Hongze Ma

An Integrated Methodology for Design of Distribution Chain

PhD-thesis 2003:96

Faculty of Engineering Science and Technology
Narvik University College

NTNU

URN:NBN:no-3444

An Integrated Methodology for Design of Distribution Chain

Hongze Ma

Narvik University College
Narvik, Norway

Submitted to the Norwegian University of Science and Technology (NTNU) in partial fulfillment of the requirements for a Doctor of Engineering Degree

Narvik, 2003

URN:NBN:no-3444

PREFACE

This research is carried out during September 1999 and September 2003, and this dissertation is presented at Norwegian University of Science and Technology, Norway in fulfillment of the requirements for a Doctor of Engineering Degree.

First of all, I would like to express my cordial gratitude to my supervisors Professor Ziqiong Deng, Professor Øyvind Bjørke and Professor Kesheng Wang for their guidance and encouragement in the course of this research work.

My special thanks also go to Associate Professors Arne Lakså, Per Åge Ljunggren, and Børre Bang for their valuable advice and kind help.

Further, I would like to thank my colleagues and friends at Narvik University College: Dr. Reggie Davidrajuh, Dr. Wei Deng Solvang and Dr. Bjørn Solvang for their valuable advice, discussions and friendship.

Finally, my warmest thanks go to my wife Yanqin Zhang for her love, encouragement and patience.

Hongze Ma
February 2003
Narvik, Norway

URN:NBN:no-3444

ABSTRACT

In today's buyer market, the key question for an enterprise is how to sell products rather than how to produce products. That is why the study on distribution part of a supply chain is attracting extensive attentions from both academics and industry. In this dissertation, an integrated methodology is developed to design a distribution chain. According to this methodology, a distribution chain is designed by following three phases:

(1) Problem formulation phase. In this phase, the present situation for the host enterprise is analyzed, and the goal to design this distribution chain is set. By this analysis, the objective and constraints for designing the distribution chain are determined.

(2) System design phase. In this phase, first, all possible distributors are evaluated through a trilogy:

- Determine the factors needed to be considered when evaluating a possible distributor.
- Collect data from geographically distributed distributors by a mobile agent based information acquisition system.
- Evaluate possible distributors quantitatively by a FL (Fuzzy Logic)-ABL (Array Based Logic) inference engine.

After evaluation, a set of eligible distributors are selected as candidates for designing this distribution chain.

With these candidates at hand, a set of models, formulae and algorithms are developed to design a distribution chain. To determine the exact customer demand at each retailer (candidate), an ANN (Artificial Neural Network) model is developed to estimate the retailer's market share in its customer zone. By this estimated market share, the customer demands at retailers are determined, and the configuration of a distribution chain, including the number and location of distributors, is determined by MIP (Mixed Integer Programming) model. The inventory control parameters at each node of this distribution chain are optimized by probability theory, and routes for vehicles to deliver products between different nodes are optimized by genetic algorithm. After this, the designing process for a distribution chain is finished.

(3) Performance evaluation phase. To verify the design result, a new form of Petri net, combinatorial Petri net, is developed, and the performance of the distribution chain designed above is evaluated by this newly developed Petri net form. If the performance is not satisfactory, the distribution chain needs to be re-designed.

All models, algorithms and formulae used in this dissertation have been implemented by computer applications. This gives possibility to realize automatic design of a

distribution chain. At last, a numerical example is given to illustrate how to apply this methodology in practice.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYM

ABC	Activity Based Costing
ABL	Array Based Logic
AHP	Analytical Hierarchical Process
ANN	Artificial Neural Network
API	Application Programming Interface
BLPC	British Library Public Catalogue
COA	Center Of Area
COS	Center Of Sums
CVRP	Capacitated Vehicle Routing Problem
DEDS	Discrete Event Dynamic System
DOM	Document Object Model
DTD	Data Type Definition
FL	Fuzzy Logic
GMP	Generalized Modus Ponens
GMT	Generalized Modus Tollens
GUI	Graphical User Interface
HTML	HyperText Markup Language
IP	Internet Protocol
ISI	Institute of Scientific Information
ISP	Integer Stochastic Programming
MCI	Multiplicative Competitive Interaction
MIP	Mixed Integer Programming
MNL	Multinomial Logit
MOM	Mean of Maxima
OCLC	Online Computer Library Center
PDF	Probability Density Function
SABL	Structural Array Based Logic
SAX	Simple API for XML
SMNL	Switching Multinomial Logit
SMTP	Simple Mail Transfer Protocol
TSP	Travelling Salesman Problem
URL	Uniform Resource Locator
VRI	Vehicle Releasing Indicator
XML	eXtensible Markup Languag

URN:NBN:no-3444

CHAPTER 1 INTRODUCTION

1.1 Research Motivation

In today's buyer market, the enterprises that can win the market will win the competition. That is why the study on market has aroused extensive interests in both academics and industry. Under such circumstance, for an enterprise, the question on how to sell its products is becoming more and more important. So the study on the parts that are closely related to market is becoming the main concern for decision makers. But, as indicated in chapter 2, compared with their significance, the research on them is still unsatisfactory. So, in this dissertation, we will take these parts as our research object.

