

different uncertainty cases. We do not list $E[RE]$ since it can be computed directly by using corollary 1. Also note since $r \leq m$ number of machines under repair will usually be no greater than total number of machines waiting for repair, Table 4.3 is the demand queue length for each design under different uncertainty cases. We can see when the machine resource is fixed, increasing operator resource causes $E[O]$ and $E[RE]$ to increase but causes $E[L]$ and $E[B]$ to decrease. This illustrates an idea that resource investment alone may not reduce the total cost as conjectured in section 4.3.

Table 4.1 Traffic intensity ρ' under different uncertainty cases

$r \setminus m$	Case 4.1				Case 4.2				Case 4.3				Case 4.4			
	2	3	4	5	2	3	4	5	2	3	4	5	2	3	4	5
1	0.554	0.373	0.283	0.229	0.576	0.389	0.295	0.240	0.554	0.373	0.283	0.229	0.576	0.389	0.295	0.240
2	0.550	0.366	0.275	0.220	0.552	0.368	0.277	0.222	0.550	0.366	0.275	0.220	0.552	0.368	0.277	0.222
3		0.366	0.275	0.220		0.366	0.275	0.220		0.366	0.275	0.220		0.366	0.275	0.220
4			0.275	0.220			0.275	0.220			0.275	0.220			0.275	0.220
5				0.220				0.220				0.220				0.220

Table 4.2 $E[O]$ for each design under different uncertainty cases

(m, r)	Case 4.1	Case 4.2	Case 4.3	Case 4.4
(2, 1)	1.803	1.734	1.803	1.734
(2, 2)	1.818	1.811	1.818	1.811
(3, 1)	2.679	2.570	2.679	2.570
(3, 2)	2.726	2.712	2.726	2.712
(3, 3)	2.727	2.725	2.727	2.725
(4, 1)	3.533	3.380	3.533	3.380
(4, 2)	3.632	3.607	3.632	3.607
(4, 3)	3.636	3.632	3.636	3.632
(4, 4)	3.636	3.635	3.636	3.635
(5, 1)	4.360	4.158	4.360	4.158
(5, 2)	4.535	4.496	4.535	4.496
(5, 3)	4.545	4.538	4.545	4.538
(5, 4)	4.545	4.544	4.545	4.544
(5, 5)	4.545	4.545	4.545	4.545

Table 4.3 Demand queue length for each design under different uncertainty cases

(m, r)	Case 4.1	Case 4.2	Case 4.3	Case 4.4
(2, 1)	1.609	1.885	2.791	3.587
(2, 2)	1.568	1.586	2.672	2.725
(3, 1)	1.131	1.258	1.268	1.457
(3, 2)	1.102	1.111	1.225	1.238
(3, 3)	1.102	1.103	1.224	1.225
(4, 1)	1.043	1.128	1.083	1.196
(4, 2)	1.024	1.029	1.055	1.063
(4, 3)	1.023	1.023	1.054	1.055
(4, 4)	1.023	1.023	1.054	1.054
(5, 1)	1.018	1.082	1.034	1.116
(5, 2)	1.006	1.009	1.016	1.020
(5, 3)	1.005	1.006	1.014	1.015
(5, 4)	1.005	1.005	1.014	1.014
(5, 5)	1.005	1.005	1.014	1.014

In Neuts and Lucanton (1979), the author designed variant customer arrival rates according to the number of operative servers. The author observed that π is not affected by the arrival rate but was affected by the number of repairman. Actually this can be easily comprehended since π is the stationary probability vector for Q and Q is nothing to do (independent) with arrival rate. In cases 4.3 and 4.4, we assume there are alternate arrival (market demand) rates for all possible states of operative servers and we obtain the same results. Therefore cases 4.1 and 4.3 have the same stationary probability vectors and so do cases 4.2 and 4.4. This fact also causes cases 4.1 and 4.3 to have the same number of operative machines and so do cases 4.2 and 4.4 as identified in table 4.2. This also explains why ρ' for cases 4.1 and 4.3 are the same and so are ρ' for cases 4.2 and 4.4 as identified in table 4.1 since the effective arrival rate for all cases are all the same. Generally speaking, the average queue length of cases 4.2, 4.3 or 4.4 is worse than 4.1 with case 4.4 being the worst. This explains the fact that case 4.4 is the most uncertain.

Next we used (4.3) to search for the optimal resource design, which is the work of step 10. Since the effective arrival rate is fixed at unity, expected backorder level can be

thought of as average response time from Little's formula: $E[B] = E[L] = \lambda E[W] = E[W]$. And the last cost component of (4.3) can be thought of as the cost of customer waiting time. Usually the objective set by production and distribution departments are different. The former often cares about operation costs including production, repairing, and inventory holding. Except for operation costs such as transportation, repairing, inventory holding, the latter also has to pay penalty costs owing to delayed transportation or waiting-related complaints from customers. We can therefore write down different optimization models for different contributors as well as for the whole supply system as follows. Denote i as stage number.

$$\text{(Production) } \min TC_p = \sum_{i=1}^2 \{c_h E[L_i] + c_o E[O_i] + c_r E[RE_i]\}$$

$$\text{(Distribution) } \min TC_d = c_h E[L_3] + c_o E[O_3] + c_r E[RE_3] + c_b \sum_{i=1}^3 E[L_i], \quad (4.5)$$

$$\text{(System) } \min TC_s = TC_p + TC_d$$

Note here we count the queue length at the first stage. Local optima for individual contributors and global optima for the whole system from searching on (4.5) are shown in table 4.4. Notice here we add a reliable supply system (case 4.0), which is a canonical $M/M/m$ tandem QN under the same parameter setting for comparison purpose. Since case 4.0 is a reliable SC, there is no repairman and incurred repair cost.

Table 4.4 Local and global optimal solutions under different uncertainties

	Production			Distribution			System				
	Design	Design	E[TC _p]	Design	Design	E[TC _d]	Design	Design	E[TC _s]		
Case 4.0	(2, $_$)	(2, $_$)	5.333	(5, $_$)	(5, $_$)	(4, $_$)	184.983	(4, $_$)	(4, $_$)	(4, $_$)	194.589
Case 4.1	(2, 1)	(2, 1)	12.427	(5, 3)	(5, 3)	(4, 3)	193.399	(4, 3)	(4, 3)	(4, 3)	218.398
Case 4.2	(2, 1)	(2, 1)	12.289	(5, 4)	(5, 4)	(4, 3)	193.387	(4, 3)	(4, 3)	(4, 3)	218.362
Case 4.3	(2, 2)	(2, 2)	13.58	(5, 3)	(5, 3)	(4, 3)	196.355	(4, 3)	(4, 3)	(4, 3)	224.025
Case 4.4	(2, 2)	(2, 2)	13.591	(5, 4)	(5, 4)	(4, 4)	196.352	(4, 4)	(4, 4)	(4, 4)	224.016

In an SC analysis, conflicts usually exist between contributors owing to different objective settings as shown in (4.5). Table 4.4 reveals this fact. The optima for each contributor are not the same. For example, in case 4.1, the distributor has to pay high penalty cost of delayed delivery, therefore, he would prefer the waiting time at the upstream supply, the production, to be less. Since more resource, in this case, design (5, 3) yields the least waiting, the distributor would like the producer to invest more resource. On the contrary, the producer does not directly respond to customers, the less resource investment is enough for him to balance the cost. The optima of the whole supply system are just the compromise solutions, which yield the most beneficial results for the integrated system. Other interesting findings with respect to this specific application problem have been observed. First, multiple solutions exist. For example, in case 4.1, distributor will probably prefer to choose the design of (4, 3) instead of (4, 4) though both selections will yield the same performance measures as can be told from tables 4.2 and 4.3. However, if operator-hiring cost is included, (4, 3) will be better than (4, 4) practically. That's the reason why we put the less resource as the optimum. Second, it happens that the optimal number of servers selected by respective contributors under all cases is the same, while the repairman capacity is not. Lastly, we can tell the degree of impact of an unreliable SC when it is compared to a reliable one under different cases. For example, queue lengths and optimal costs of cases 4.3 and 4.4 are greater (worse) than cases 4.1 and 4.2 in local and global analyses when compared to case 4.0. We can also identify which factor causes the impact most. Since the optimal costs for case 4.1 and 4.2 are very close and so are them for cases 4.3 and 4.4, it seems the impact of repairman uncertainty is less significant than the uncertainty of market fluctuation under the cost structure specified in this example.

To test if the output performance measures of the studied QBD model actually deliver “exact” solutions, we conducted a full-scaled CTMC model for a 2-stage tandem QN. Assume the available machines at stage 1 and 2 are either 2 or 3. We then solve the whole

CTMC through the support of SPNP, a GSPN tool. Since SPNP can only solve finite state Markov chain, such as $M/M/m/K$, we used an approximate model by assigning a very big buffering capacity, i.e. $K-m$ for each input queue. Table 4.5 shows the comparison between decomposition method and GSPN (cf. table 4.3). It seems our decomposition method and the GSPN-generated outputs are very close. Since the GSPN-generated outputs are just approximation of the original infinite queue. The quality of the decomposition methods is verified. Notice in table 4.5 there are several unavailable values in case 4.4, which are owing to the explosion of state space and we will discuss this later in section 4.5.

Table 4.5 Approximation results obtained from GSPN for different uncertainty cases

$(m1, r1)^a$	$(m2, r2)^b$	Case 4.1		Case 4.2		Case 4.3		Case 4.4	
(2, 1)	(3, 1)	(1.609 ^c , .135 ^d)	0.2 ^e	(1.885, 1.276)	0.7	(2.678, 1.263)	2.2	(3.502, 1.456)	1.2
(2, 1)	(3, 2)	(1.609, 1.106)	0.2	(1.878, 1.135)	1.3	(2.678, 1.263)	3.6	N/A	N/A
(2, 1)	(3, 3)	(1.609, 1.106)	0.2	(1.878, 1.119)	0.9	(2.678, 1.219)	2.2	N/A	N/A
(2, 2)	(3, 1)	(1.568, 1.133)	0.2	(1.608, 1.262)	0.9	(2.581, 1.261)	2.0	N/A	N/A
(2, 2)	(3, 2)	(1.568, 1.105)	0.1	(1.560, 1.118)	1.1	(2.581, 1.218)	2.0	N/A	N/A
(2, 2)	(3, 3)	(1.568, 1.104)	0.1	(1.560, 1.104)	0.9	(2.581, 1.217)	2.0	N/A	N/A
(3, 1)	(2, 1)	(1.131, 1.615)	0.1	(1.257, 1.921)	1.0	(1.298, 2.750)	1.9	(1.458, 3.661)	1.1
(3, 1)	(2, 2)	(1.131, 1.574)	0.2	(1.257, 1.649)	2.0	(1.246, 2.633)	1.6	N/A	N/A
(3, 2)	(2, 1)	(1.102, 1.611)	0.2	(1.113, 1.885)	0.1	(1.253, 2.742)	2.0	N/A	N/A
(3, 2)	(2, 2)	(1.102, 1.570)	0.1	(1.114, 1.564)	0.8	(1.247, 2.633)	1.6	N/A	N/A
(3, 3)	(2, 1)	(1.102, 1.611)	0.1	(1.105, 1.860)	0.8	(1.252, 2.742)	2.0	N/A	N/A
(3, 3)	(2, 2)	(1.102, 1.570)	0.1	(1.101, 1.562)	0.8	(1.246, 2.633)	1.6	N/A	N/A

Note: a, b: resource design at stage 1, 2. c, d: queue length at stage 1, 2. e: percentage mean absolute deviation = $\{ \text{abs}(c - \text{exact } 1) / \text{exact } 1 + \text{abs}(d - \text{exact } 2) / \text{exact } 2 \} / 2 \times 100$, exact 1, 2: queue length at stage 1, 2 obtained from decomposition. N/A: not available.

As for more involved generalized Jackson network with arbitrary Markovian routing, it seems we can apply the same procedure as stated above. For a make-to-stock (MTS) tandem queue, where some of I_j 's may not be zeroes, the approximation model of L & Z for analyzing a tandem QN may be suitable here. The queue length obtained at each queueing subsystem should be able to serve as the input to the model of L & Z. However,

the accuracy of this approach is verified only under specific conditions as revealed in chapter 3. As a final remark, our problem under study is an $M/M^*/m$ for cases 4.1 and 4.2 and $MMPP/M^*/m$ for cases 4.3 and 4.4. Alternatively, we can treat service process as phase type and the queueing system becomes $M/PH/m$ and $MMPP/PH/m$ for cases 4.1 and 4.2 and for cases 4.3 and 4.4 respectively.

4.5 Computation issue

The major flaw of applying MMPP modeling is that the stationary probability vector becomes intractable when states increase. However if it's known that the repairmen are identical, which means if σ_{j1} and σ_{j2} are the same, we can reduce the states by making suitable arrangement. In this case the states no longer stand for the state of each individual repairman but for available repairmen. The difference between MMPP and the proposed states reduction technique is $(\text{number of machine} + 1) \times (2^{\text{number of repairman}} - (\text{number of repairman} + 1))$. The difference becomes large if the number of repairman is high. However, MMPP is more flexible, for example, it allows modeling non-identical server/repairmen.

In the above computation procedure, we use successive substitution method to obtain, R and x_i , $i \leq m - 1$ for steps 6 and 7 respectively. Alternatively one may investigate other solution algorithm such as Block Gauss Seidel (BGS) or Gaussian elimination as is usually adopted in solving finite CTMC. However, this is beyond our scope of study. In this study we set all the convergence criteria to be within 0.000001 for finding R and 0.001 for finding x_i , $i \leq m - 1$. We used $R = 0$, $x_0 = 0$, $x_i = 1$, $1 \leq i \leq m - 1$ as starting guess solutions. All the numerical experiments are implemented on a Pentium IV 2.0 GHz PC. The average CPU time ranges from several seconds to several minutes for a single (m, r) problem to converge, depending on the size of the states. Successive substitution as is used in this study seems to converge in acceptable time. In computing the values for table 4.5 we also

encounter the problem of convergence. For example in case 4, the states generated by GSPN for problem set (3, 1), (2, 1) is over one million (1278400) and the transitions generated is even more (amounts to 8888374). It causes nearly one hour to find the stationary probabilities. The exhaustive CPU operations plus not enough memory make the other trials fail.

4.6 Concluding remarks

Resource design of supply systems has to reflect the uncertain environment to lower the cost. In this chapter we have shown how to solve such problems by using an algorithmic approach. We assumed major uncertainties are from supply and/or demand. This chapter provides an analytically tractable framework for “designing” such supply system. In this chapter we assume the SC adopts MTO policy. Under this assumption QN can be employed to investigate system dynamics. Like most of the other analytic models for analyzing a QN we used the same decomposition approach. The complex system dynamic of the studied QN is analyzed by decomposing it into several isolated queueing subsystems. The linkage of subsystems is then straightforward by solving the traffic (arrival) process into each subsystem and transforming it into equivalent server capacity. The MMPP models can then be developed for different uncertainty scenarios to investigate impacts of uncertainties on system behavior for each subsystem. The performance measures are then obtained from solving the developed QBD model. Finally, simple search on pre-specified objective function resolves the resource design problem. Specifically we used this algorithmic approach to solve optimal resource design problem for a realistic multi-echelon supply system. The objective is to find the trade-off cost for different contributors and compromise solution for the whole SC. For the verification of the quality of the decomposition approach we used finite-state-based GSPN model to approximate the original infinite state CTMC and obtain satisfactory results.

Existing literature of stochastic supply systems usually assumes single-server and neglect unreliable supply from upstream. Our model relaxes the usually adopted single-server assumption and investigates system behavior under different uncertainty cases, which should be more practical than other studies. Though the results are not satisfactory for MTS case. It works well under MTO mode as shown in this chapter. We also proved that the average number of operative machines is equal (proportional) to the average number of machines under repair when mean time to failure and mean time to repair are the same (proportional) by using a matrix algebraic approach. This property is not explicitly related to repairman.

We also showed the degree of impact of an unreliable SC when it was compared to a reliable one under different cases. We showed how to use the proposed method to identify the most significant uncertainty factor, which may cause the most serious impact under specific cost structure. The decision-maker of an SC may use this information to improve the uncertain input as studied herein in a most efficient way. Finally, this study shows the algorithmic approach for solving QBD-decomposed model for more realistic and complex stochastic SC problem is feasible.

Chapter 5 Optimal buffer and capacity design under service constraint

5.1 Introduction

One of the most urgent issues in SCM is that there is still lacking integrated analytic model of an SC under uncertainty consideration. The uncertainty may come from supply and/or demand randomness, stochastic delay from upstream supply among others. Performance measures of a successful SC are usually described as minimum operating costs, maximum selling benefits, acceptable customer waiting time, flexibility etc. How do we model the uncertain interaction between/among SC distributors such that the goal(s) of SCM can be achieved in the long run? Many critical success factors (CSF) contribute to the successful operation of an SC. Well-organized plan(s) such as flawless supply, supply quality, adequate stock level and/or service rate are no doubt the CSF to successful operation of an SC. (Note in this chapter we treat service rate as capacity). With today's complex topology of an SC and its inherent sophisticated system dynamic, the modeling of the SC with uncertainty concern is becoming a challenge. In the past, most researches related to stochastic modeling of an SC concentrated on the evaluation models. Accurate and efficient evaluation models have been reported to approximate the system dynamic of an SC. Among those works, most focus on independent models. The reason why few integrated models can be found in the literature may be that exact closed-form solutions only exist in the simplest systems. For a more complex SC, closed-form solutions are usually unknown and therefore approximation or simulation is the only viable approach. However, with the evolutions of analytic models for SCM, integrated analytic models of SC have been addressed more often than before. Moreover, the studies related to the optimization of an SC are still very few even nowadays. The optimization problems may include: how to find adequate stocking and capacity levels such that system performance is

maximized and so on. Some researches related to buffer and/or capacity optimizations have been reported. However, due to the complex topology of a real SC, most explore the tandem form.

Buffer allocation problem (BAP) is the well-known topic in production system. BAP belongs to the combinatorial and discrete integer optimization problem. The problem relates to where and how much to stock at each stage such that the averaged total system-operating cost is minimized per unit time. This problem deems buffer as the only decision variable to optimize the system performance. Usually the operating cost includes holding and other penalty cost, which can include backorder or lost cost depending on operation policy of the decision maker. On the other hand, capacity design, or capacity allocation problem (CAP) seems to be not so popular as BAP. Though, its importance cannot be neglected. This problem relates to work load sequence. Since we may express capacity as a continuous variable, the analysis of how this factor may impact the performance of an SC is even more difficult. In this chapter, we investigate how system performance may be influenced by these two control variables independently or simultaneously. We demonstrate that through careful deployment of these parameters system performance may be greatly enhanced.

MTS is usually referred to as planned-inventories at the output buffers of intermediate stages along the SC to fight against random demand needs. For MTO mode, no output buffer is designed and the buffer is just the “rooms” of waiting spaces in front of each servicing facility when the server is busy. The “rooms” may be infinite or finite depending on costs and/or space limitation. To facilitate our study, in this chapter we give MTO another meaning that only the end stage of an SC can hold inventory while all the other stages cannot. Since an SC may possess several supply functions, the traditional production modes such as MTS, and MTO are used herein for the SCM context. Further, we notice the unrealistic assumption of unlimited supply in classical multi-echelon inventory control

models for distribution systems. We therefore put our efforts on analyzing the congested nature of an SC.

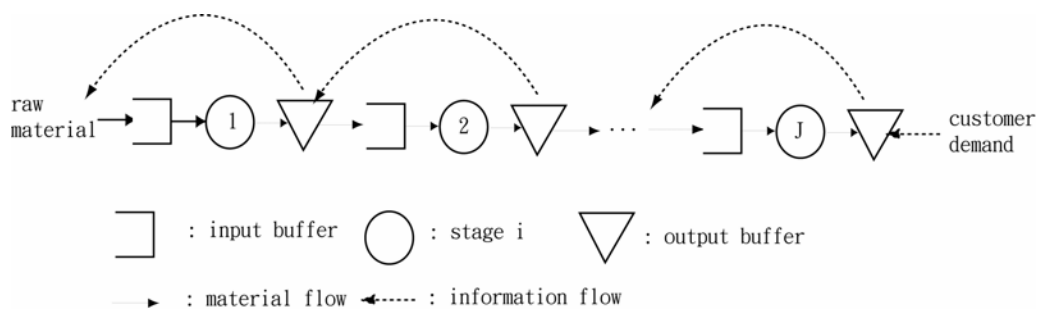
Our goals in this chapter are two-fold. First, we want to find the most suitable optimization methods that can find the optima of an SC. Especially, we investigate the state-of-the-art meta-heuristic search methods on the studied problem. Second, we investigate if there is other modeling approach that can assist in transforming a more general SC into a tandem form to ease the analysis work. Here “general” refers to both topology and system dynamic of an SC. Specifically we want to find the optimal stocking and/or capacity level(s) at each stage of an SC such that the pre-specified customer service level at the ending stage is satisfied. From the perspective of the queueing theory, the whole SC can be viewed as an arbitrary configured QN. The analysis of such supply network with planned inventories at each stage is difficult due to the intractable interaction between/among stages (see chapter 3 for more details). For a tandem and small-scaled problem involving only few stages, the classic approaches such as enumerative search or derivative-based method is usually enough to solve the problem. For a larger and more general SC, alternative optimization measures need to be sought after. In this chapter we wish to provide adequate solution models for different topology and difficulty levels of system dynamics. Additionally, we want to find if there is any rules-of-thumb for resource expansion planning. Herein we treat buffer and capacity as our resource under study. Finally, we explored the impacts of upstream unavailability and imperfect quality. To our knowledge, the robust concern in SC optimization is still not much in the literature. The findings and observations from empirical studies may provide valuable managerial insights in strategic planning of an SC.

The basic assumptions are the same as before. Additionally, we assume there is more than one type of product. However, common production process proceeds until later distribution stage. Thereinafter, product differentiation may continue. Since this study is

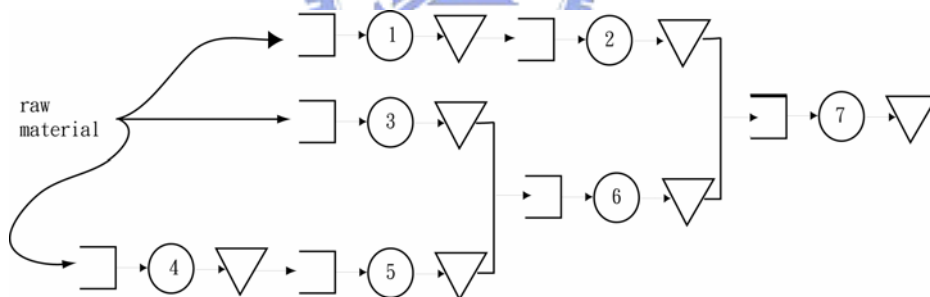
stochastic-based, performance values are obtained as expected value in steady state and are used for objective values of evaluation functions for optimization processes.

5.2 Methodology

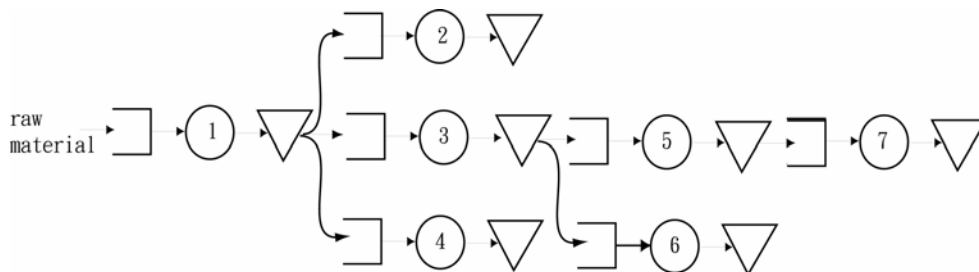
Basically we can decompose three basic topology forms relating to a congested SC: tandem, assembly, and distribution as shown in fig. 5.1. All belong to dual-buffer design with both input and output buffers, which is specifically tailored for processing network with planned inventories. We use base-stock inventory control as studied herein.



a A canonical tandem model with base stock control



b An assembly model



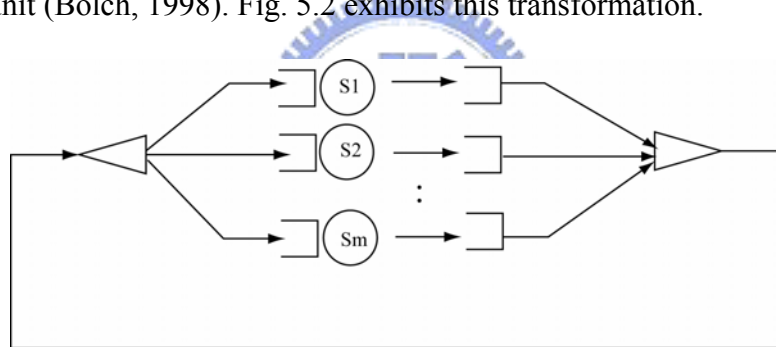
c A distribution model

Fig. 5.1 Three basic congested supply systems

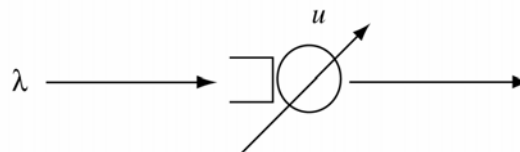
Detailed operation of base-stock control can be referred in chapter 3. Note bold and italic font type refers to vector or matrix. Italic font type refers to variables and normal font type refers to constants. We assume demand arrival rate to be Poisson process and all the other rates to be exponential distributed.

5.2.1 Flow Equivalent System

For using matrix analytic model, especially QBD decomposition to modeling the above tandem and distribution systems as stochastic processes, please refer to chapter 3. For the assembly system with multiple parallel suppliers (processors), we formulate each processor as an $M/M/1$ queueing system with identical processing rate. We thus can treat it as a fork-join system and transform it into a Flow Equivalent System (FES) with single processing unit (Bolch, 1998). Fig. 5.2 exhibits this transformation.



a An assembly system with m suppliers



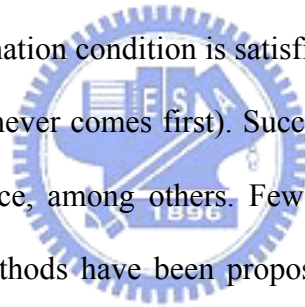
b An equivalent FES

Fig. 5.2 Transformation from an assembly system into an $M/M/1$ FES.

5.2.2 Meta-heuristic methods

Genetic algorithm (GA) is a parallel processing algorithm. It mimics the natural evolution process of species through crossover and mutation operators to get better

“fitness” values generation by generation. Variants of GA use different selection procedures to propagate its offspring such as Rowlett wheel selection (RWS), ranking, etc. there are also variants of GA operators such as arithmetic crossover, directional crossover, non-uniform mutation etc. to increase the performance of genetic algorithm in terms of convergence and accuracy. To get rid of early convergence or local optima, these variant operators have reported successful instances (Michalewicz, 1999). In this chapter we use some of these variant operators to enhance the performance of GA. The algorithm begins at generating initial population of chromosomes. Each chromosome is the representative of decision variable(s) of interest, in the form of real, binary, or gray-code. After the evaluation of the population, GA enters into generation cycles of selection and crossover (recombination) and mutation. Some selection procedures use elitism and the GA cycles ends when pre-specified termination condition is satisfied (usually in terms of convergence or maximum run length, whichever comes first). Successful GA applications were mostly reported in engineering, science, among others. Few were reported in the SC area. To handle constraints, several methods have been proposed such as variable(s) elimination, penalty addition, infeasible points repairing, decoder method etc. In this chapter, we select decoder method as our constraint handling routine for *SL* requirement. Decoder method for continuous variable optimization has been reported (Koziel and Michalewicz, 1998). Simulated Annealing (SA) is a random search procedure. Different from GA, SA selects its next search point in the neighborhood of the previous one. SA tries to stabilize its searching process through adjusting the so-called “cooling temperature”. This method originates from metallurgy crystallization process. For minimization problem such as ours, downhill movement is always accepted. However, uphill movement is accepted if Metropolis criterion is satisfied. Metropolis criterion is based on the “energy” (value of evaluation function) deviation between new and old points and has larger probability in the earlier stage of temperature setting. Unlike GA, which has several alternatives to handle



constraint(s), SA only searches the feasible directions. To hasten its convergence performance, Goffe and Rogers (1994) developed a method based on dynamic step length adjustment of movement as follows.

$$VM_i^{new} = \begin{cases} VM_i^{old} \times (1 + \frac{C(R_i - U)}{L}), & \text{if } R_i > U \\ VM_i^{old} / (1 + \frac{C(L - R_i)}{L}), & \text{if } R_i < L \\ VM_i^{old}, & \text{otherwise,} \end{cases}$$

where VM represents dynamic step length, R_i is the ratio of accepted moves for direction (variable) i , U and L is the pre-specified parameters for upper and lower ratios respectively, both constants. C is a pre-specified parameter, a constant. The spirit of VM is to broaden the step length for more promising direction and shrink the step length for less promising direction. Empirical studies show this adjustment converge very well for our problems. We translate the original C++ code of constrained GA obtained from public domain into Matlab version and modify the existing SA code obtained from public domain by adding SL requirement. The Matlab platform is chosen for its excellence in handling matrix operations. These operations frequently occur in our model. Appendix C lists all the codes.

5.3 Optimization

In the following exposition, we assume the performance measures of interests are TC and SL . The decision variables are buffer and capacity. TC include intermediate inventory and end stage inventory holding costs, backorder costs, and may include operation cost when needed.

5.3.1 Tandem topology

For tandem topology of an SC, closed-form solutions only exist in special cases. Refer to a. of fig. 5.1, Let S_j denote output buffer of stage j , $1 \leq j \leq J$. First we focus on BAP, the following procedures find the optimal buffers, S^* for two special cases.

Case 5.1 $S_1 = S, S_j = 0, 1 < j \leq J$.

For this case, only the first stage has buffer. Since the ending buffer is zero, no SL consideration is needed and hence is an unconstrained optimization problem. Suppose we focus our attention on inventory holding and end stage backorder costs only, neglecting any other incurred costs during operation. We show how to derive the general form of this problem starting from a 2-stage setting. We use L & Z to get performance measures of interests and then take partial derivative of them as shown in the following derivative-based method:

$$r_1 = 1, c_1 = [-v_1] = [-(u_1 - \lambda)], p_1 = \lambda(\lambda I_1 - c_1)^{-1} = \frac{\lambda}{u_1} = \rho_1, \pi_1 = r_1 p_1 = \rho_1.$$

We get the average backorder level at stage 1

$$E[B_1] = \pi_1 p_1^S (I_1 - P_1)^{-1} \mathbf{1} = \frac{\rho_1^{S+1}}{1 - \rho_1}, E[I_1] = S - \pi_1 (I_1 - P_1)^{-1} \mathbf{1} + E[B_1] = S - \frac{\rho_1}{1 - \rho_1} + \frac{\rho_1^{S+1}}{1 - \rho_1}.$$

Similarly

$$r_2 = [r_1 p_1^S, 1 - r_1 p_1^S \mathbf{1}] = [\rho_1^S, 1 - \rho_1^S], c_2 = \begin{bmatrix} -v_1 & v_1 \\ 0 & -v_2 \end{bmatrix} = \begin{bmatrix} -(u_1 - \lambda) & (u_1 - \lambda) \\ 0 & -(u_2 - \lambda) \end{bmatrix},$$

$$p_2 = \lambda(\lambda I_2 - c_2)^{-1} = \begin{bmatrix} \rho_1 & \rho_2(1 - \rho_1) \\ 0 & \rho_2 \end{bmatrix}, \pi_2 = r_2 p_2 = [\rho_1^{S+1} \rho_2(1 - \rho_1^{S+1})].$$

$$E[B_2] = \pi_2 p_2^0 (I_2 - P_2)^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = [\rho_1^{S+1} \rho_2(1 - \rho_1^{S+1})] \begin{bmatrix} \frac{1}{1 - \rho_1} & \frac{\rho_2}{1 - \rho_2} \\ 0 & \frac{1}{1 - \rho_2} \end{bmatrix} = \frac{\rho_1^{S+1}}{1 - \rho_1} + \frac{\rho_2}{1 - \rho_2}.$$

Continue this way, we can find general form for a J -stage system.

$$E[B_J] = \frac{\rho_1^{S+1}}{1 - \rho_1} + \sum_{j=2}^J \frac{\rho_j}{1 - \rho_j}.$$

$$E[I_1] = S - \frac{\rho_1}{1 - \rho_1} + \frac{\rho_1^{S+1}}{1 - \rho_1}, E[I_j] = 0, 2 \leq j \leq J.$$

Therefore, we can write the optimization function as follows:

$$\begin{aligned} \text{Min } TC(S) &= c_b E[B_J] + h_{wip} E[I_1] \\ &= c_b \left(\frac{\rho_1^{S+1}}{1-\rho_1} + \sum_{j=2}^J \frac{\rho_j}{1-\rho_j} \right) + h_{wip} \left(S - \frac{\rho_1}{1-\rho_1} + \frac{\rho_1^{S+1}}{1-\rho_1} \right). \end{aligned} \quad (5.1)$$

Take partial derivative of TC with respect to S and let it be 0, we get

$$\begin{aligned} \frac{\partial TC}{\partial S} &= 0, \\ \frac{b\rho_1^{S+1} \ln(\rho_1)}{1-\rho_1} + h_{wip} + \frac{h_{wip}\rho_1^{S+1} \ln(\rho_1)}{1-\rho_1} &= 0, \end{aligned}$$

Note $\frac{\partial a^{f(x)}}{\partial x} = \frac{\partial f(x)}{\partial x} a^{f(x)} \ln a$. Reorganize terms, we get

$$\rho_1^{S+1} = \frac{-h_{wip}(1-\rho_1)}{(b+h_{wip}) \ln(\rho_1)}.$$

Finally, we get the closed-form solution of the optimal buffer S^* for this special case:

$$S^* = \ln\left(\frac{-h_{wip}(1-\rho_1)}{(b+h_{wip}) \ln(\rho_1)}\right) / \ln(\rho_1) - 1. \quad (5.2)$$

Note since $\frac{\partial^2 TC}{\partial S^2} = \frac{b\rho_1^{S+1} [\ln(\rho_1)]^2}{1-\rho_1} + \frac{h_{wip}\rho_1^{S+1} [\ln(\rho_1)]^2}{1-\rho_1} > 0, \because \rho_j < 1$, S^* is indeed a

minimum. Also S^* only relates to cost parameters h_{wip} and c_b , and ρ_1 , not to other stages.

For non-integer solution we have to compare the TC of its nearest integer to get S^* . We show this by an example as follows.

If the parameter is $c_b = 10$, $h_{wip} = 1$, and $\rho_1 = 0.5$. Plugging in (5.2) we immediately get $S^* = 2.9$. Since the optimal solution is not integer, from (5.1) we compare $TC(2) = 4.75 + a$ and $TC(3) = 4.375 + a$, where a is a constant. Since $TC(3) < TC(2)$. $S^* = 3$.

Case 5.2 $S_j = 0, 1 \leq j \leq J-1, S_J = S$.

For this case, only end stage has buffer and we may also want to handle SL . Closed-form solution seems not so easy to derive by using derivative-based method. Here

we proposed an easier solution method instead. Since all the upstream stages use MTO, there is no expected inventory except for the end stage. Suppose SL is not greater than some pre-specified level. From the property of CPH we have $SL = \Pr\{T > t\} = r_j p_j^S e^{Ct} \mathbf{1} \leq \beta$, where C stands for lead-time distribution of CPH type.

We get the constraint optimization problem as below.

$$\text{Min } TC(S) = c_b \pi_j p_j^S (I_j - p_j)^{-1} \mathbf{1} + h_{wip} (S - \pi_j (I_j - p_j)^{-1} \mathbf{1} + \pi_j p_j^S (I_j - p_j)^{-1} \mathbf{1}).$$

S.T.

$$r_j p_j^S \mathbf{1} \leq \beta.$$

In the above constraint we assume $t = 0$. Intuitively, $E[B]$ decreases in S , and $E[I]$ increases in S . We can show TC is convex-like, and we should always find S^* theoretically. It's also known SL decrease in S (the more stock, the more protection). Recognizing these we make the following rule based on convexity property of the studied problem.

Let $S' = \min\{S, r_j p_j^S \mathbf{1} \leq \beta\}$,

If $S^* > S'$, select S^* ,

Else select S' .



Again, we illustrate how it works through an example.

Suppose we have a four-stage tandem SC. Parameter settings are as follows. $\lambda = 1$, $u_1 = u_2 = 1.25$, $u_3 = u_4 = 2.5$, $\mathbf{h}_{wip} = [0.5, 0.5, 0.5, 0.5]$ (note both holding costs at input and output buffers of all intermediate stages are all the same), $c_h = 1$, $c_b = 10$. Starting from $S = 0$, we calculate $E[B]$, $E[I]$, and incurred TC . The results are graphed in fig. 5.3. From fig. 5.3, we see that $S^* = (0, 0, 0, 25)$, with $TC = 23.628$, $SL = 0.1$. Now, suppose $\beta = 0.05$, applying the decision rule, we can immediately find $S' = (0, 0, 0, 31)$, with $TC = 25.192$, $SL = 0.049$. Since $S^* < S'$, we select S' as the optimum. If $\beta = 0.15$, we can immediately find $S' = (0, 0, 0, 22)$, with $TC = 24.361$, $SL = 0.142$. Since $S^* > S'$, we select S^* as the optimum.

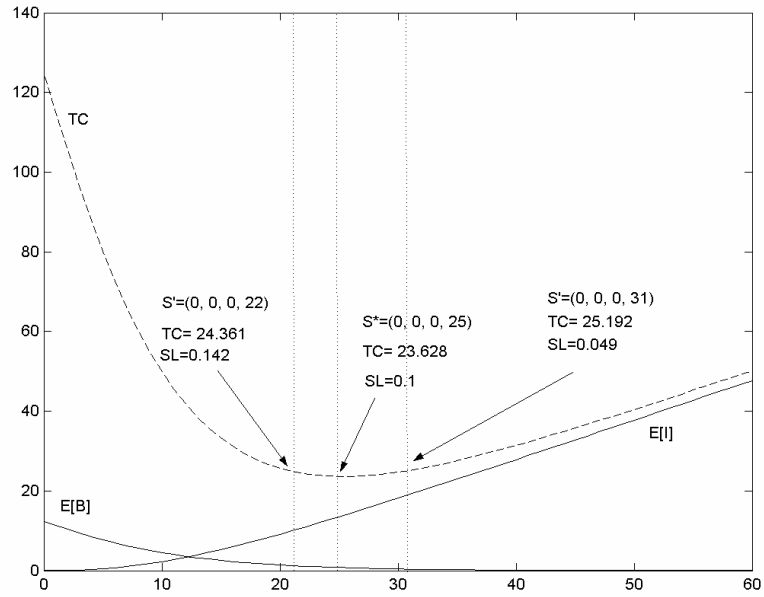


Fig. 5.3 Using convexity-based method to search a tandem SC.

For a general case, the intermediate planned inventories do not have to be zeros. Traditional methods like derivative or convex-based simple rule will not work. However, enumerative method may still apply. To demonstrate how it works and what it cannot, we propose a model, which aims at a broader SC context. We explore the alternative measures when enumerative method also fails. Fig. 3.2 (without feedback) of chapter 3 acts as such test bed. We illustrate how to solve it as follows.

Case 5.3 $S_j \geq 0, 1 \leq j \leq J$.