Generally, a supply chain centered with host enterprise is composed of three parts: supply, production and distribution part. To satisfy the customer demand, maximize its profit and win the competition in the increasingly globalized economy, the host enterprise needs first to analyze the market and understand the customer demand, decide what and how much to produce based on this analysis, and then it may begin to plan its production process and organize its supply chain. Obviously, the distribution part plays crucial role for the success of distribution chain management. Of course, these three parts are related with each other, and we should study them simultaneously. But, for limited energy, source and time, it is difficult to cover all these three parts in one dissertation. If we care about all parts in detail, maybe, the resulted model will be too large to be solved, or the solution will be too general to be applied in practice. To avoid these problems, in this dissertation, we mainly concentrate on the distribution part (which is closely related to market) of a supply chain, and formally define it as distribution chain (detail is shown in chapter 2).

Like any other systems, before implementing a distribution chain, we need to design it. Distribution chain is a large system. Once formed, it is difficult and costly to change it. Obviously, the quality of distribution chain design has long term influence on its management. This property further emphasizes the significance of designing a distribution chain properly. For these reasons, we will take distribution chain design as our main concern in this dissertation.

1.2 Research Objectives

The main objective for this dissertation is to develop a methodology for the design of distribution chain. The general model of the distribution chain to be designed is shown in Figure 1-1. By the methodology developed here,

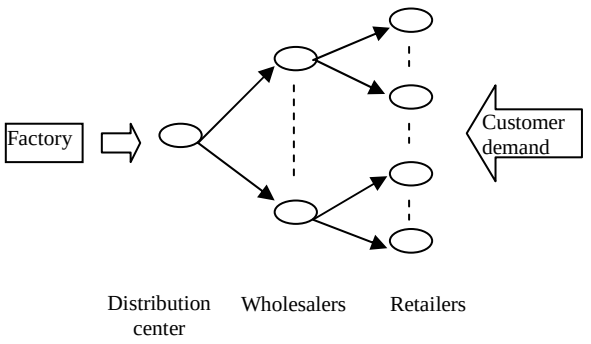


Figure 1-1 General model of the distribution chain to be designed

we can get following design results:

- Configuration of the distribution chain, including number and locations of retailers and wholesalers, and the assignment of retailers to wholesalers.
- Inventory control policy and parameters at each node of the distribution chain.
- Routes for vehicles to deliver product between different nodes (i.e. from distribution center to wholesalers, and from wholesalers to retailers).

To get these results, the methodology for distribution chain design must have following functions:

- *Formulate the problem*. Before designing a distribution chain, the situation for the host enterprise needs to be analyzed. Based on this analysis, constraints and objective for this design can be identified.
- *Collect data from possible distributors*. To some extent, designing a distribution chain can also be viewed as selecting or locating distributors. Before designing, we need to collect data from all possible distributors. Our task is to develop an approach to acquire information from these geographically distributed distributors efficiently and economically. Fortunately, Internet technology can help us to achieve this goal.
- *Evaluate all possible distributors*. We can imagine that, when designing a distribution chain, how large it will be for both the number of possible distributors and the scale of information for one possible distributor. Facing such large scale of information, we need to develop an efficient method to evaluate individual distributor quantitatively, and then select a set of eligible ones to design the distribution chain.
- *Determine configuration of the distribution chain*. As the possible distributors were only evaluated individually above, the ones selected previously can only act as candidates for designing a distribution chain. In this step, the entire structure of distribution chain will be optimized, and final acceptance/rejection of candidates (including wholesalers and retailers) will be decided. Obviously, a mathematical model needs to be founded to accomplish this optimization process.
- *Determine inventory control policy and parameters for each node of the distribution chain*. Inventory control is an inevitable issue in distribution chain design and operation. In this design methodology, we will use a simulation based model to determine when and how much to order at retailers and wholesalers.
- *Optimize routes for vehicles to deliver product between different nodes*. Delivering product between different nodes is another important issue in distribution chain design and operation. When delivering products between different nodes, a lot of routes can be options. Our mission is to develop an algorithm to select the optimal one.
- *Verify the design results*. After determining the configuration of the distribution chain, inventory parameters at each node, and routes for delivering products between different nodes, the design process has been finished. The last step is to verify these design results by evaluating the performance of this designed distribution chain.

URN:NBN:no-3444

Finally, to design a distribution chain efficiently, all models, algorithms, and formulae used in this methodology need to be computerized.

1.3 Thesis Structure

This thesis is composed of 9 chapters, and they are organized as follows.

In chapter 2, after a simple introduction to supply chain, distribution chain is formally defined, and the state of the art for current research on distribution chain is illustrated. Based on the analysis of different research fields in distribution chain management, distribution chain design is identified as our main concern in this dissertation.

In chapter 3, a literature review on distribution chain design is carried out, and most of the existing design methodologies are analyzed. After this analysis, structure of the integrated methodology for distribution chain design is turned out.

In chapter 4, the following questions are answered: what kind of distribution chain can be designed by the methodology developed in this dissertation? And what is the design objective for this methodology?

In chapter 5, a module for evaluating possible distributors is described. This evaluation is realized by three steps: determine factors needed to be considered when evaluating a distributor, acquire information from distributors, and evaluate a distributor by a FL (Fuzzy Logic)-ABL (Array Based Logic) inference engine. After this evaluation, a set of distributors are selected as candidates in designing the distribution chain.

In chapter 6, with these candidates at hand, a set of models, formulae and algorithms are developed to design a distribution chain. First, to determine the exact customer demand at each retailer, an ANN (Artificial Neural Network) based model is developed to estimate the retailer's market share in its customer zone. Then, based on the estimated market share, configuration of the distribution chain is optimized by MIP (Mixed Integer Programming) model, inventory control parameters at each node of the distribution chain are determined by simulation, and product delivering routes between different nodes are identified by genetic algorithm. An iterative design process is used to guide the sequence in applying these models in practice.

In chapter 7, to verify the design result turned out above, a new Petri net form, combinational Petri net, is put forth, and performance of the designed distribution chain is evaluated by this newly developed Petri net form.

In chapter 8, a case study is carried out to illustrate how to apply the methodology developed in this dissertation in designing a distribution chain.

In chapter 9, the methodology developed in this dissertation is summarized, some conclusions are given, and the future research direction in this area is illustrated.

URN:NBN:no-3444

CHAPTER 2 DISTRIBUTION CHAIN

2.1 Introduction

In recent years, supply chain management is becoming more important than the manufacturing process itself [Yam et al., 2000]. By proper supply chain management, the host enterprise can coordinate all activities in the supply chain, and cooperate with other enterprises so as to minimize its cost and maximize its profit. A supply chain centered with host enterprise may be divided into three parts: supply, production and distribution part, as shown in Figure 2-1 [Solvang, 2001]. Supply part mainly deals with the activities for procurement of raw materials or parts that are needed to produce the end products. Production process includes entities and activities for manufacturing parts or semi-finished products and assembling them into end products. Distribution part includes the entities and activities for distributing and delivering end products to consumers. As mentioned previously, the distribution part is our main concern in this dissertation.

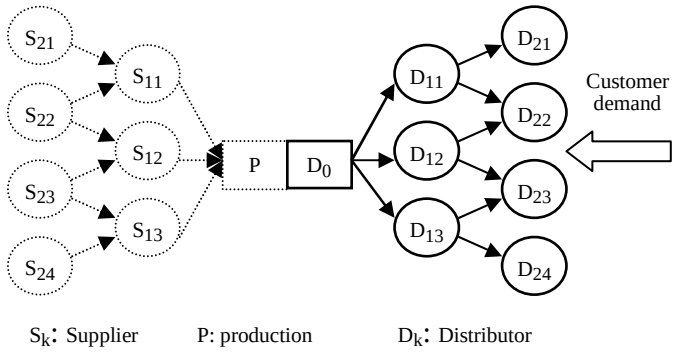


Figure 2-1 Supply chain structure.

2.2 Definition and Life Cycle of Distribution Chain

Definition. In existing literatures, normally the distribution part of a supply chain is referred as distribution network or production-distribution system. To give an explicit concept, we separate the distribution part from supply chain, and formally define it as distribution chain:

A distribution chain is a network of facilities, including distribution center at the host enterprise, and wholesalers and retailers geographically distributed all over the world. These facilities are connected by transportation lines. This network performs store and delivery of end products to fulfill the customer demand efficiently and maximize profit for the host enterprise effectively.

Two issues are addressed in this definition: the scope and mission of a distribution chain. To focus on the study of distribution part, the production process itself is not included in a distribution chain, so the scope of a distribution chain is narrowed and only distribution center, wholesalers and retailers are included in it. The mission of a distribution chain is two-folded: serving the customers and maximizing profit for the enterprise. For an enterprise, maximizing profit is the intuitive aim. At the same time, it also needs to satisfy the consumer's requirements to meet its long term goal. The

decision makers in the enterprise need to achieve nice balance between two aspects. After designing the distribution chain, some valuable information, such as the necessary production rate for the host enterprise, etc., may be gotten to guide the planning of the production process.