The scenario of our fabricated SC is similar to chapter 3 with the exception that all the processing will never fail and the quality is 100% perfect. The mathematical model of our optimization problem is listed below.

$$\text{Min } TC(\mathbf{S}, \mathbf{u}) = \mathbf{h}_{wip} \mathbf{WIP} + c_h E[I] + c_b E[B] + \sum_{j=1}^J c_j u_j$$

S.T.

$$u_j \geq \underline{u}_j,$$

$$SL \leq \beta,$$

$$S_j \in \text{Integer}^+ \cup \{0\}, u_j \in \text{Real}^+.$$

For BAP we fix \mathbf{u} to be some constants, as a result the last term of the TC function is a constant. For CAP we fix \mathbf{S} to be some constants. The main purpose of adding the capacity incurred cost is that by adding penalty to resource expansion we can restrict the search space of \mathbf{u} in more reasonable range. Besides, not like the buffer incurred cost, which only occurs in the beginning of the evaluation horizon, capacity incurred cost is forever. The setting seems reasonable. For problems related to BAP, we fix $u_j = 1.25$ for each stage. We set up cost structure as follows. This cost structure will be used throughout the remaining of this chapter. We let $\mathbf{h}_{wip} = (0.5, 0.5 + x, 0.5 + 2x, \dots)$ and $\mathbf{c}_h = (0.5, 0.5y, 0.5y^2, \dots)$ where $x = 0.1$ and $y = 1.1$. The arrangement makes the cost of \mathbf{WIP} to be linearly increasing while that of inventory to be geometrically increasing. We set $c_j = 1, \lambda = 1, \beta = 0.1$. For BAP, we solve with enumerative method and compare the results with meta-heuristic methods. Below are our proposed algorithms for enumerative-based and meta-heuristic based methods.

The enumerative optimization procedure (we call it ENU hereinafter) in Algorithm 5.1 is borrowed from Boyaci & Gullego (2001). Though their procedure assumes constant supply lead-times we find it should work for our problem as well. Starting from $(s_T, s, 1)$, Procedure (s_T, s, k) recursively calls itself and enumerating all feasible solutions. Note $C(s_T)$ represents the best cost therein.

As introduced earlier, meta-heuristic approach such as GA and SA are the modern global optimization techniques, which are suitable for highly nonlinear, derivative-free problem. If capacity is the decision variable, the enumerative search fails. Here we try to

solve the same BAP by pre-testing the power of meta-heuristic methods. Later we show how to extend its capability of handling more complex problem like CAP. As Michalewicz (1999) reveals, data structure plus genetic algorithm constitute evolution programs. For the problem under study, two approaches may be used to represent data structure: float and binary coding. Using binary coding is natural for this integer-programming problem. Alternatively we use float representation due to the dimension of the decoded hypercube, which will be used as constraint handling routine. For GA operator of crossover, we used both simple and arithmetic crossover. Simple crossover is well known. The latter guarantees any crossover happens within convex limits of decision variables. After the crossover, we let the whole population undergo non-uniform mutation. As mutation is known as background operator, the purpose for non-uniform mutation is to provide higher diversification in the earlier stage of the mutation operation while speed up convergence in later generations. We used simple ranking as our selection scheme to decide which offspring to put into the mating pool. Elitism is also implemented to keep better chromosome to propagate to the next generation. As mentioned earlier, there are some constraint-handling routines. Some penalty methods try to find the most suitable penalty ratio to hasten their convergence speeds. Usually many runs with many penalty ratios have to be tested. Though the penalty method is well known for solving constraint problem, it's problem dependent and the performance is not superior to decoder method (Michalewicz, 1999). Herein we use decoder technique as our main searching scheme after we try several unsuccessful runs on penalty method. After the decoded value is mapped to the real problem domain, we round the mapped variables for later function evaluations. Numerical study shows the applicability of this approach. Below we briefly organize the decoder method first introduced in Koziel and Michalewicz (1998) as shown in procedure 5.1.

Algorithm 5.1

Step 1: Set $s_k = 0$

Step 2: Do while S_k is feasible

 If $k < J - 1$

 Call Procedure ($s_T, s, k+1$).

 Else

 Set $s_J = s_T - s_{J-1} - \dots - s_1$,

 Cost = $C(s_1, \dots, s_J)$,

 If Cost $< C(s_T)$, Then

$C(s_T) = \text{Cost}$,

$S^* = (s_1, \dots, s_J)$.

 Endif

Endif

Set $s_k = s_k + 1$

Enddo



Procedure 5.1

Step 1: Find an initial feasible reference point $r_0 \in F$, where F stands for feasible set.

Step 2: Define a one-to-one mapping f between the hypercube $[-1, 1]^n$ and the search

space ω . Then the mapping $f : [-1, 1]^n \rightarrow \omega$ can be defined as the following:

$$f(y) = x, \text{ where } x_i = y_i \frac{u(i) - l(i)}{2} + \frac{u(i) + l(i)}{2}, \text{ for } i = 1, \dots, n. u(i) \text{ and } l(i) \text{ are}$$

upper and lower bounds for original design variables. We can see this is a linear

transformation approach, which starts at $(y_i, x_i) = (-1, l(i))$ and ends at

$$(y_i, x_i) = (1, u(i)).$$

Step 3: A line segment between r_0 and any point s at the boundary of ω is defined as

$$L(r_0, s) = r_0 + t(s - r_0), \text{ for } 0 \leq t \leq 1.$$

Step 4: Clearly if F is convex, then L intersects F in exactly one point. If F is concave, then

L may intersect F more than one point. Generally, we may find feasible segments

along L by defining a reverse mapping $\delta : \bigcup_{i=1}^k (t_{2i-1}, t_{2i}) \rightarrow (0, 1]$ as follows:

$$\delta(t) = (t - t_{2i-1} + \sum_{j=1}^{i-1} d_j) / d, \text{ where}$$

$$d_j = t_{2i} - t_{2i-1}, \quad d = \sum_{j=1}^k d_j, \quad \text{and } t_{2i-1} < t \leq t_{2i}.$$

The feasible region for each design vector is through binary search.

Step 5: Finally, given any $a \in [0, 1]$, we can immediately find its accurate feasible

position at L : $\gamma(a) = t_{2i-1} + d_j \frac{a - \delta(t_{2j-1})}{\delta(t_{2j}) - \delta(t_{2j-1})}$. Usually we let $a = y_{\max}$.

Though Procedure 5.1 is designed for continuous variable, we use it to solve our BAP problem. We use it through simple rounding procedure. Algorithm 5.2 below lists our procedures for implementing constrained GA (we call it CGA hereinafter).

Algorithm 5.2

Step 1: Find initial feasible reference point

Step 2: Random generation of initial population within bounds of the hypercube

Do Decoder mapping

Calculate fitness value by the returned mapped variables

Step 3: Sort fitness value of initial population

Step 4: Do while not termination condition

Call Genetic algorithm {Selection, Crossover, Mutation, Elitism}

Do Decoder mapping.

Calculate fitness value by the returned mapped variables

Enddo

Note, the decoder method always return the feasible solution. Alternatively, simulated annealing only searches the feasible region. We use SA to solve both BAP and CAP. The algorithm for our constrained simulated annealing procedures (we call it CSA hereinafter) is shown in Algorithm 5.3.

Algorithm 5.3

Step 1: Set $n = 0$, Initialize T_n, x_n, α, VM etc.

Step 2 : Do while $T_n \geq T_{\min}$

 Do while $m \leq \text{Maxiter of Current temperature}$

 Do while $n \leq \text{Maxiter of VM adjustment}$

 Do while *each variable is searched*

$[x_{\text{new}}, C_{\text{new}}] = \text{Simulated annealing}(x_n, sl)$.

 If $(SL \leq \beta \text{ and } C_{\text{new}} < C_{\text{curbest}})$

$C_{\text{curbest}} = C_{\text{new}}$,

$x_{n+1} = x_{\text{new}}$.

 If $C_{\text{new}} < C_{\text{best}}$

$C_{\text{best}} = C_{\text{new}}$

 EndIf

 ElseIf $(SL \leq \beta \text{ and } C_{\text{new}} \geq C_{\text{cur}}(x) \text{ and } p \leq \exp^{-(\Delta f/T)})$

$x_{n+1} = x_{\text{new}}$.

 Else

$x_{n+1} = x_n$.

 EndIf

 Set $n = n + 1$.

 Enddo

Enddo

Calculate acceptance ratio

Apply VM adjustment

Convergence test.

Enddo

$$T_{n+1} = \alpha T_n.$$

Enddo

Tables 5.1 to 5.7 show the results of applying the above methods. Since we can achieve the maximal service quality with minimum total buffers, \underline{S}_T can be found by putting all the buffers at the last stage and increase the S_T level until SL is satisfied. Table 5.1 to 5.3 relate to BAP. Table 5.1 seeks the optima by fixing $S_T = \underline{S}_T$. Note the mapping nature of Decoder hinders itself from applying CGA directly for this buffer restriction case. It shows CSA performs well except for when $J = 7$. However, the deviation from the optimal solution is very tight. In table 5.2, we release the minimum buffer restriction and compare the results by using CSA and CGA. It shows the performance of CSA with no S_T restriction is superior to that with restriction. Table 5.3 is the comparison of efficiency test for different meta-heuristic methods. It shows ENU is suitable for small-scaled problem. The computation time increase exponentially and therefore intractable for large-scaled problem even though it can find the optimal solution. It seems CSA outperforms CGA both in accuracy and efficiency for our BAP. Therefore we focus on the CSA method throughout the remaining of this chapter. Table 5.4 reports applying CSA on CAP. We fix \mathcal{S} to be the \mathcal{S}^* of table 5.2. It seems the performance is even better when ρ is also optimized. Also note the difference between table 5.2 and 5.4. The SL^* of CAP is superior to that of BAP. Table 5.5 reports applying CSA on BAP and CAP simultaneously. The output of BAP is served as input to CAP and vice versa iteratively until the solution converges. Table 5.6 reports

that we apply CSA directly on the same problem. Notice the performances are the same between these two approaches. Theoretically they should be near. However, the direct method saves much time than iterative method. Direct method clearly outperforms the iterative. In conclusion, it reveals that when we optimize S and ρ simultaneously, the performance of TC^* is the best though SL^* is the worst in most of the cases. Note, Table 5.7 demonstrates the applicability of CSA on larger SC. Except for tables 5.1, 5.2 and 5.3, All the test results of CSA are obtained by choosing the best from 5 random runs with 500 000 iteration per run.

The following is our observation for the above tests. Comparing tables 5.4 and 5.6, it seems that S_T decreases when capacity increases (which is the same as work load decreases) to achieve the best performance. We will investigate this issue further in later experiments. Liu and Yao (2004) observed that higher workload should sequence first in tandem supply system. Spinellis et al. (2000) also noticed the same pattern in finite-buffered supply system. Their SA optimization results suggested buffer should be positioned in increasing order while workload should be arranged in decreasing order. Basically from tables 5.1 to 5.4 we get the same results with the previous studies. Notice we add backorder cost and use different evaluation model, which are different from Liu and Yao (2004). Ours is of dual-buffer design with the first buffer to be infinite, which is different from Spinellis et al. (2000). In table 5.6, workload sequence is consistent with table 5.4, however the “pattern” for buffer is not so obvious. Table 5.7 shows the optimization results for larger SC using SA for $J = 10, 20,$ and 30 . Similar “pattern” seems to maintain in load-sequence allocation. However, the same inconsistency occurs in buffer allocation. To obtain more “reasonable” results, we may have to add more evaluation iterations. In conclusion, for extreme cases that only either end of an SC has buffers, classical methods are adequate to find the global optimum. For general case, meta-heuristics find the near optimal solutions very well. All

the tests are implemented on a Pentium IV 2.0 GHz PC.

Table 5.1 Optimal buffer allocation using CSA, and ENU (with minimum S_T)

J	S_T	S^*		TC^*		SL^*	
		ENU	CSA	ENU	CSA	ENU	CSA
4	29	(0, 0, 6, 23)	(0, 0, 6, 23)	26.924	26.924	0.093	0.093
5	34	(0, 0, 5, 6, 23)	(0, 0, 5, 6, 23)	33.896	33.896	0.099	0.099
6	40	(0, 0, 5, 6, 7, 22)	(0, 0, 5, 6, 7, 22)	41.099	41.099	0.093	0.093
7	45	(0, 0, 4, 6, 6, 8, 21)	(0, 0, 5, 6, 6, 6, 22)	49.019	49.022	0.098	0.098

Note: The results of CSA are obtained by choosing the best from 5 random runs with 10 000 iteration per run. The major parameter settings are 85 for initial temperature and 0.8 for cooling rate and remain the same herein.

Table 5.2 Optimal buffer allocation using CGA, CSA (no S_T restriction)

J	CSA		CGA		CSA	CGA	TC^*	SL^*	
	S^*	S_T	S^*	S_T				CSA	CGA
4	(0, 1, 7, 23)	31	(0, 3, 5, 23)	31	26.630	26.634	(27.824, 1.129)	0.071	(0.068, 0.008)
5	(0, 1, 6, 8, 22)	37	(2, 1, 8, 6, 20)	37	33.567	33.72	(35.548, 2.843)	0.069	(0.080, 0.008)
6	(0, 1, 6, 6, 7, 22)	42	(3, 1, 8, 2, 9, 21)	44	40.998	41.390	(43.250, 1.960)	0.075	(0.065, 1.013)
7	(0, 0, 5, 6, 6, 8, 21)	46	(3, 4, 6, 3, 6, 10, 17)	49	48.945	49.665	(52.540, 4.652)	0.089	(0.072, 0.017)

Note: The results are obtained by choosing the best from 10 random runs with 500 000 iterations and 100 generations per run for CSA and CGA respectively. The major parameter setting is 0.8 for the probability of crossover and 0.01 for the probability of mutation. For CSA, since the convergence of applying VM works well, almost all the outputs are the same except for when $J = 7$, where the outputs are very close. Therefore we only report the statistics for CGA. The first parameter in the parentheses is the mean while the second parameter is the standard deviation for respective performance measures for 10 runs.

Table 5.3 Time performance of CSA, CGA and ENU

J	Time (CPU sec)		
	ENU ^a	CSA ^b	CGA ^c
4	0.251	< 30	> 60
5	0.471	< 30	> 60
6	120.77	< 60	> 60
7	2057.18	< 60	> 60

Note: Number of iterations for CSA: 10 000. Number of generations for CGA: 100. a: ($S_T = \underline{S}_T$), b: ($S_T = \underline{S}_T$) and (no S_T restriction) c: (no S_T restriction)

Table 5.4 Optimal workload allocation using CSA

J	S*	ρ^*	TC*	SL*
4	(0, 1, 7, 23)	(0.82, 0.70, 0.67, 0.64)	23.322	0.015
5	(0, 1, 6, 8, 22)	(0.81, 0.75, 0.71, 0.68, 0.65)	30.009	0.011
6	(0, 1, 6, 6, 7, 22)	(0.86, 0.74, 0.70, 0.67, 0.64, 0.62)	36.312	0.017
7	(0, 0, 5, 6, 6, 8, 21)	(0.87, 0.76, 0.72, 0.68, 0.65, 0.63, 0.61)	42.854	0.019

Note: Most runs obtain the same results except for N=7, where the results are very close and the best is reported. Note the minimum capacity required for all the stages to satisfy SL is 1.23 for N = 4, 5, and 1.24 for N = 6, and 1.25 for N = 7. However, we release this restriction by letting all the lower bounds to be 1.01, such that stability of the queueing system is maintained.

Table 5.5 Optimal buffer and workload allocations using iterative method (J=4)

Iteration	S*	TC	SL	ρ^*	TC	SL
0	(0, 1, 7, 23)	26.630	0.071	(0.82, 0.70, 0.67, 0.64)	23.345	0.012
1	(3, 4, 5, 11)	20.176	0.066	(0.77, 0.65, 0.62, 0.59)	18.994	0.021
2	(1, 4, 4, 9)	17.623	0.067	(0.71, 0.61, 0.59, 0.55)	16.923	0.027
3	(0, 3, 3, 8)	16.070	0.079	(0.65, 0.57, 0.55, 0.52)	15.478	0.037
4	(0, 2, 3, 7)	15.103	0.074	(0.60, 0.54, 0.52, 0.49)	14.883	0.042
5	(0, 2, 2, 7)	14.662	0.061	(0.58, 0.52, 0.50, 0.49)	14.634	0.048
6	(0, 1, 2, 7)	14.477	0.070	(0.55, 0.50, 0.49, 0.48)	14.443	0.053
7	(0, 0, 2, 7)	14.350	0.081	(0.53, 0.48, 0.47, 0.46)	14.314	0.062
8	(0, 1, 2, 6)	14.303	0.065	(0.53, 0.48, 0.47, 0.45)	14.308	0.068
9	(0, 0, 2, 6)	14.299	0.096	(0.50, 0.47, 0.46, 0.45)	14.252	0.081
10	(0, 0, 2, 6)	14.252	0.081	Stop		

Table 5.6 Optimal buffer and workload allocations using CSA

J	S*	ρ^*	TC*	SL*
4	(0, 0, 2, 6)	(0.50, 0.47, 0.46, 0.45)	14.252	0.081
5	(0, 0, 2, 2, 5)	(0.51, 0.47, 0.46, 0.45, 0.43)	17.795	0.093
6	(0, 0, 1, 2, 1, 6)	(0.51, 0.47, 0.46, 0.45, 0.44, 0.43)	21.439	0.096
7	(0, 1, 1, 0, 1, 3, 5)	(0.51, 0.47, 0.46, 0.44, 0.44, 0.43, 0.41)	25.212	0.094

Note: Smaller cases (N=4, 5) converge very fast, though most runs obtain different results (but close), and the best is reported. For N=6, 7, maximum iteration (500 000) are achieved and the best results are reported.

Table 5.7 Optimal buffer and workload allocations using CSA for larger SC.

J	S*	ρ^*	TC*	SL*
10	(0, 0, 0, 4, 0, 3, 0, 1, 2, 5)	(0.54, 0.49, 0.48, 0.48, 0.45, 0.44, 0.42, 0.41, 0.41, 0.39)	37.277	0.100
20	(0, 1, 1, 2, 1, 1, 1, 3, 0, 1, 0, 3, 0, 1, 2, 0, 1, 0, 2, 4)	(0.57, 0.43, 0.53, 0.50, 0.44, 0.58, 0.51, 0.38, 0.31, 0.41, 0.40, 0.37, 0.34, 0.35, 0.40, 0.25, 0.38, 0.28, 0.30, 0.29)	89.669	0.096
30	(9, 2, 1, 1, 4, 2, 5, 6, 1, 1, 0, 10, 0, 1, 1, 0, 3, 0, 3, 1, 1, 0, 1, 2, 0, 1, 0, 1, 1, 2)	(0.87, 0.57, 0.53, 0.33, 0.36, 0.37, 0.24, 0.26, 0.76, 0.24, 0.66, 0.33, 0.36, 0.33, 0.48, 0.40, 0.46, 0.22, 0.14, 0.35, 0.52, 0.47, 0.29, 0.04, 0.04, 0.34, 0.02, 0.04, 0.12, 0.06)	324.018	0.069

Note: CPU Time statistics: J = 10: < 300, J = 20: < 600, J = 30: < 1500

5.3.2 A more general topology

For more general SC such as fig. 5.4, several supply topologies are incorporated in a single SC. For the ease of exposition, here we model an abstract version of fig. 1.1 in chapter 1. However, the same solution method applies to fig. 1.1 as well. We use several $M/M/1$ to represent each inbound logistics (S_j) of the supplier and we assume there is only one $M/M/1$ outbound logistics (T1). The single distribution process (T2) is Hyper-exponential distributed. Notice, if we use a dedicated transportation route for each retailer, the analysis will be a slightly different. Under such configuration, we can use sequential refinement method of Lee and Zipkin (1995) for the analysis for distribution system. However, sequential refinement method is similar to L & Z. It differs by treating each dedicated transportation route as a $M/M/1$ queue and calculates it respectively. In the past, simulation seems to be the only choice for analyzing such complex structure. However, with the assistance of other stochastic modeling procedure, we can still transform the original complex structure into a tractable tandem-form as explained in subsection 3.2.2 of chapter 3 and subsection 5.2.1 of chapter 5. Now the supplier may be from multiple sources. Suppose we have 5 functional stages: supply (S), assembly (A), manufacture (M), outbound logistics (T1) and distribution (T2). Note here T2 may include the processing needed for product differentiation and transporting tailored for non-identical customers' needs as illustrated in inbound logistics of fig. 1.1 in chapter 1.

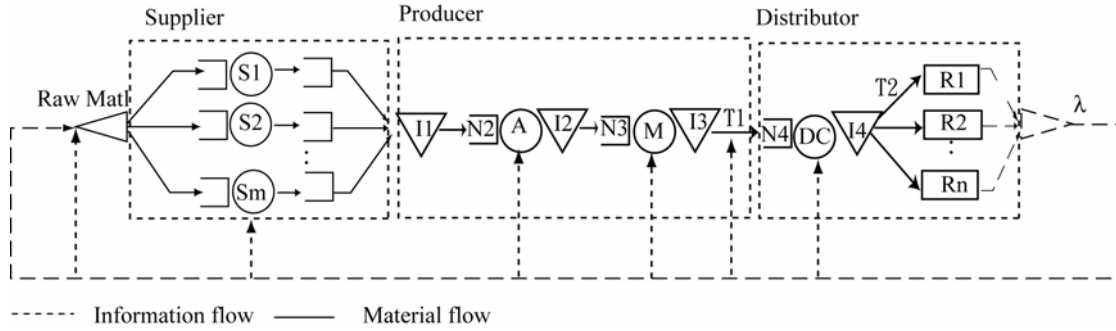


Fig. 5.4 A more general SC model.

Suppose we want to solve BAP and CAP simultaneously with the restriction that the service level for each retailer is not larger than some threshold levels and the arithmetically averaged service level is not larger than some pre-specified level. We formulate the optimization process as follows.

$$\begin{aligned}
 \text{Min } TC(\mathbf{S}, \mathbf{u}) &= h_{wp} \mathbf{WIP} + \sum_{j=1}^R c_h E[I_j] + \sum_{j=1}^R c_b E[B_j] + \sum_{j=1}^J c_j u_j \\
 \text{S.T.} \\
 u_j &\geq \underline{u}, \\
 SL_j &\leq \beta_j, \\
 SL &\leq \beta, \\
 S_j &\in \text{Integer}^+ \cup \{0\}, u_j \in \text{Real}^+.
 \end{aligned}$$

All the parts from different supply sources will be collected whenever one from each supplier is available and immediately sent to the inbound warehouse, I_1 of the producer. We neglect the transferring process of inbound logistics and transit inside producer. Suppose we have 2 suppliers and 4 retailers. The supply distributions are all exponential with identical rates. Thus the results of subsection 5.2.1 can be applied here. Under this setting, the mean response time of the FES is 3/2 times larger than that of the original $M/M/1$ queue (Bolch, 1998). Assume the parameters for retailer demand rates are $(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (0.1, 0.2, 0.3, 0.4)$, distribution rates for T2 are $\mathbf{u} = (1, 2, 3, 4)$. Notice here we assume non-identical retailers with different serving (distribution) rates. The \underline{u} is set 1.01 to preserve the stability in a queuing system. Remember the original formula of the

number of order in the system of a $M/PH/1$ is $L = (1 - \rho)\alpha_* \mathbf{R}(\mathbf{I} - \mathbf{R})^{-2} \mathbf{1}$ (see (3.9) of chapter 3 for details), however we have to modify this performance measure to account for different distribution rate u_j . Applying Little's formula, we get the actual queue length for each retailer $Lq = L - \rho$, and so the actual sojourn time for each distribution process is $Ws = Lq / \lambda + 1 / u_i$. The accuracy of the modified approximation model is verified through simulation study. We setup the experimental design in table 5.8 for this verification process. For the sake of simplicity, we neglect the cost incurred in capacity when making the comparison. Table 5.9 (related to TC and other measures) and 5.10 (related to SL) show the results. The deviations of the approximated model (App) and the simulation model (Sim) of all performance measures are very tight for most of the test cases. High deviations occur at some expected backorder levels and service level where both app and sim are extremely low. From the high accuracy of the tables, we are confident that the approximation model is also sufficient to act as the evaluation function for the optimization process of such more general SC topology.

From section 5.1, the roles played by buffer and capacity seem to be the same. The increase in one variable may decrease the other to maintain low cost while striving to preserve SL requirement. Observations derived from tables 5.9 and 5.10 justify our conjecture. The following lists the simple rules-of-thumb after observing the results of the tables. Note rules 5.1 and 5.2 relate to TC while rules 5.3 and 5.4 relate to SL .

Rule 5.1 If all resources are low, increase capacity only, not else to decrease cost. This measure decreases TC from 31.552 to 7.29 as seen from table 5.9. Any other measure results in more TC . This is because no matter what resource(s) is/are increased, $E[B_s]$ decreases. However, the “side effect” of increasing capacity is the mildest. It decreases WIP and increase a little $E[I_s]$ while all other measures increase either WIP and/or $E[I_s]$ significantly.

Rule 5.2 If there is/are any resource(s), which is/are already high, it may be not necessary to increase any other resource(s) to prevent from increasing cost. When resource(s) is/are already high, it may be enough to fight against demand uncertainty and therefore adding extra resource(s) result(s) in marginal decrease in $E[B_s]$ but increase either WIP and/or $E[I_s]$ significantly. The overall result is not beneficial.

Rule 5.3 If all resources are low, increase any resource(s) as much as possible to increase service quality. Basically, SL is closely related to $E[B_s]$. When $E[B_s]$ decreases service quality increases (i. e. SL decreases). Since any resource expansion plan lowers the risk of demand uncertainties, more resources increase service quality more.

Rule 5.4 If there is/are any resource(s), which is/are already high, the benefit of increasing any other resource(s) may be marginal. As can be seen from table 5.10, when resource(s) are high enough, adding more resource(s) is/are useless to decrease SL (adding intermediate buffer causes SL to remain 0.01).

To conclude, adding more capacity seems to be the most beneficial measure to reduce TC if buffer is low and the capacity incurred cost is neglected. However, when SL is also concerned, the decision may be different. Decision maker has to make adequate measure to trade-off between cost reduction and service requirement. In this example our rules-of-thumb cope up with the reasoning of system dynamics. However, it may not be true under different assumptions. For example, if different cost structures and other factors not concerned herein are adopted it may result in different conclusions. For the ease of decision, adequate optimizer such as introduced in this chapter seems to fit in. Suppose the required service level for retailers is $\beta = (0.1, 0.2, 0.3, 0.4)$ and $\beta = 0.15$. Note β is the averaged service requirements. Table 5.11 is such exercise. Interestingly, the positioning of the workload sequence in distribution stage and the positioning of the buffer in other stages are consistent with the previous results.

Table 5.8 DOE setting for approximation validation of more general topology

Decision Factors	Levels
Capacity (u_1, u_2, u_3, u_4)	Low: (2.25, 1.5, 1.5, 1.5) High: (6, 4, 4, 4)
Intermediate buffer (S_1, S_2, S_3, S_4)	Low: ((1, 1, 1, 1) High: (4, 4, 4, 4)
Ending buffer (S_5)	Low: ((1, 1, 1, 1) High: (4, 4, 4, 4)

Note: (u_1, u_2, u_3, u_4) are service rates for supply, assembly, manufacture, and outbound logistics respectively; (S_1, S_2, S_3, S_4) are base stock levels for I_1, I_2, I_3 , and DC respectively. S_5 is a vector, representing base stock levels for different retailers.

Table 5.9 Approximation validation (1).

Capacity	Intermediate buffer	Ending buffer	TC			WIP			$E[I_5]$			$E[B_5]$		
			Sim	App	%Err	Sim	App	%Err	Sim	App	%Err	Sim	App	%Err
Low	Low	Low	31.552	31.602	0.16	7.471	7.679	2.79	1.822	1.725	-5.35	2.443	2.604	6.58
Low	Low	High	19.369	19.053	-1.63	7.471	7.679	2.79	11.729	11.493	-2.01	0.350	0.372	6.31
Low	High	Low	15.792	15.882	0.57	16.233	16.236	0.02	3.080	3.036	-1.42	0.471	0.473	0.36
Low	High	High	23.054	23.056	0.01	16.233	16.236	0.02	14.645	14.598	-0.32	0.035	0.034	-2.86
High	Low	Low	7.290	7.165	-1.72	4.552	4.594	0.93	3.318	3.266	-1.57	0.164	0.160	-2.32
High	Low	High	17.450	17.413	-0.21	4.552	4.594	0.93	15.156	15.107	-0.32	0.003	0.001	-70.00
High	High	Low	13.028	12.813	-1.65	16.448	16.449	0.00	3.392	3.373	-0.56	0.136	0.122	-10.66
High	High	High	23.487	23.482	-0.02	16.448	16.449	0.00	15.257	15.252	-0.03	0.002	0.001	-70.00

Note: %Err = (App - Sim) / Sim × 100 %

Table 5.10 Approximation validation (2).

Capacity	Intermediate buffer	Ending buffer	$(SL_1, SL_2, SL_3, SL_4, SL)$		
			Sim	App	%Err
Low	Low	Low	(0.384, 0.544, 0.642, 0.706, 0.569)	(0.360, 0.524, 0.619, 0.672, 0.543)	-4.57
Low	Low	High	(0.008, 0.047, 0.109, 0.18, 0.086)	(0.007, 0.043, 0.100, 0.169, 0.080)	-6.98
Low	High	Low	(0.167, 0.224, 0.268, 0.304, 0.241)	(0.167, 0.212, 0.259, 0.280, 0.230)	-4.77
Low	High	High	(0.001, 0.005, 0.013, 0.023, 0.011)	(0.001, 0.005, 0.014, 0.024, 0.011)	0.00
High	Low	Low	(0.131, 0.168, 0.202, 0.233, 0.184)	(0.133, 0.157, 0.183, 0.208, 0.170)	-7.61
High	Low	High	(0, 0.001, 0.001, 0.002, 0.001)	(0.000, 0.001, 0.002, 0.004, 0.002)	100.00
High	High	Low	(0.119, 0.145, 0.17, 0.193, 0.157)	(0.120, 0.145, 0.164, 0.177, 0.152)	-3.18
High	High	High	(0, 0, 0.001, 0.001, 0.001)	(0.000, 0.000, 0.001, 0.003, 0.001)	0.00

Note: $SL = (SL_1 + SL_2 + SL_3 + SL_4) / 4$.

Table 5.11 Optimal buffer and workload allocations for more general SC.

J	$(S_1, S_2, S_3, S_4, S_5)^*$	$(\rho_1, \rho_2, \rho_3, \rho_4, \rho_5)^*$	TC*	SL*
5	(0, 0, 2, 3, 2, 2, 2)	(0.46, 0.48, 0.47, 0.46, 0.84, 0.63, 0.52, 0.45)	26.946	(0.03, 0.08, 0.14, 0.18, 0.11)
6	(0, 0, 1, 2, 3, 2, 2, 2)	(0.46, 0.48, 0.47, 0.46, 0.45, 0.84, 0.63, 0.52, 0.45)	30.875	(0.03, 0.09, 0.14, 0.19, 0.11)
7	(0, 0, 1, 1, 2, 3, 2, 2, 2)	(0.47, 0.48, 0.47, 0.47, 0.46, 0.44, 0.83, 0.62, 0.52, 0.45)	34.946	(0.04, 0.09, 0.14, 0.19, 0.12)

Note: CSA achieved convergence after 71 200 iterations for J = 5, 81 000 iterations for J = 6 and 90 000 iterations for J = 7.

5.4 Discussion

We have discussed the suitable optimization solutions for all the proposed problems of tandem and more general SC forms and related topics of resource expansion planning. It's well known that the parameter settings of meta-heuristic methods such as the “cooling rate” in SA, probabilities of crossover and mutation etc. may play important role in performance enhancement. For the sake of brevity, we neglect this investigation in this chapter. Now we turn to the sensitive part of the optimization process as well as other issues related to reliability of an SC. To see all the impacts on TC and service quality, we set different values of β starting from 0.01 to 0.3, incrementing at 0.01.

5.4.1 Penalty of BC model

Assume $\mu = 2.5$, all the other operation parameters, including cost structure are the same as in case 5.3 of subsection 5.3.1. If the cost structure is that c_b plays dominant factor then we conjecture that our method will find the optimal buffer configuration such that the backorder level is low enough while satisfying the service constraint. This effect is the so-called “penalty of the BC model” (Boyaci and Gallego, 2001). Here BC stands for backorder cost. We illustrate this effect by the following tests. Let c_b have different levels: 0.5, 1, 5, and 10. We run the optimization model. The final results of TC* and SL* against different β levels are shown in fig. 5.5 and 5.6. Fig. 5.5 shows TC* decreases when β increases. In all β settings, TC* becomes high when c_b is set large. Because the S_T of high

c_b will be higher than that of low c_b to prevent the backorder cost from getting too high. That explains why the $E[B]$ of high c_b converges earlier than low c_b as in fig. 5.6. Since high c_b incurs high backorder costs, it explains why TC of high c_b is higher than that of low c_b at the same β (fig. 5.5). To conclude, the higher of c_b , the less change in S^* , and therefore the less change in TC and $E[B]$ no matter how β degenerates. On the contrary, when c_b is low and β increases, $E[B]$ increases. However, this increase is compensated by the decrease in $E[I]$, and ultimately results in the decreases of TC . This finding seems to cope up with our original conjecture and validate the so-called “penalty of the BC model”.

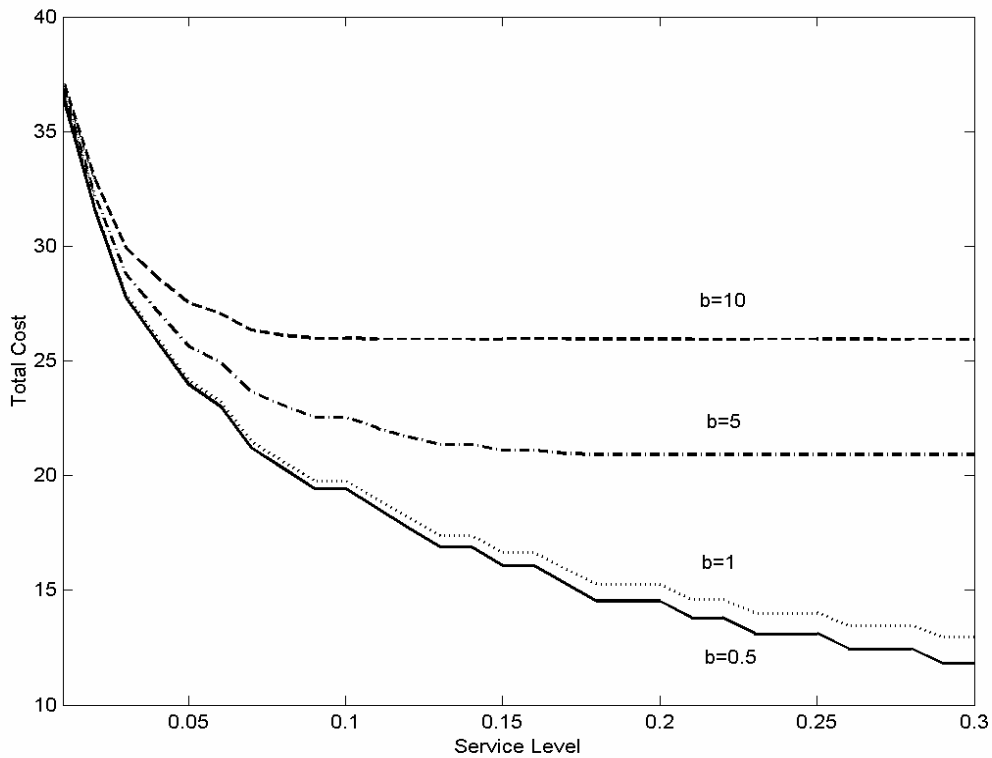


Fig. 5.5 TC as a function of β under different cost structure

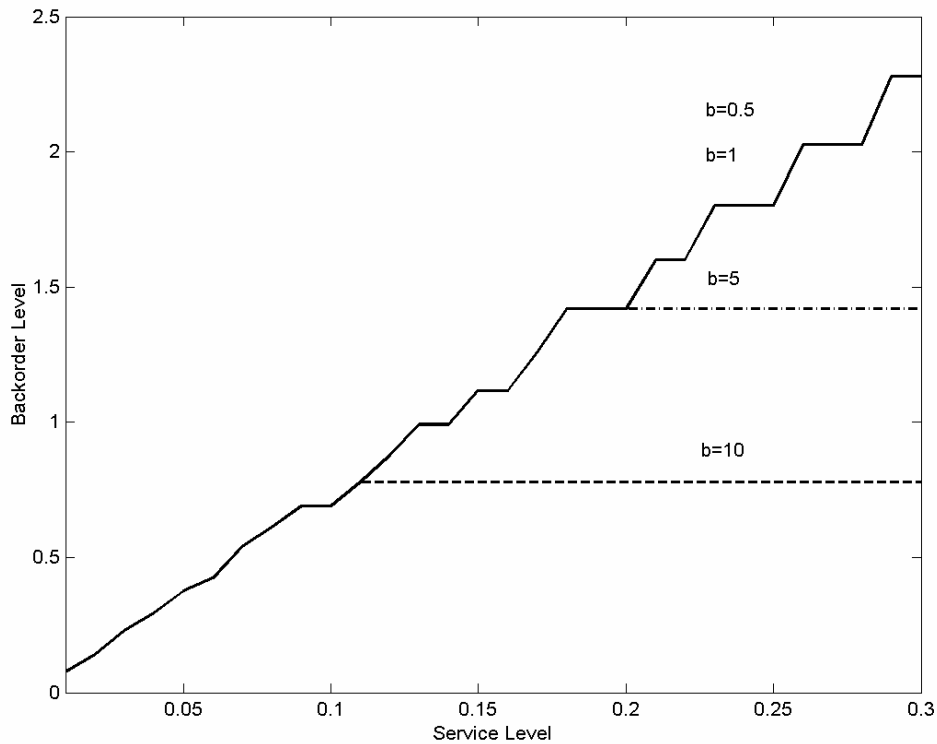


Fig. 5.6 Backorder level as a function of β under different cost structure

5.4.2 Impact of unavailability

Random upstream unavailability may affect the performance of an SC. In this analysis we try to find under what circumstance the impact will be the most disastrous. The decision factors include arrival and service rates of regular jobs as well as irregular jobs such as maintenance or breakdown. Here we analyze breakdown case only. The DOE setting for unavailability test is in table 5.12. Under regular mode, each stage behaves like an isolated $M/M/1$ queueing system. With breakdown incorporated, W_s is lengthened. However, we don't know exactly what factor(s) contribute(s) to the impact of performance. Table 5.13 lists all the possible factors that may be the candidates. Except for the factors in table 5.12, the derivations of the other three factors, i. e., ρ , c_a^2 , c_s^2 are more involved (Wang, 1993). Here we use a different approach as compared with chapter 3. Specifically, we treat the stochastic process as a superposition of regular and breakdown jobs. The

purpose is to acquire the variation in both arrival and service processes. ρ is derived in (3.10) of chapter 3. Below we derive c_a^2 and c_s^2 .

Since breakdown and repair processes are independent random variables, we get squared coefficient of variation of breakdown cycle as $c_b^2 = \frac{c_\xi^2 \cdot \frac{1}{\xi^2} + c_\gamma^2 \cdot \frac{1}{\gamma^2}}{(\frac{1}{\xi} + \frac{1}{\gamma})^2}$. Since all

services are exponential, both c_ξ^2 and c_γ^2 equal 1. From the definition, we get $\lambda_b = 1/(1/\xi + 1/\gamma)$, $p = \lambda_r / (\lambda_r + \lambda_b)$, $q = 1 - p$. So we may write $c_a^2 = p \cdot 1 + q \cdot c_b^2$. Since we may express $E[T_s] = p \cdot 1/u + q \cdot 1/\gamma$, c_s^2 can be derived similarly:

$$c_s^2 = \frac{p \cdot \frac{1}{u^2} (1 + c_u^2) + q \cdot \frac{1}{\gamma^2} (1 + c_\gamma^2) - E^2[T_s]}{E^2[T_s]}. \text{ Also } c_\xi^2 \text{ and } c_\gamma^2 \text{ equal 1.}$$

Notice regular service can be treated as $M/M/1$ queueing system and random unavailability can be treated as $GI/G/1$ queueing system. Table 5.13 lists the comparison of the Ws under these two service modes. It reveals the following interesting fact: Ws in $GI/G/1$ is not infected by ρ , c_a^2 , or c_s^2 individually. It is influenced by the synthetic effects of λ , u , ξ , and γ . For example, when arrival rate is high, service rate is low, breakdown rate is high, and repair rate is low (No. 6), the impact is the highest. The performance degradation, %dev rise as high as near 6 times of the original performance. On the contrary, when arrival rate is low, service rate is high, breakdown rate is low, and repair rate is high (No. 11), the impact is the lowest. These effects cope up with our intuitions. Since Ws is impacted by λ , u , ξ , and γ , we may want to investigate its influence on system performance. Table 5.14 lists the impacts on TC and SL of unavailability by different stages under the worst case (No. 6) and $S = (16, 16, 16, 4)$, where 4 is the base stock for each retailer. Apparently, the impact increases in stage. Table 5.15 lists the impacts under the best case

(No. 11) and $\mathbf{S} = (4, 4, 4, 1)$. Interestingly, we find that upstream unavailability may not always cause adverse results. In table 5.15, TC of upstream unavailability occurred before the second stage is no worse than the original $M/M/1$ QN. Generally, the breakdown increases WIP, decrease $E[I]$ and increase $E[B]$. The service level gets worse because of large $E[B]$. Looking closely at the components of WIP (WIP , not shown here), we find at the broken stage the L increases (except for the first stage, which we don't count) but its $E[I_i]$ decreases and so is/are its downstream stage(s). $E[B]$ may increase if the impact of breakdown upstream last to the end stage or remain the same if the impact is neglected.

Table 5.12 DOE setting for unavailability test.

Decision Factors	Levels	
λ	Low: 0.05	High: 0.1
u	Low: 0.125	High: 0.25
ξ	Low: 0.0125	High: 0.025
γ	Low: 0.125	High: 0.25

Table 5.13 Comparison of regular and random unavailability mode.

No.	λ	u	ξ	γ	ρ	c_a^2	c_s^2	$M/M/1$	$GI/G/1$	%dev
1	0.1	0.25	0.025	0.25	0.44	0.9742	1	6.667	8.143	22.14
2	0.1	0.25	0.025	0.125	0.48	0.976	1.2076	6.667	10.462	56.92
3	0.1	0.25	0.0125	0.25	0.42	0.9852	1	6.667	7.379	10.69
4	0.1	0.25	0.0125	0.125	0.44	0.9858	1.1509	6.667	8.429	26.43
5	0.1	0.125	0.025	0.25	0.88	0.9742	1.0916	40.000	76.000	90.00
6	0.1	0.125	0.025	0.125	0.96	0.976	1	40.000	272.000	580.00
7	0.1	0.125	0.0125	0.25	0.84	0.9852	1.053	40.000	53.500	33.75
8	0.1	0.125	0.0125	0.125	0.88	0.9858	1	40.000	78.667	96.67
9	0.05	0.25	0.025	0.25	0.22	0.9565	1	5.000	5.744	14.87
10	0.05	0.25	0.025	0.125	0.24	0.9591	1.2479	5.000	6.737	34.74
11	0.05	0.25	0.0125	0.25	0.21	0.9733	1	5.000	5.367	7.34
12	0.05	0.25	0.0125	0.125	0.22	0.9742	1.2148	5.000	5.846	16.92
13	0.05	0.125	0.025	0.25	0.44	0.9565	1.1509	13.333	16.000	20.00
14	0.05	0.125	0.025	0.125	0.48	0.9591	1	13.333	19.692	47.69
15	0.05	0.125	0.0125	0.25	0.42	0.9733	1.0951	13.333	14.621	9.66
16	0.05	0.125	0.0125	0.125	0.44	0.9742	1	13.333	16.286	22.14

Note: $M/M/1$: W_s of normal service mode; $GI/G/1$: W_s of random unavailability mode. % dev = $(GI/G/1 - M/M/1) / (M/M/1) \times 100\%$

Table 5.14 Impact of unavailability by different stages (worst case).

Stage	TC	% dev	SL	WIP	$E[I]$	$E[B]$
$M/M/1$ QN	38.957		0.067	48.128	12.147	0.275
S1	72.399	85.85	0.211	31.477	9.991	4.667
S2	107.750	176.59	0.288	58.130	8.853	6.983
T1	143.590	268.59	0.403	63.347	7.131	10.479
T2	198.108	408.53	0.581	71.328	4.469	15.797

Note: $S = (16, 16, 16, 4)$. % dev = $[(TC \text{ of unavailability impact}) - 38.957] / 38.957 \times 100\%$

Table 5.15 Impact of unavailability by different stages (best case).

Stage	TC	% dev	SL	WIP	$E[I]$	$E[B]$
$M/M/1$ QN	9.912		0.059	12.000	3.764	0.015
S1	9.903	-0.09	0.059	11.982	3.764	0.015
S2	9.912	0.00	0.059	12.000	3.764	0.015
T1	9.912	0.00	0.059	12.001	3.764	0.015
T2	9.927	0.15	0.063	12.019	3.748	0.017

Note: $S = (4, 4, 4, 1)$. % dev = $[(TC \text{ of unavailability impact}) - 9.912] / 9.912 \times 100\%$

5.4.3 Impact of poor quality

Poor quality causes reworking and capacity loss (Chapter 3). Applying Jackson's rule, the influenced servers include the point where poor quality is identified, all the way back to the first stage. Assume $\lambda = 0.01$, \mathbf{u} all equal to 0.055, fig. 5.7 to 5.10 graph the impacts of δ on TC and SL by different feedback location for $S = (4, 4, 4, 1)$ and $S = (16, 16, 16, 4)$ respectively. δ ranges between 0.1 and 0.8, incrementing at 0.01. Basically we obtain similar results as in the analysis of server unavailability. The impact may not always be adverse when δ is not so high. This is because server(s) in the feedback range cause(s) capacity to loose and therefore L to increase in front of those servers. As in the server unavailability analysis, $E[I_j]$'s of those servers and their downstream stage(s) decrease. The synthetic effect is the trade-off of the above two impacts. It depends on the buffer size. When buffer is low and δ is very high, capacity may loose dramatically with not enough buffer to fight against demand uncertainty. It causes the $E[B]$ to increase very high and so

is TC . Fig. 5.7 shows this effect. When buffer is high, TC may not rise so much even when δ is very high. The end result is like fig. 5.9. However, SL will be no better than when δ is 0. Even when δ is small the feedback occurred in T2 may start to degrade significantly as shown in fig. 5.8. Also we notice the impact of TC and SL is large when the feedback occurs in downstream stages at high δ .

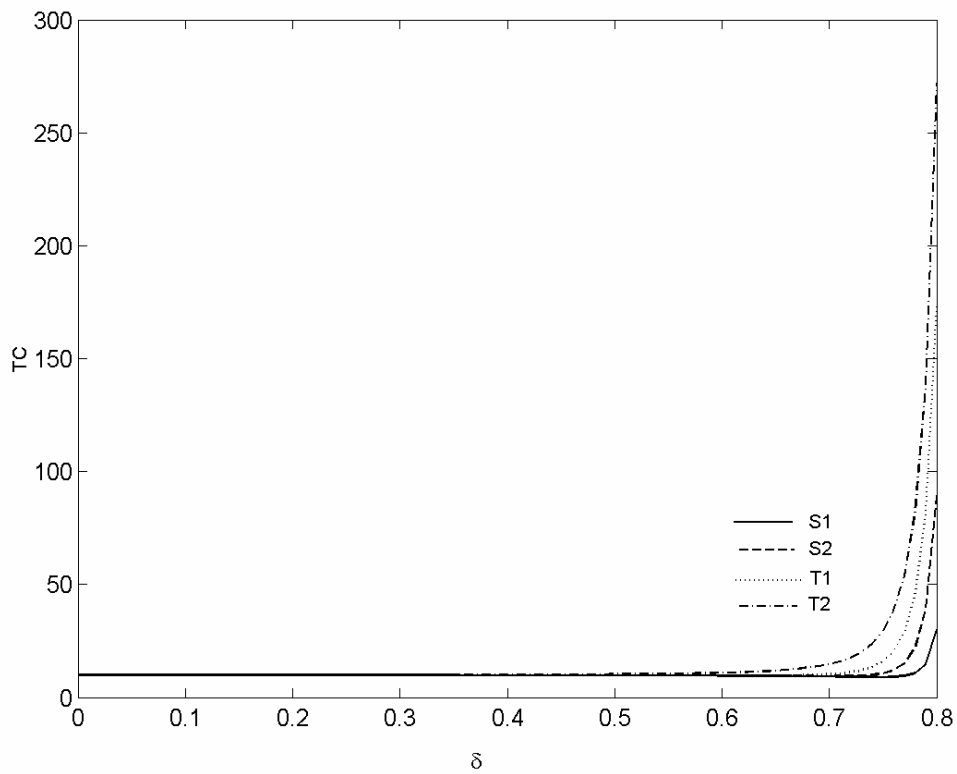


Fig. 5.7 Impact of δ on TC by different feedback location $S = (4, 4, 4, 1)$.

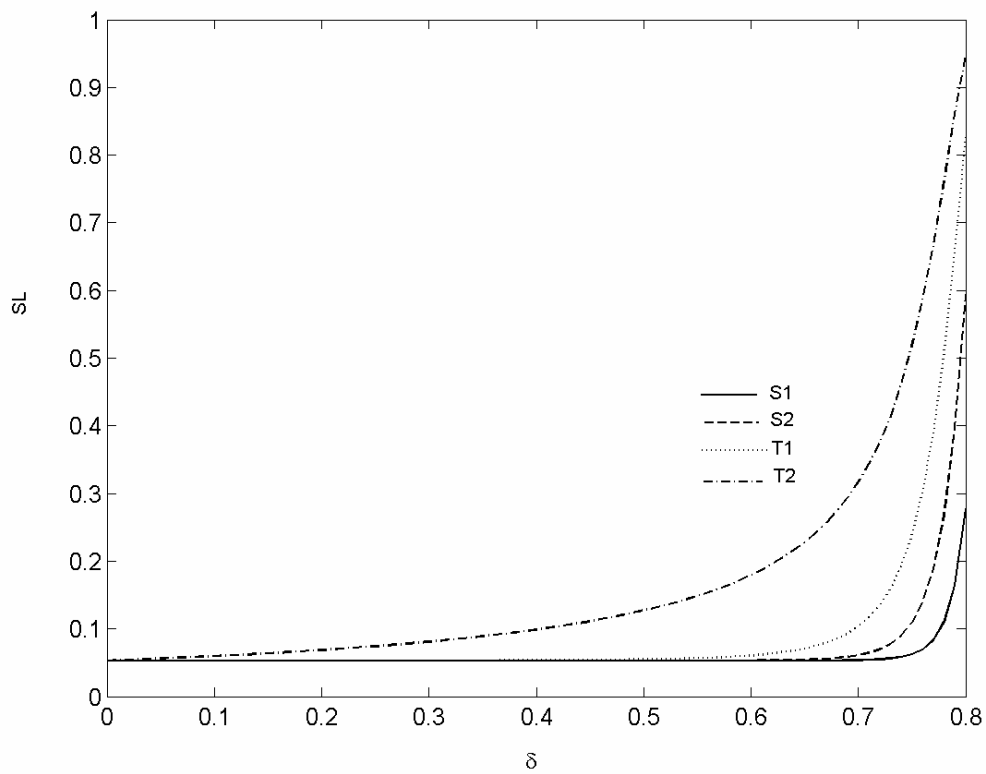


Fig. 5.8 Impact of δ on SL by different feedback location $S = (4, 4, 4, 1)$.

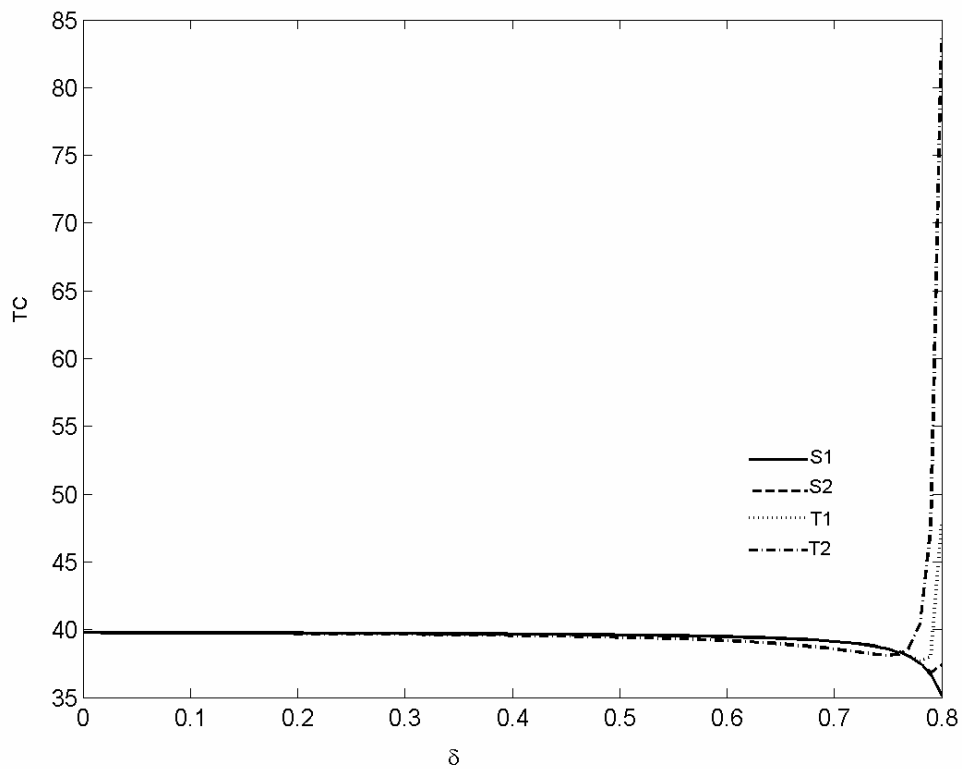


Fig. 5.9 Impact of δ on TC by different feedback location $S = (16, 16, 16, 4)$.

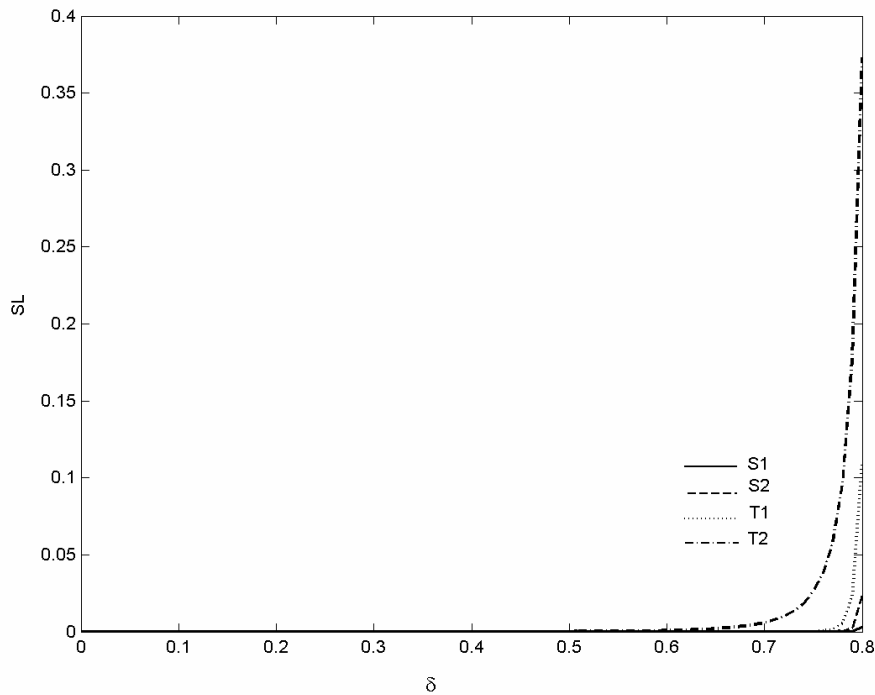


Fig. 5.10 Impact of δ on SL by different feedback location $S = (16, 16, 16, 4)$.

5.5 Concluding remarks

This chapter tries to solve the buffer and capacity allocation problems of an SC under random supply and demand environment. We showed that our analytic model combined with the meta-heuristic search, specifically Simulated Annealing is an efficient and effective approach for optimization of such supply chain problems. Herein we proposed adequate solution measures of classical or modern approach to solve supply chain problems with different topologies and problem formats. For more general SC topology, we developed a naïve and original method by simply transforming the original topology into tandem form through QBD process and another existing stochastic model in the literature. Numerical study shows accurate results when compared to simulation runs. The observations therein may provide simple rules-of-thumb for decision maker of an SC for resource expansion planning. By incorporating feedback and server breakdown factors into a supply chain study, the study provides more practical solutions as compared to other

related works. The discussion on the impacts of upstream unavailability and imperfect quality may provide valuable information for decision maker to get deeper insights into the system behavior of an SC under unreliable supply situations. The system designer can thus incorporate these uncertainty factors into the optimization process to get more robust results. To our knowledge, such investigation on unreliable supply chain is still not much.

In the numerical study, we find that MTS (with intermediate inventories) is not inferior to MTO (with end-stage inventories only) under any service requirement. Special case occurs when all holding cost settings are the same, there is no need to move stocks upstream to lower the holding costs and therefore MTS is degenerated o MTO with the same performance. The test of penalty of backorder cost is consistent with another related work with similar problem but different processing settings. Also "the pattern" of resource positioning seems to be consistent with other related works.



Chapter 6 Conclusions

The most important contribution of this study is that we extend the original work of L & Z. We fit it into an SC domain by the adequate “transformation” of non-tandem systems. The sophisticated processing logic inside each echelon can also be handled by our proposed approach, for example server unavailability herein. Though not exhibited herein, we believe this approach can also handle any arbitrary service distribution since phase type can approximately represent any probability distribution (Svoronos and Zipkin, 1991) and $/PH/1$, where the arrival process can be of Markovian or phase-type, is a well-developed domain of QBD process (Neuts, 1994).

We also show the possibility of extending the original model to handle other order/replenish scheme. Specifically that of (r, q) type is successfully tested though the accuracy is satisfactory only under specific conditions. Integrating the results of QBD modeling for parallel processing with L & Z is also tested with limited success. To make L & Z more general, more efforts are needed in the future.

In conclusion, we have demonstrated that by using the matrix analytic approach, the evaluation of a complex SC using base stock as control logic, performs as expected through simulation verification. The relative errors between Approximation and Simulation are all below 10% for retailers adopting MTO policies in our test problems. When all the retailers also adopt the MTS policy, numerical studies show that the approximation is accurate for medium traffic intensity and acceptable for high traffic intensity. This shows that the matrix analytic approach by combining L & Z and other decomposition technique such as QBD for supply chain analysis is feasible, therefore its application is not limited to tandem processing queue as reported by L & Z and Zipkin (1995) but also for broader context. Herein we show its application on a tandem SC where the end stage is a distribution system with identical or non-identical retailers.

The non-identical case refers to non-identical demand rates with non-identical distribution processing rates. Also we transformed and incorporate a fork-join type supplier's process into our evaluation model to make it more general. In the literature of the stochastic production-distribution system, most models are developed and analyzed separately. Those models are usually difficult to integrate as one single model.

Next, the MMPP models are developed for different scenarios related to uncertainty to investigate respective system behavior for each individual stage under uncertainty concern. After linking together all the stages the system behavior of the whole chain is analyzed by queueing network analysis (QNA) under MTO supply mode. However, the QBD decomposition for parallel processing under various uncertainties may not be suitable under MTS mode. From the study herein, the application only works under specific conditions. The application of the analytic approach on resource allocation, specifically server and repairman decision is demonstrated through an illustrative example. The results show the analysis provides valuable managerial insights.

Finally, we demonstrated various optimization methods for the optimization of various SC topologies under service constraints. We showed the impact on system performance due to upstream unavailability or poor quality under different service levels. The unreliability concern in SCM is seldom explored in the literature, to our knowledge. Simchi-Levi et al. (2003) mentioned that the impact on the performance of an SC due to 911-terror-attack in 2001 is influential, including delayed delivery, delayed customs, poor communications. On the other hand, the impact on an SC due to supply and demand uncertainty in the long run is obscure. Herein, we assessed the above argument through quantitative analysis. To sum up, we characterize the major contributions of this work:

1. Extending L & Z to allow not just for tandem supply network but also for more general supply form. The impact of upstream unavailability on the performance of an SC in the long run is made clear through empirical study.

2. Proposing feasible modeling approach to investigate the impact of non-stationary demand and unreliable service processes on the performance of an SC.
3. Relaxing the single server assumption usually used in tandem queueing models for SC analyses. Though numeric study is satisfactory for low traffic intensity only.
4. Exploring the possibility of extending the analytic model to allow for other inventory control policies.
5. Investigating cases in applying classical or modern optimization methods for solving stochastic SC problems. The obtained results consolidate several previous researches with new insights.

We also prove that the average number of operative machines is equal (proportional) to the average number of machines under repair when mean time to failure and mean time to repair are the same (proportional) by using a matrix algebraic approach. This property is not explicitly related to repairman. During the process of this study we encountered several problems, which raised opportunities for further study. Below we itemize these problems and propose some possible strategies for research.

Evaluation model

The QBD combined with L & Z forms the mainframe of our evaluation model. We tested the performance of the proposed evaluation model in a tandem setting. As can be seen herein, the approximation model has its limitation, to attain a more general setting, alternative evaluation model consisting of new decomposition method and new approximation method may be necessary for other topology and system dynamic of an SC not addressed herein. For example, problems dealing with generalized QN with arbitrary input and service rate and planned inventory setting are still open.

System dynamic inside each echelon

In this work we used abstract level of SC modeling. Though chapter 4 discussed several details in modeling the behavior of an SC, it's not enough. To unveil the System

dynamic inside each echelon and relate it to system performance, more thorough studies should be done. For example, herein we assumed identical processing rates inside assembly logic, what if they are non-identical. Also we used simple model to represent transportation activity. However, practical logistics is usually more complicated than the model formed herein. Optimizing the SC model with more practical logistics concern is an interesting area. Also future study may investigate the impact of uncertainties on resource design decision as functions of correlated effect of MMPP input.

Optima seeking

Due to accuracy reason, we do not put all the parameters investigated herein into a nutshell and optimize it. However, such modeling approach may be necessary for practical use. Further, the evaluation function of the studied problem is more complicated than other engineering optimization problems using the same meta-heuristic search methods. When evaluation model is changed or becomes more intractable because of more complex SC structure and/or system dynamics, more efficient and effective search algorithm other than meta-heuristic as suggested herein may have to be developed. Heuristic approaches such as those in Boyaci and Gallego (2001) are examples. Alternatively, in addition to GA and SA, other state-of-the-art meta-heuristics techniques such as Ant colony algorithm, scatter search, Tabu search among others, may provide different solution flavors.

Control policy

As we reviewed in chapter 2, there are many other control policies which are attributed to “pull type” production/inventory control as studied herein. In this study we used this (r, q) control policy by analogous transformation of the performance formula of Zipkin (2000). It works well under some situations. However, further study regarding extension of the analytic model to other inventory control schemes, such as (s, S) or KANBAN may have to be addressed in the future. On the other hand, periodic review policy as opposed to continuous review policy studied herein may also be investigated.

Buffer design

We used dual-buffer design herein to develop our evaluation model. The input buffer is assumed infinite. There are more SC designs with only one finite buffer due to space limitation or cost concern. The single buffer design has to take blocking effect into its modeling logic. It may be interesting to compare the performance between single and dual buffer designs under various SC topologies. Also dual buffer with finite waiting line is an alternative design, which may be more realistic due to practical reason.

Value of the information

In this study, we used centralized information sharing scheme. When demand is generated at the end stage, all of the stages along the chain immediately reflect this information. It is well known that decentralized information sharing scheme is the cause of so-called “bullwhip effect” in an SC. The demand information is exaggerated toward upstream stages and the variation of demand is amplified along the chain to upstream stages. Further study may investigate the saving of cost due to the centralized information.

Push-pull boundary

In Simchi-Levi et al. (2003), “pull” type of inventory control policy is suitable for the case when demand is highly uncertain and no economy of scale exists. Under these conditions MTO supply mode is adequate. On the contrary, when demand uncertainty is low and economy of scale exists, MTS supply mode is more adequate. However, mixed type of supply mode is possible when one adequately selects the so-called “Push-pull boundary”. Before the boundary, push (MTS) supply is conducted while pull (MTO) supply is executed after the boundary. The performance comparison between our model and the design of this mixed policy is an interesting research area.

The role of the warehouse

Recently, there are several arguments regarding the role of the warehouse in distribution stage. Cross-docking policy argues the inventory holding cost can be largely

saved and so is the supply lead-time. Direct sale policy argues there is no need for the existence of the warehouse. The supply of the manufacturer may directly serve the end stage retailers to save operation cost. Since it is obviously shown herein, the complex interaction of the SC decides the performance of an SC, more thorough studies may have to be conducted to justify which policy is most suitable under what conditions.

As we stated in literature review of chapter 2, high-level modeling approach may be another direction for SCM such as Stochastic Petri net (SPN). Most researches on SPN or generalized SPN (GSPN) can be found in computer and telecommunication areas. Alexander (2001) first showed how to use MGM as the main solving process for QBD model. They illustrated how to map from an infinite Stochastic Petri net (iSPN) model to a QBD process for MGM analysis. Trivedi (2002) illustrated the technique of transforming the reachability graph (RG) obtained from SPN to corresponding CTMC generator matrix in detail. Arns et al. (2002) used Proc/B, a notational description tool, which can then be translated to queueing network or SPN for performance calculation. They used the notation model to compare ordinary distribution and web retailing by constructing synchronized (parallel) sub-models of Proc/B as SPN model and solve the underlying CTMC and then aggregated them as a single queue. Then the overall QN is solved by product-form solution. Under their approach, all the underlying transformation and calculation were automatically accomplished.

Appendix A QBD process and simulation model

A.1 QBD Process

Define the phase type distribution of a Markov process as a stochastic process having parameters m, G_* , and α_* if it can be expressed as a first-passage time random variable, that is, $T = \min\{t \geq 0 : Y_t = \Delta\}$, for a Markov process with state space $E = \{1, \dots, m, \Delta\}$, generator

$$G = \left[\begin{array}{c|c} G_* & G_\Delta \\ \hline \mathbf{0} & 0 \end{array} \right],$$

and the initial probability vector $\alpha = (\alpha_* | 0)$. The generator of an $M/PH/1$ queue can then be represented in the following matrix form:

$$Q = \left[\begin{array}{ccccccc} -\lambda & \lambda\alpha_* & & & & & \\ G_\Delta & G_* - A & A & & & & \\ & G_\Delta\alpha_* & G_* - A & A & & & \\ & & G_\Delta\alpha_* & G_* - A & A & & \\ & & & G_\Delta\alpha_* & G_* - A & A & \\ & & & & G_\Delta\alpha_* & G_* - A & \dots \\ & & & & & \vdots & \ddots \end{array} \right] \quad (A.1.1)$$

Where $A = \lambda I$. Then we can find the steady-state probability vector $\mathbf{p} = (p_0 | p_{11}, p_{12}, \dots | p_{21}, p_{22}, \dots | \dots) = (p_0 | \mathbf{p}_1 | \mathbf{p}_2 | \dots)$ as follows.

Combining (A.1.1) with the equations $PQ = \mathbf{0}$ yields the following system of equations,

$$\begin{aligned} -\lambda p_0 + p_1 G_\Delta &= 0 \\ \lambda\alpha_* p_0 + p_1(G_* - A) + p_2 G_\Delta\alpha_* &= \mathbf{0} \\ p_1 A + p_2(G_* - A) + p_3 G_\Delta\alpha_* &= \mathbf{0} \\ p_2 A + p_3(G_* - A) + p_4 G_\Delta\alpha_* &= \mathbf{0} \\ &\vdots \end{aligned} \quad (A.1.2)$$

The solution of (A.1.2) involves the characteristic equation

$$A + R(G_* - A) + R^2 G_\Delta\alpha_* = \mathbf{0}. \quad (A.1.3)$$

The matrix-geometric solution of (A.1.3) is

$$p_n = cR^n \text{ for } n = 1, 2, \dots, \quad (\text{A.1.4})$$

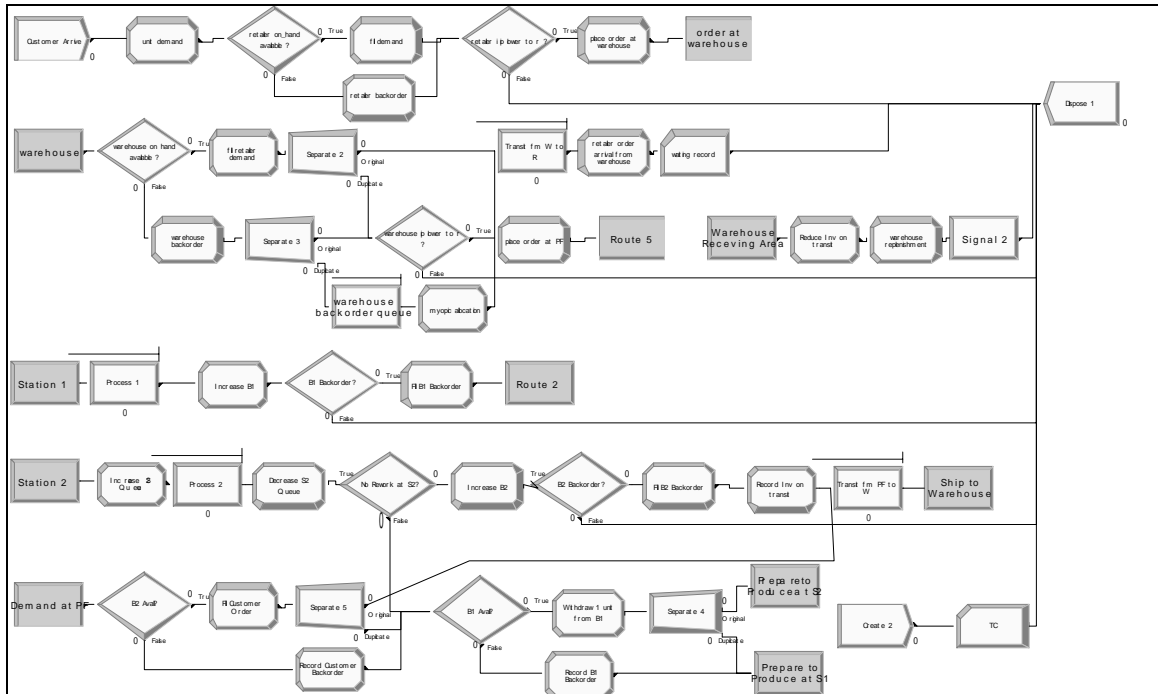
where c is a vector of constants. It can be shown (Neuts, 1994, p 84.) that

$$R = \lambda(A - \lambda \mathbf{1} \alpha_* - G_*)^{-1} \quad (\text{A.1.5})$$

And $c = p_0 \alpha_* = (1 - \rho) \alpha_*$, where $\rho = \frac{\lambda}{\mu} = -\lambda \alpha_* G_*^{-1} \mathbf{1}$, $\mathbf{1}$ is a column vector of ones,

whose dimension is chosen to fit the context.

A.2 Simulation Model of 3.4 by using Arena (Kelton et al., 2002)



Appendix B Moment derivations and related proofs

B.1 First and second moment derivations of chapter 4

Here we show the derivation of $u^{(1)}$, which is not seen in Neuts and Lucanton (1979).

$$\text{Let } u^{(1)} = \sum_{i=1}^{m-2} ix_i + E[L].$$

$$\begin{aligned} E[L] &= x_{m-1}(m-1) + x_{m-1}mR + x_{m-1}(m+1)R^2 + \dots \\ E[L]R &= x_{m-1}(m-1)R + x_{m-1}mR^2 + \dots \\ E[L](I-R) &= x_{m-1}(m-1) + x_{m-1}R + x_{m-1}R^2 + \dots \\ E[L] &= x_{m-1}(m-1)(I-R)^{-1} + x_{m-1}R(I-R)^{-2} \\ &= x_{m-1}(I-R)^{-2}[(m-1)I - (m-2)R] \quad \# \end{aligned} \tag{B.1.1}$$

Then we derive the second moment as follows. Let $u^{(2)} = \sum_{i=1}^{m-2} i^2 x_i + E[L^2]$. Then

$$\begin{aligned} E[L^2] &= x_{m-1}(m-1)^2 + x_{m-1}m^2R + x_{m-1}(m+1)^2R^2 + \dots, \\ E[L^2]R &= x_{m-1}(m-1)^2R + x_{m-1}m^2R^2 + \dots, \\ E[L^2](I-R) &= x_{m-1}(m-1)^2 + x_{m-1}R(2m-1) + x_{m-1}R^2(2m+1) + \dots, \end{aligned}$$

Let

$$\begin{aligned} E[L^2]' &= x_{m-1}R(2m-1) + x_{m-1}R^2(2m+1) + \dots, \\ E[L^2]'R &= x_{m-1}R^2(2m-1) + x_{m-1}R^3(2m+1) + \dots, \\ E[L^2]'(I-R) &= x_{m-1}R(2m-1) + x_{m-1}R^2(2R^2)(I-R)^{-1}, \\ E[L^2]'R &= x_{m-1}R^2(2m-1) + x_{m-1}R^3(2m+1) + \dots, \end{aligned}$$

So,

$$\begin{aligned} E[L^2] &= x_{m-1}(m-1)^2(I-R)^{-1} + x_{m-1}R(2m-1)(I-R)^{-2} + x_{m-1}(2R^2)(I-R)^{-3} \\ &= x_{m-1}(I-R)^{-3}[(m-1)^2(I-R)^2 + (2m-1)R(I-R) + 2R^2]. \end{aligned}$$

B.2 Proof of proposition 1

Assume the generator of a particular server queue is as follows. All the sub-matrix is similar to (4.4) of chapter 4 with m servers.

$$Q = \begin{bmatrix} \Xi_1 & \Gamma_1 & & & \\ \Psi_2 & \Xi_2 & & & \\ & \Psi_3 & \ddots & & \\ & & & \ddots & \\ & & & & \Xi_{m+1} \end{bmatrix}.$$

Since $\pi Q = \mathbf{0}$, after eliminating the last equivalence equation, we have

$$\pi_0(G - \Gamma_1) + \pi_1\Psi_2 = 0 \quad (\text{B.2.1})$$

$$\pi_0\Gamma_1 + \pi_1(G - \Psi_2 - \Gamma_2) + \pi_2\Psi_3 = 0, \quad (\text{B.2.2})$$

\vdots

$$\pi_{m-2}\Gamma_{k-1} + \pi_{m-1}(G - \Psi_k - \Gamma_k) + \pi_m\Psi_{m+1} = 0.$$

Form (B.2.1), we have

$$\pi_0G + \pi_1\Psi_2 = \pi_0\Gamma_1. \quad (\text{B.2.3})$$

Substitute (B.2.3) into (B.2.2) and after eliminating and arrange terms we have

$$\pi_0G + \pi_1G + \pi_2\Psi_3 = \pi_1\Gamma_2.$$

Continue doing in this way, finally we get

$$\begin{aligned} & \pi_0G + \pi_1\Psi_2 + \pi_0G + \pi_1G + \pi_2\Psi_3 + \cdots + \\ & \pi_0G + \pi_0G + \pi_1G + \cdots + \pi_{m-1}G + \pi_m\Psi_{m+1}. \\ & = \pi_0\Gamma_1 + \pi_1\Gamma_2 + \cdots + \pi_{m-1}\Gamma_m. \end{aligned}$$

Multiply by unity vector of proper dimension on both sides yield

$$\begin{aligned} & \left(\sum_{i=1}^m \pi_i\Psi_{i+1} \right) \cdot \mathbf{I} + \pi_0G + \pi_0G + \pi_1G + \cdots \\ & + \pi_{m-1}G \cdot \mathbf{I} = \left(\sum_{i=0}^{m-1} \pi_i\Gamma_{i+1} \right) \cdot \mathbf{I}. \end{aligned}$$

Since $G \cdot \mathbf{I} = \mathbf{0}$ by property of CTMC, we get

$$\left(\sum_{i=1}^m \pi_i\Psi_{i+1} \right) \cdot \mathbf{I} = \left(\sum_{i=0}^{m-1} \pi_i\Gamma_{i+1} \right) \cdot \mathbf{I}.$$

Plug in the relationship

$$\Psi_i = (i-1)\zeta I, \text{ and}$$
$$\Gamma_i = \gamma J_i.$$

If $\gamma = \zeta$, from the definitions of E[O] and E[RE], we have the result.

B.3 Proof of corollary 1

The proof immediately follows from PROPOSITION 1 by rearranging terms.



Appendix C Matlab codes of algorithms

C.1 CSA

CSA contains 3 functions: CSABCAP, simannBcap, OptSABCap.

C.1.1 CSABCAP code

```
function CSABCAP
global N lamda h1 h2 fid
fid = fopen('CSAout.m','w');
% Initialize
t=cputime;
sa_t=85;
sa_rt=0.8;
sa_nt=5;
sa_ns=20;
fun_name='OptSABCap';
run=5;
s_l=0.1;
for N=4:7
    for iter=1:run
        c=cell(1,N);p=c;r=c;pai=c;
        B=zeros(1,N);W=B;I=B;h1=B;h2=B;lamda=1;delta=0.1;
        i=1:N;h1=0.5+delta*(i-2);h2=0.5*(1+delta).^(i-1);
        u=zeros(1,N);v=u;
        for i=1:N
            c{i}=zeros(i);p{i}=c{i};r{i}=zeros(1,i);pai{i}=r{i};
        end
        % Decide searching bounds
        UB1=100*ones(1,N);LB1=zeros(1,N);
        UB2=100*ones(1,N);LB2=1.01*ones(1,N);
        % Run CSA
        [xopt,fopt,SLOpt,rhopt,Wopt,Iopt,Bopt]=SimAnnBcap(fun_name, LB1, UB1,...
            LB2, UB2, sa_t, sa_rt, sa_nt, sa_ns, s_l);
        fprintf(fid,'N %d Run %d xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt\n',N,iter);
        fprintf(fid,');
        for i=1:N
            if i==N
                fprintf(fid,'%d)',xopt(i));
            else
                fprintf(fid,'%d, ',xopt(i));
            end
        end
        end
        for i=N+1:2*N
            fprintf(fid,'%0.2f ',xopt(i));
        end
        end
        fopt_1=fopt-sum(xopt(N+1:2*N));
        fprintf(fid,' %0.3f %0.3f %0.3f ',fopt,fopt_1,SLOpt);
        fprintf(fid,');
        for i=1:N
```

```

        if i==N
            fprintf(fid, '%.2f', rhopt(i));
        else
            fprintf(fid, '%.2f, ', rhopt(i));
        end
    end
    for i=2:N
        fprintf(fid, '%f', Wopt(i));
    end
    for i=1:N
        fprintf(fid, '%f', Iopt(i));
    end
    for i=1:N
        fprintf(fid, '%f', Bopt(i));
    end
    fprintf(fid, '\n');
end
end
fclose(fid);
e=cputime-t;
fprintf('operation duration: %f,e);

```

C.1.2 simannBcap code

```

function [xopt,fopt,SLOpt,rhopt,Wopt,Iopt,Bopt]=simannBcap(func, LB1, UB1, LB2, ...
    UB2, sa_t, sa_rt, sa_nt, sa_ns, s_l)
global N R fid
sa_neps=4; %number of times eps
sa_eps=1e-6; %convergence criteria
sa_maxeval=500000; %maximum number of function evaluations
sa_nacc=0; %number of acceptations
sa_nevals=0; %number of evaluations
fstar=Inf*ones(sa_neps,1); %last optimum at each sa_nt
x(1:N-1+R)=LB1(1:N-1+R)+(UB1(1:N-1+R)-LB1(1:N-1+R)).*rand(1, N-1+R);
%starting values for buffer
u(1:N-1+R)=LB2(1:N-1+R)+(UB2(1:N-1+R)-LB2(1:N-1+R)).*rand(1, N-1+R);
%starting values for capacity
[f,SL]=feval(func,round(x),u); %function evaluation with parameters x u
xopt=[round(x) u];
fopt=f;
SLOpt=SL;
fstar(1)=f;

VM1=(UB1-LB1); %maximum step size
VM2=(UB2-LB2); %maximum step size

while 1
    nup=0; %number of uphill movements
    nrej=0; %number of rejections
    nnew=0; %number of new global optimum
    ndown=0; %number of downhill movements
    lnobds=0;

```

```

nacp=zeros(N-1+R,1);

for m=1:sa_nt % loop number for each temperature VM(step adjusting) occur for each
    % loop
    for j=1:sa_ns % loop number for the purpose of adjusting step, each variable try sa_ns
        % times
        for h=1:N-1+R % for each variable
            if sa_nevals>=sa_maxeval
                disp('max function evaluations achieved')
                return
            end
            xp=x;
            up=u;
            xp(h)=x(h)+VM1(h)*(2*rand(1,1)-1.0);
            if (xp(h)<LB1(h)) | (xp(h)>UB1(h))
                xp(h)=LB1(h)+(UB1(h)-LB1(h))*rand(1,1);
            end
            up(h)=u(h)+VM2(h)*(2*rand(1,1)-1.0);
            if (up(h)<LB2(h)) | (up(h)>UB2(h))
                up(h)=LB2(h)+(UB2(h)-LB2(h))*rand(1,1);
            end
            [fp,SL,rho,W,I,B]=feval(func,round(xp), up);
            sa_nevals=sa_nevals+1;
            if (fp<=f & SL<=s_1)
                x=xp; % trial replace best sol x
                u=up; % trial replace best sol u
                f=fp;
                sa_nacc=sa_nacc+1;
                nacp(h)=nacp(h)+1;
                ndown=ndown+1;
                if fp<fopt
                    xopt=[round(xp) up];
                    fopt=fp;
                    SLopt=SL;
                    Wopt=W;
                    Iopt=I;
                    Bopt=B;
                    rhopt=rho;
                    sa_opteval=sa_nevals;
                    nnew=nnew+1;
                end
            else % function value increases
                p=exp((f-fp)/sa_t);% Metropolis' criteria
                pp=rand(1,1);
                if (pp<p & SL <=s_1)
                    x=xp;
                    u=up;
                    f=fp;
                    sa_nacc=sa_nacc+1;
                    nacp(h)=nacp(h)+1;
                    nup=nup+1;
                end
            end
        end
    end
end

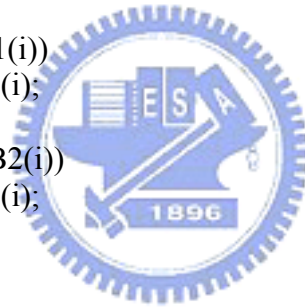
```



```

        else
            nrej=nrej+1;
        end
    end
end
end
%adjust maximal step size vm sa_ns times have passed, each variable
% hascalculatedrespective nACP(i) if ratio of accept of the variable (less function
% value) or prob. of escape is high (function value not so high) then enlarge the
% search area within the neighborhood of the variable; if ratio of accept is low then
% shrink search area within the neighborhood of the variable;
c=ones(N-1+R,1)*2;
for i=1:N-1+R
    ratio=nACP(i)/sa_ns;
    % Step length adjustment, Note if 0.4<=ratio<=0.6 no change happens
    if ratio>0.6
        VM1(i)=VM1(i) * (1+c(i)*(ratio-0.6)/0.4);
        VM2(i)=VM2(i) * (1+c(i)*(ratio-0.6)/0.4);
    elseif ratio <0.4
        VM1(i)=VM1(i) / (1+c(i)*((0.4-ratio)/0.4));
        VM2(i)=VM2(i) / (1+c(i)*((0.4-ratio)/0.4));
    end
    if VM1(i)>(UB1(i)-LB1(i))
        VM1(i)=UB1(i)-LB1(i);
    end
    if VM2(i)>(UB2(i)-LB2(i))
        VM2(i)=UB2(i)-LB2(i);
    end
end
end
for i=1:N-1+R
    nACP(i) = 0;
end
end
% check termination criteria for the current temperature, if the current optimal f (before
% changing temperature) less than global optimal (fopt) within eps, quit; notice fp (new
% function value) could become current optimal f because of Metropolis selection process
% fstar(1)=f; f(1)* is current optimum for this temperature
quit = (((fstar(1)-fopt) <= sa_eps) & (SL<=s_l));
% guarantee current optimum is global optimum
% if within 4 times of temperature reduction current optimal f and any of previous
% temporary optimal difference within eps, quit. means dwindling
if any(abs(fstar-f)>sa_eps)
    quit=0;
end
if quit
    disp(['simulated annealing achieved termination after ', num2str(sa_nevals),' evals']);
    return
end
sa_t=sa_t*sa_rt;% reduce temperature
fstar(2:4)=fstar(1:3);
% continue from current optimum

```




```

x=xopt(1:N-1+R);
u=xopt(N+R:2*(N-1+R));
f=fopt;
end %while

```

C.1.3 OptSABCap code

```

function [TC,SL,rho,W,I,B]=OptSABCap(x,u)
% Performance evaluation (Object function) for constrained SA (CSA)
global N lamda h1 h2
N=4;delta=0.1;i=1:N;lamda=1;h1=0.5+delta*(i-2);h2=0.5*(1+delta).^(i-1);
v=u-lamda;
c{1}=[-v(1)];r{1}=[1];T=0;
for i=2:N
    c{i}=[c{i-1} zeros(i-1,1)-sum(c{i-1},2);zeros(1,i-1) -v(i)];W(i)=lamda/v(i);
end
for i=1:N
    p{i}=lamda*inv(lamda*eye(i)-c{i});
end
pai{1}=r{1}*p{1};
for i=2:N
    r{i}=[r{i-1}*p{i-1}^x(i-1) 1-r{i-1}*p{i-1}^x(i-1)*ones(i-1,1)];pai{i}=r{i}*p{i};
end
for i=1:N
B(i)=pai{i}*p{i}^x(i)*inv(eye(i)-p{i})*ones(i,1);I(i) = x(i) - pai{i}*inv(eye(i) - p{i})* ...
    ones(i,1) + B(i);
end
SL=r{N}*p{N}^x(N)*expm(c{N}*T)*ones(N,1);
WIP=sum(I+W)-I(N);
TC=sum(h1.*W+h2.*I)+10*B(N)+sum(u);
rho=1./u;

```

C.2 CGA

CGA contains 13 functions: CGA, initialize_r0, validate_r0, GeneticAlgorithm, doHomomorphMap, SimpleCrossover, ArithmeticCrossover, Xmutation, fineBoundaryResolution, delta, deltaInverse, sum_tSegments, Optga.

C.2.1 CGA code

```

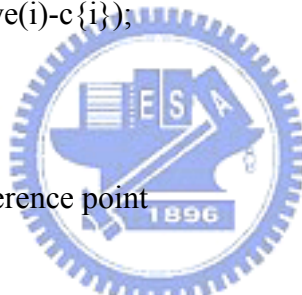
function CGA
global nG nPi nP nSC nAC nVar bounds ctr Pc Pm fid FINE_RESOLUTION ...
COARSE_RESOLUTION xmapped Xdummy r c p pai W h1 h2
% GA parameters
run = 5;
nG = 100; % total number of generation
nPi = 24; % population size (Fixed)
nP = 12; % Mating pool size
nSC = 6; % number of simple crossover - each yields 2 children
nAC = 6; % number of Arithmetic crossover - each yields 2 children
Pc = 0.8; % probability of crossover
Pm = 0.01; % probability of mutation
MAP_RESOLUTION = 5.0e-5;

```

```

FINE_RESOLUTION = 8;% feasible space searching
COARSE_RESOLUTION = 15; %feasible space searching
fid = fopen('CGAout.m','w');
for nVar=4:7
bounds=repmat([0 30],1,nVar);
% Initialize
c=cell(1,nVar);p=c;r=c;pai=c;t=cputime;
B=zeros(1,nVar);W=B;I=B;h1=B;h2=B;lamda=1;WIP=0;delta=0.1;
i=1:nVar;h1=0.5+delta*(i-2);h2=0.5*(1+delta).^(i-1);
u(i)=1.25;v(i)=u(i)-lamda;
for i=1:nVar
    c{i}=zeros(i);p{i}=c{i};r{i}=zeros(1,i);pai{i}=r{i};
end
Xdummy=zeros(nPi,nVar);xmapped=Xdummy;
XA=zeros(nP,nVar);NM=zeros(nPi,nVar);
% Compute c, p of L & Z '92
c{1}=[-v(1)];
for i=2:nVar
    c{i}=[c{i-1} zeros(i-1,1)-sum(c{i-1},2);zeros(1,i-1) -v(i)];W(i)=lamda/v(i);
end
for i=1:nVar
    p{i}=lamda*inv(lamda*eye(i)-c{i});
end
r{1}=[1];
pai{1}=r{1}*p{1};
for iter = 1:run% random runs
% Initial generated feasible reference point
[ret,r0]=initialize_r0;
if (ret==1)
    fprintf('initial feasible reference point : ');
    for i=1:nVar
        fprintf('%f',r0(i));
    end
    fprintf('generated\n');
end
% initial random population
x = -1 + rand(nPi,nVar) * 2;
% HomomorphMapping
[xmapped]=doHomomorphMap(x,r0);
[fval] = Optga(xmapped);
[Yvect,Ip]=sort(fval);
% Initial populated population
Xgen=zeros(nPi,nVar);Fgen=zeros(nPi);
for i = 1:nPi
    Xgen(i,:)=x(Ip(i),:);
    Fgen(i)=Yvect(i);
end
Xbest=Xgen(1,:);
Fbest=Fgen(1);
fprintf(fid,'Run %d -----GA initial-----\n',iter);
fprintf(fid,'Fbest: %f\n',Fbest);

```



```

ctr = 0; % Initialize iteration counter
while (ctr~==nG)
    ctr = ctr + 1;
    [x,f,Xgen,Fgen]=GeneticAlgorithm(@Optga,Xgen,Fgen,r0);
    fval(ctr) = f;
    % Preserving the best feasible solution
    if (fval(ctr) < Fbest)
        Fbest=fval(ctr);
        Xbest=round(x);
        fprintf(fid,'Generation:%d, Current best value: %f\n',ctr,fval(ctr));
    end
end
[invalid,nlineq] = validate_r0(Xbest);
fprintf(fid,'N: %d Run: %d -----GA results-----\n',nVar,iter);
fprintf(fid,'Xbest: ');
for i=1:nVar
    fprintf(fid,'%d ',Xbest(i));
end
fprintf(fid,'\n');
fprintf(fid,'Fbest: %f\n',Fbest);
fprintf(fid,'SL: %f\n',nlineq+0.1);
end
end
fclose(fid);
e=cputime-t;
fprintf(1,'operation duration: %f,e);

```



C.2.2 initialize_r0 code

```

function [ret,r0]=initialize_r0
global nVar bounds
TRIES=10000;
for k = 0:99
    for count = 1:10 * TRIES
        for i = 1:nVar
            r0(i) = rand(1,1)*(bounds(2*i)- bounds(2*i-1)) + bounds(2*i-1);
        end
        [invalid] = validate_r0(r0);
        if(~invalid)
            fprintf('Valid reference point found!!\n');
            fprintf('Starting CGA...\n\n');
            ret=1;
            return
        end
    end
end
end
fprintf('Cannot find valid reference point.\n');
ret=0;

```

C.2.3 validate_r0 code

```

function [invalid,nlineq] = validate_r0(x)
global r c p pai W nVar h1 h2

```

```

x=round(x);invalid = 0;
for i=2:nVar
    r{i}=[r{i-1}*p{i-1}^x(i-1) 1-r{i-1}*p{i-1}^x(i-1)*ones(i-1,1)];pai{i}=r{i}*p{i};
end
for i=1:nVar
B(i)=pai{i}*p{i}^x(i)*inv(eye(i) - p{i})*ones(i,1); I(i) = x(i) - pai{i}*inv(eye(i) - ...
    p{i})*ones(i,1) + B(i);
end
i=1:nVar;
%SL=r{nVar}*p{nVar}^x(nVar)*expm(c{nVar}*0)*ones(nVar,1);
nlineq(1)=r{nVar}*p{nVar}^x(nVar)*expm(c{nVar}*0)*ones(nVar,1) - 0.1;
for i = 1:1
    if( nlineq(i) > 0.0 )
        invalid = 1;
        return
    end
end
end

```

C.2.4 GeneticAlgorithm code

```

function [XCurbest,FCurbest,Xgen,Fgen]=GeneticAlgorithm(objfun,Xgen,Fgen,r0)
% select top 50% populations to propogate to the next generation,
% Offspring replace the top 50% populations for each crossover operator, then
% non-uniform mutation for whole population
global nG nPi nP nSC nAC nDC nI nVar bounds fid
% Record best chromosome in the beginning of each generation;
XBest=Xgen(1,:);
Best=Fgen(1);
% Operator --- SimpleCrossover, ArithmeticCrossover.
[XSChild] = SimpleCrossover(Xgen);
[XAChild] = ArithmeticCrossover(Xgen);
i=1:2*nAC;
Xgen(i,:)=XAChild(i,:);
Xgen(2*nAC+i,:)=XSChild(i,:);
[XNMu]= XMutation(Xgen);
Xgen = XNMu;
[n m] = size(Xgen);
[xmapped]=doHomomorphMap(Xgen,r0);
[fval] = Optga(xmapped);
Xdummy=Xgen;
[Yvect,Ip]=sort(fval);
XCurbest=xmapped(Ip(1),:);
FCurbest=Yvect(1);
clear Xgen fval% clear memory for next generation
Xgen=zeros(nP,nVar);
i = 1:nP;
% Elitist (if current best not superior to last generation, replace the last chromosome w/ the
% last best)
if (FCurbest<Best)
    Xgen(i,:)=Xdummy(Ip(i),:);
    Fgen(i)=Yvect(i);
else

```

```

    Xgen(i,:)=Xdummy(Ip(i),:);
    Fgen(i)=Yvect(i);
    Xgen(nP,:)=XBest;
    Fgen(nP)=Best;
End

```

C.2.5 doHomomorphMap code

```

function [xmapped]=doHomomorphMap(x,r0)
global nVar nPi bounds MAP_RESOLUTION COARSE_RESOLUTION fid
[n,m]=size(x);
infeasible = 0;
j=1:n;
ymax(j)=max(abs(x),[],2);
% find sVect
y=zeros(nPi,nVar);
for i=1:nVar
    y(:,i)=x(:,i)./ymax';
end
i=1:nVar;
s = repmat(1/2*(bounds(2*i) + bounds(2*I - 1)),nPi,1) + repmat( 1/2*(bounds(2*i) - ...
    bounds(2*i-1)),nPi,1).*y;
s_r0=s-repmat(r0,nPi,1);
for j=1:nPi
% Get tVect
tCount = 1;
for i = 1:COARSE_RESOLUTION
    tempVect = r0 + s_r0(j,:)*(i / COARSE_RESOLUTION);
    [invalid,nlineq] = validate_r0(tempVect);
    if(invalid ~= infeasible)
        infeasible = invalid;
        tCount = tCount + 1;
    end
end
oddCount = mod(tCount,2);
lastIsInvalid = invalid;
[invalid,nlineq] = validate_r0(s(j,:));
boundaryIsInvalid = invalid;
% Populate regions
tVect(1) = 0.0;infeasible = 0;tCount = 2;
for i = 1:COARSE_RESOLUTION
    tempVect = r0 + s_r0(j,:)*(i / COARSE_RESOLUTION);
    [invalid,nlineq] = validate_r0(tempVect);
    if(invalid ~= infeasible)
        [tVect(tCount)] = fineBoundaryResolution(s_r0(j,:),i,r0,infeasible);
        tCount = tCount + 1;
        infeasible = invalid;
    end
end
end
if(oddCount & (~lastIsInvalid) & (~boundaryIsInvalid))
    tVect(tCount) = 1.0;
elseif(oddCount & (~lastIsInvalid) & boundaryIsInvalid)

```

```

tVect(tCount) = fineBoundaryResolution(s_r0(j,:), COARSE_RESOLUTION, r0, ...
    infeasible);
elseif((~oddCount) & lastIsInvalid & (~boundaryIsInvalid))
    [tVect(tCount)]=fineBoundaryResolution(s_r0(j,:),COARSE_RESOLUTION,r0, ...
        infeasible);
    tCount = tCount + 1;
    tVect(tCount) = 1.0;
else
    tCount = tCount -1;
end
% Do mapping!!
[deltaInvVal] = deltaInverse(y_max(j), tVect, tCount);
xmapped(j,:) = r0 + s_r0(j,:)*deltaInvVal;
end

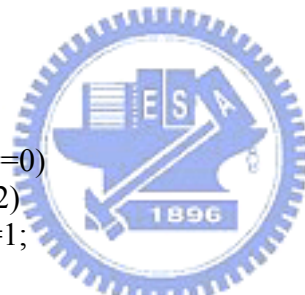
```

C.2.6 SimpleCrossover code

```

function [XS] = SimpleCrossover(X)
% creating children by simple 1-point crossover
global nVar nSC Pc
first=0;one=0;
XS = X;
for i=1:2*nSC
    r = rand(1,1);
    if (r<Pc)
        first=first+1;
        if (mod(first,2)==0)
            if (nVar==2)
                point=1;
            else
                point=floor(nVar*rand(1,1));
            end
            for j=1:point
                XS(one,j) = X(i,j);
                XS(i,j) = X(one,j);
            end
        else
            one=i;
        end
    end
end
end

```



C.2.7 ArithmeticCrossover code

```

function [XA] = ArithmeticCrossover(X)
% creating children by whole Arithmetic crossover
global nVar Pc nAC
XA=X;
first=0;one=0;
[n m] =size(X);
for j=1:2*nAC
    r = rand(1,1);
    if (r<Pc)

```

```

        first=first+1;
        if (mod(first,2)==0)
            r1 = rand(1,1);
            r2 = 1 - r1;
            for i=1:nVar
                XA(one,i) = r1*X(one,i) + r2*X(j,i);
                XA(j,i) = r2*X(one,i) + r1*X(j,i);
            end
        else
            one=j;
        end
    end
end
end

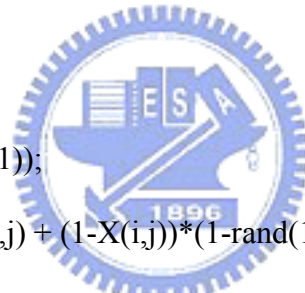
```

C.2.8 Xmutation code

```

function [NM] = XMutation(X)
% non-uniform whole mutation
global ctr nG bounds Pm
NM=X;
b=6;
[n m] =size(X);
for i = 1:n
    r1=rand(1,1);
    if(r1<=Pm)
        for j = 1:m
            r2 = round(rand(1,1));
            if (r2==0)
                NM(i,j) = X(i,j) + (1-X(i,j))*(1-rand(1,1)^((1-ctr/nG)^b));
            else
                NM(i,j) = X(i,j) - (X(i,j)-(-1))*(1-rand(1,1)^((1-ctr/nG)^b));
            end
        end
    end
end
end
end

```



C.2.9 fineBoundaryResolution code

```

function [tVect] = fineBoundaryResolution(s_minus_r0,i,r0,infeasible)
% This routine implements binary search within each boundary of feasible region
global MAP_RESOLUTION FINE_RESOLUTION COARSE_RESOLUTION
upper = i;
lower = i - 1.0;
for i = 1:FINE_RESOLUTION
    middle = (upper + lower) / 2.0;
    %if( ((upper - middle) < MAP_RESOLUTION) | ((middle - lower) <
    %MAP_RESOLUTION) )
    %break;
    %end
    [invalid,nlineq] = validate_r0(r0+s_minus_r0*(middle / COARSE_RESOLUTION));
    if(invalid ~= infeasible)
        upper = middle;
    else

```

```

        lower = middle;
    end
end
if (infeasible)
    tVect = upper / COARSE_RESOLUTION;
else
    tVect = lower / COARSE_RESOLUTION;
end

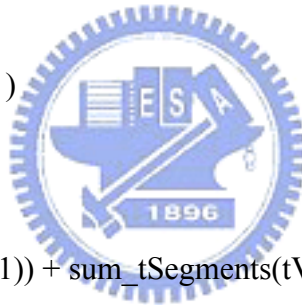
```

C.2.10 delta code

```

function [delt] = delta(t, tVect, tCount)
% t is assumed to be 'feasible'
[d] = sum_tSegments(tVect, tCount / 2);
if(d == 0.0)
    delt = 0.0;
    return
end
% find min subscript
subscript = 1;
while(1)
    if( (subscript * 2) >= tCount )
        break
    end
    if( tVect(2*subscript) >= t )
        break
    end
    subscript = subscript + 1;
end
delt = ( (t - tVect(2*subscript - 1)) + sum_tSegments(tVect, subscript - 1) ) / d;

```



C.2.11 deltaInverse code

```

function [deltaInvVal] = deltaInverse(a, tVect, tCount)
[d] = sum_tSegments(tVect, tCount / 2);
if(d == 0.0)
    deltaInvVal = 0.0;
    return
end
% find min subscript
subscript = 1;
while(1)
    if( (subscript * 2) >= tCount )
        break
    end
    [delt] = delta( tVect(2*subscript), tVect, tCount );
    if( delt >= a )
        break
    end
    subscript = subscript + 1;
end
tHi = tVect(2*subscript);
tLow = tVect(2*subscript - 1);

```



```

[delta_tHi] = delta(tHi, tVect, tCount);
[delta_tLow] = delta(tLow, tVect, tCount);
denom = delta_tHi - delta_tLow;
if(denom <= 0.0)
    deltaInvVal = tHi;
    return
end
numer = a - delta_tLow;
if(numer <= 0.0)
    deltaInvVal = tLow;
    return
end
if(a >= delta_tHi)
    deltaInvVal = tLow;
    return
end
dj = tHi - tLow;
deltaInvVal = tLow + ( dj * (a - delta_tLow) / denom );

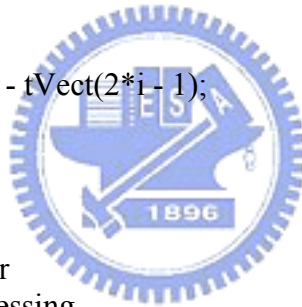
```

C.2.12 sum_tSegments code

```

function [total] = sum_tSegments(tVect, qtySegs)
total = 0;
for i = 1:qtySegs
    total = total + tVect(2*i) - tVect(2*i - 1);
end

```



C.2.13 Optga code

```

function [fopt,SL]=Optga(x)
global r c p pai W N h1 h2 nVar
% This function use batch processing
% Performance evaluation (Object function) for constrained GA (CGA)
x=round(x);
[n,m]=size(x);
for j=1:n
    for i=2:nVar
        r{i}=[r{i-1}*p{i-1}^x(j,i-1) 1-r{i-1}*p{i-1}^x(j,i-1)*ones(i-1,1)];pai{i}=r{i}*p{i};
    end
    for i=1:nVar
        B(i)=pai{i}*p{i}^x(j,i)*inv(eye(i)-p{i})*ones(i,1);I(i)=x(j,i)-pai{i}*inv(eye(i) - ...
            p{i})*ones(i,1)+B(i);
    end
    fopt(j)=sum(h1.*W+h2.*I)+10*B(nVar);
    SL=r{nVar}*p{nVar}^x(nVar)*expm(c{nVar}*0)*ones(nVar,1);
End

```

C.3 ENU

ENU contains 1 functions: OptEnumLarge.

C.3.1 OptEnumLarge code

```

function [xbest, fbest, SLbest, Bbest, Ibest]=OptEnumLarge

```

```

clear all
global N r c p pai T s_1 W h1 h2 b co u fbest xbest SLbest Bbest Ibest
% Initialize
t=cputime;N=4;b=1;co=1;
c=cell(1,N);p=c;r=c;pai=c;
B=zeros(1,N);W=B;I=B;h1=B;h2=B;lamda=1;WIP=0;delta=0.1;xbar=zeros(1,N-1);
i=1:N;h1=0.5+delta*(i-2);h2=0.5*(1+delta).^(i-1);
h1=[0.5 0.5 0.5 0.5];h2=[0.5 0.5 0.5 1];
u(i)=2.5;v(i)=u(i)-lamda;
for i=1:N
    c{i}=zeros(i);p{i}=c{i};r{i}=zeros(1,i);pai{i}=r{i};
end
% Compute c, p of L & Z '92
c{1}=[-v(1)];
for i=2:N
    c{i}=[c{i-1} zeros(i-1,1)-sum(c{i-1},2);zeros(1,i-1) -v(i)];W(i)=lamda/v(i);
end
for i=1:N
    p{i}=lamda*inv(lamda*eye(i)-c{i});
end
% Initialize r, pai of L & Z '92
r{1}=[1];
pai{1}=r{1}*p{1};
% Initialize xbest, fbest; Assign allowable waiting time and SL
s_1=0.15;
%for s_1=0.1:0.1:0.8
x=zeros(1,N);T=0;SLbest=1.0;
% Find minimum total stock required (st) to achieve SL
for s=0:100
    x(N)=s;
    for i=2:N
        r{i}=[r{i-1}*p{i-1}^x(i-1) 1-r{i-1}*p{i-1}^x(i-1)*ones(i-1,1)];pai{i}=r{i}*p{i};
    end
    for i=1:N
        B(i)=pai{i}*p{i}^x(i)*inv(eye(i)-p{i})*ones(i,1);I(i)=x(i)-pai{i}*inv(eye(i)-p{i})* ...
            ones(i,1)+B(i);
    end
    SL=r{N}*p{N}^x(N)*expm(c{N}*T)*ones(N,1);
    if SL<=s_1
        st=x(N);
        SLbest=SL;
        break
    end
end
if SL > s_1
    fprintf('Can not find ST in 100, Enlarge search space');
end
% Reset design variable
x=zeros(1,N);xbest=x;fbest=inf;
% Call optimization procedure
[xbest, fbest, SLbest, Bbest, Ibest]=opt(st,x,1);

```



```

%end
e=cputime-t;
fprintf('operation duration: %f',e);
% Enumeration starts here
function [xbest, fbest, SLbest, Bbest, lbest]=opt(st,x,k)
global r p pai c T N W s_1 b co u fbest xbest SLbest Bbest lbest
x(k)=0;
% Find searching bound for each stage
for s=1:100
    x(k)=s;
    if k==1
        x(N)=st-x(k);
    else
        j=1:k;
        x(N)=st-sum(x(j));
    end
    for i=2:N
        r{i}=[r{i-1}*p{i-1}^x(i-1) 1-r{i-1}*p{i-1}^x(i-1)*ones(i-1,1)];pai{i}=r{i}*p{i};
    end
    SL=r{N}*p{N}^x(N)*expm(c{N}*T)*ones(N,1);
    if SL>s_1
        xbar(k)=x(k)-1;
        break
    end
end
x(k)=0;
while (x(k)<=xbar(k))
    if k<N-1
        opt(st,x,k+1);% Recursive call itself when not reaching stage J-1, J
    else
        x(N)=st-sum(x(1:N-1));
        [TC,Bak,lend]=f(x);
        if TC<fbest
            SLbest=r{N}*p{N}^x(N)*expm(c{N}*T)*ones(N,1);
            fbest=TC;
            xbest=x;
            Bbest=Bak;
            lbest=lend;
        end
    end
    x(k)=x(k)+1;
end
% Performance evaluation
function [TC,Bak,lend]=f(x)
global N r c p pai T s_1 W h1 h2 b co u Bbest lbest
for i=2:N
    r{i}=[r{i-1}*p{i-1}^x(i-1) 1-r{i-1}*p{i-1}^x(i-1)*ones(i-1,1)];pai{i}=r{i}*p{i};
end
for i=1:N
    B(i)=pai{i}*p{i}^x(i)*inv(eye(i)-p{i})*ones(i,1);I(i)=x(i)-pai{i}*inv(eye(i)-p{i})*...
        ones(i,1)+B(i);

```



end

```
Bak=B(N);lend=I;  
TC=sum(h1.*W+h2.*I)+b*B(N)+sum(co*u);
```

C.4 Typical results of solving BAP and CAP simultaneously for J = 4 and 5 using CSA

```
N 4 Run 1 xopt, fopt, SLOpt, rhopt  
0 1 1 6 1.98 2.15 2.20 2.25 14.254 0.082 0.51 0.47 0.45 0.44  
N 4 Run 2 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 0 2 6 1.98 2.15 2.18 2.24 14.251 0.081 0.50 0.47 0.46 0.45  
N 4 Run 3 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 0 2 6 1.98 2.15 2.18 2.24 14.251 0.081 0.50 0.47 0.46 0.45  
N 4 Run 4 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 1 1 6 1.98 2.15 2.20 2.25 14.254 0.082 0.51 0.47 0.45 0.44  
N 4 Run 5 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 1 1 6 1.98 2.15 2.20 2.25 14.254 0.082 0.51 0.47 0.45 0.44  
N 5 Run 1 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 1 1 1 6 1.97 2.14 2.20 2.25 2.29 17.800 0.091 0.51 0.47 0.46 0.44 0.44  
N 5 Run 2 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 0 1 1 7 1.98 2.15 2.18 2.22 2.26 17.806 0.087 0.51 0.47 0.46 0.45 0.44  
N 5 Run 3 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 0 2 2 5 1.97 2.14 2.18 2.24 2.33 17.795 0.093 0.51 0.47 0.46 0.45 0.43  
N 5 Run 4 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 0 1 3 5 1.97 2.14 2.18 2.22 2.32 17.797 0.092 0.51 0.47 0.46 0.45 0.43  
N 5 Run 5 xopt, fopt, SLOpt, rhopt, Wopt, Iopt, Bopt  
0 0 1 0 8 1.98 2.15 2.18 2.22 2.25 17.815 0.087 0.51 0.47 0.46 0.45 0.44
```



References

- 1 Ajay S. and J. M. Smith, 1997. Buffer allocation for an integer nonlinear network design problem, *Computers & Operations Research*, 24 (5) 453-472.
- 2 Abboud N. E., 2001. A discrete-time markov production-inventory model with machine breakdowns, *Computers & Industrial Engineering*, 39, 95-107.
- 3 Alexander Ost, 2001. *Performance of communications: A model-based approach with Matrix-Geometric Methods*, Springer-Verlag, Berlin Heidelberg, Germany.
- 4 Arns, M., M. Fischer, P. Camper and T. Camper, 2002. Supply chain modeling and its analytic evaluation, *Journal of the Operational Research Society*, 53 (2002) 885-894.
- 5 Axsäter, S., 1993a. Exact and approximate evaluation of batch-ordering policies for two-level inventory systems, *Operations Research*, 41 (4) 777-785.
- 6 Axsäter, S., 1993b. Continuous review policies for multi-level inventory systems with stochastic demand. Chapter 4 in S.C. Graves et al. (eds.), *Logistics of Production and Inventory*, Handbooks in OR and MS, 4, pp. 175-197, Elsevier (North-Holland), Amsterdam.
- 7 Axsäter, S., 2000a. *Inventory Control*, Kluwer Academic Publishers, Norwell.
- 8 Axsäter, S., 2000b. Exact analysis of continuous review (r, q) policies in two-echelon inventory systems with compound Poisson demand, *Operations Research*, 48 (5) 686-696.
- 9 Boyaci T. and G. Gallego, 2001. Serial Production/Distribution Systems Under Service Constraints, *Manufacturing & Service Operations Management*, 3 (1) 43–50.
- 10 Buzacott, J. A. and J. G. Shanthikumar, 1993. *Stochastic Models of Manufacturing Systems*, Prentice Hall.
- 11 Ching, W. K., 2001. Markovian Approximation for Manufacturing Systems of Unreliable Machines in Tandem, *Naval Research Logistics*, Vol. 48 (2001) 65-78.
- 12 Cinlar, E., 1975. *Introduction to Stochastic Processes*, Prentice Hall Inc., Englewood Cliffs, NJ.
- 13 Duri C., Y. Frein, M. Di Mascolo, 2000. Performance evaluation and design of base stock systems, *European Journal of Operational Research*, 127 (2000) 172-188.
- 14 Enginarlar E., J. Li, S. M. Meerkov and R. Q. Zhang, 2002. Buffer capacity for accommodating machine downtime in serial production lines, *International Journal of Production Research*, 40 (3) 601-624.
- 15 Feldman, R. M., and C. V. F., 1995. *Applied Probability & Stochastic Processes*, PWS

Publishing Co., Boston.

- 16 Goffe, F. and Rogers, 1994. Global Optimization of Statistical Functions with Simulated Annealing, *Journal of Econometrics*, 60 (1/2) 65-100.
- 17 Gupta, D. and N. Selvaraju, 2004. *Stock Positioning in Capacitated Serial Supply Systems*, Working paper, Supply Chain and Operations Research Laboratory, Department of Mechanical Engineering, University of Minnesota.
- 18 Gurgur, G. Z., 2002. *Performance Analysis and Capacity Planning of A Multi-stage, Multi-product, Decentralized and Market-driven Manufacturing Systems*, Ph. D. Dissertation, Graduate School-New Brunswick Rutgers, The State University of New Jersey.
- 19 Jackson, J. R., 1957. Networks of waiting lines. *Operations Research*, 5, 518-521.
- 20 Jackson, J. R., 1963. Jobshop-like queueing systems, *Management Science*, 10, 131-142.
- 21 Kalpakam, S. and K. P. Sapna, 1997. A lost sales inventory system with supply uncertainty, *Computers & Mathematics with Applications*, 33 (3) 81-93.
- 22 Kao, E. P. C., 1997. *An Introduction to Stochastic Processes*, Duxbury Press, Belmont, MA.
- 23 Kelton, W. D., R. P. Sadowski, D. A. Sadowski, 2002. *Simulation with Arena*, McGraw-Hill companies, Inc., US.
- 24 Koziel S. and Z. Michalewicz, 1998. *A Decoder-based Evolutionary Algorithm for Constraint Parameter Optimization Problems*, Proceedings of the 5th conference on parallel problems solving from nature, Springer Verlag, N. Y.
- 25 Latouche G., V. Ramaswami, 1999. *Introduction to Matrix Analytic Methods in Stochastic Modeling*, ASA-SIAM, US.
- 26 Lee, Y. J. and P. H. Zipkin, 1992. Tandem queues with planned inventories, *Operations Research*, 40 (5) 936-947.
- 27 Lee, Y. J. and P. H. Zipkin, 1995. Processing networks with inventories: sequential refinement systems. *Operations Research*, 43 (6) 1025-1036.
- 28 Liu, C. M. and C. L. Lin, 1994. Performance evaluation of unbalanced serial production lines, *International Journal of Production Research*, 32 (12) 2897-2914.
- 29 Liu, L., X. Liu, and D. D. Yao, 2004. Analysis and Optimization of a Multistage Inventory-Queue System, *Management Science*, 50 (3) 365 – 380.
- 30 Mahmut, P. and D. Perry, 1995. Analysis of a (Q, R, T) inventory policy with deterministic and random yields when future supply is uncertain, *European Journal of*

- Operational Research*, 84, 431-4434.
- 31 Michalewicz, Z., 1999. *Genetic Algorithms + Data Structures = Evolution Programs*, third, revised and extended edition, Springer, New York.
 - 32 Mohebbi, E., 2003. Supply interruptions in a lost-sales inventory system with random lead time, *Computers & Operations Research*, 30, 411-426.
 - 33 Neuts, M. F, and D. M. Lucanton, 1979. A Markovian queue with servers subject to breakdowns and repairs, *Management Science*, 25 (9) 849-861.
 - 34 Neuts, M. F, 1994. *Matrix-geometric Solutions in Stochastic Models*, Dover Publication Inc., New York.
 - 35 Perros, H. G. 1994, *Queueing Networks with blocking*, Oxford University Press Inc., New York.
 - 36 Pham, D. T. and D. Karaboga, 2000. *Intelligent Optimization Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, Springer, London.
 - 37 Raghavan, N. R. S. and N. Viswanadham, 2001. Generalized queueing network analysis of integrated supply chains, *Int. J. Prod. Res.*, 39(2) 205-224.
 - 38 Sherbrooke, C. C., 1992. *Optimal Inventory Modeling of Systems*, John Wiley & Sons, Inc., New York.
 - 39 Spinellis, D., C. Papadopoulos, and J. M. Smith, 2000. Large production optimization using simulated annealing, *International Journal of Production Research*, 38(3) 509-541.
 - 40 Svoronos, A. and P. H. Zipkin, 1988. Estimating the performance of multi-level inventory systems, *Operations Research*, 36 (1) 57-72.
 - 41 Svoronos, A. and P. H. Zipkin, 1991. Evaluation of one-for-one replenishment policies for multi-echelon inventory systems, *Management Science*, 37, 68-83.
 - 42 Taylor, H. M., and S. Karlin, 1994. *An Introduction to Stochastic Modeling*, revised edition, Academic Press Inc., San Diego.
 - 43 Tayur, S., Ganeshan R., and Magazine M. 1999. *Quantitative Models for Supply Chain Management*, Tayur et al. (eds.), Kluwer Academic Publishers, Boston.
 - 44 Tee, Y. S. and M. D. Rossetti, 2002. A robust study of a multi-echelon inventory model via Simulation, *International Journal of Production Economics*, 80, 265-277.
 - 45 Thomas, D. J., P. Griffin, 1996. Coordinated Supply Chain Management, *European Journal of Operational Research*, 94 (1996) 1-15.
 - 46 Trivedi, K. S., 2002. *Probability and Statistics with Reliability, Queueing and*

Computer Science Applications, John Wiley & Sons Inc., New York.

- 47 Wang, F. F., 1993. *GI/G/c with Priority Queueing System Study: Impact of Maintenance Job*, Master's project, Dept. of Industrial Engineering, Texas A & M University, College station, TX.
- 48 Yokoyama, M., 2002. Integrated Optimization of Inventory-Distribution Systems by Random Meta-Heuristic Methods and a Genetic Algorithm. *Computers & Industrial Engineering*, 42, 175-188.
- 49 Zipkin, P. H., 1988. The use of phase-type distributions in inventory-control models, *Naval Research Logistics*, 35, 247-257.
- 50 Zipkin, P. H., 1995. Processing networks with planned inventories: tandem queues with feedback, *European Journal of Operational Research*, 80 (1995) 344-349.
- 51 Zipkin, P. H., 2000. *Foundations of Inventory Management*, McGraw-Hill, New York.