Life cycle. Generally, a system life cycle is constructed by three parts [Asbjornsen, 1992]: the first part brings the system into being. Questions, such as how to develop, design and organize the system, are answered in this phase; the second part deals with problems on operation and maintenance of the system. When the environment for the system changes, the system may be ended or reconfigured, that is what to be done during the third part.

Based on such principle, the life cycle of distribution chain is divided into three phases: formation, operation, and reconfiguration or extinction phase. In the formation phase, the host enterprise needs to determine the marketing strategy, select partners, determine the structure of its distribution chain, product delivery mode, and other strategic issues. In the operation phase, all issues related to maintaining inventory, delivering product, and cooperation between different facilities (or even enterprises) are addressed. For a distribution chain, it faces a dynamic environment: market, and it needs to cope with the strong competition with other enterprises. When its environment changes remarkably, the existing distribution chain may be unfit. At this point, the host enterprise may reconfigure the distribution chain to accommodate the new environment, or totally destroy it if it is no longer profitable. The third phase is used to deal with all issues about reconfiguring or destroying a distribution chain.

2.3 Current Research Fields on Distribution Chain

As mentioned above, the study on distribution chain has attracted the attention of researchers and practitioners for several decades, and a lot of papers have been published on the distribution chain management. We have searched related papers in three databases: ISI (Institute for Scientific Information), OCLC (Online Computer Library Center) and BLPC (British Library Public Catalogue), the searching results are shown in Figures 2-2, 2-3, and 2-4.

In these figures, the words in quotation marks are the key words used during the searching process, and the followed value is the number of articles searched by the corresponding key words. For example, in the first tier of Figure 2-2, “supply chain” is the key word, and 1037 is the searching result in database ISI. In each figure, there are three tiers. For blocks in the second and third tier, the key words used in searching process are the key words in themselves plus their super-tier’s keywords. For example, for the block in the second tier of Figure 2-2, the key words actually used in searching process are “supply chain” AND “distribution”, the corresponding searching result is 151. For the right most block in the third tier, the key words are “supply chain” AND “supply chain” AND “distribution” AND “Internet”, and the corresponding searching result is 8. The sum of numbers in the third tier may be slightly greater than, rather than equal to the number in the second tier. The reason is that: some papers may cover

more than one topic (e.g. some papers concern both inventory control and routing algorithm), and such papers may be searched in more than one block.

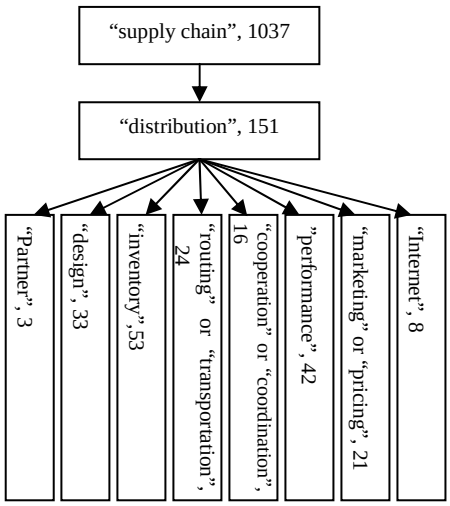


Figure 2-2. Search result in ISI

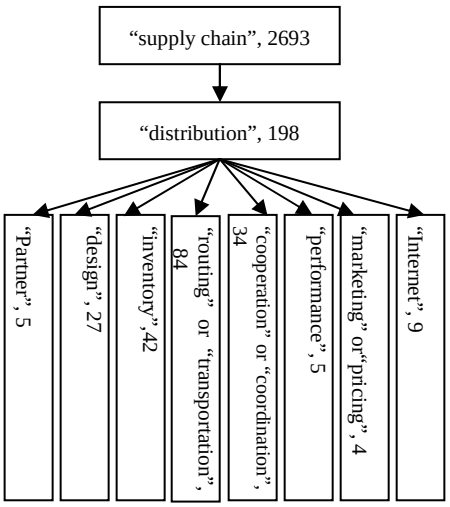


Figure 2-3. Search result in OCLC

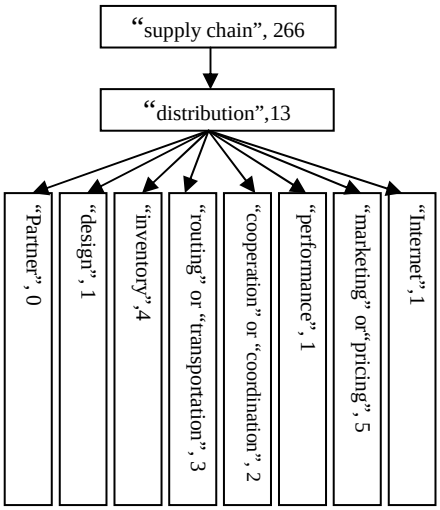


Figure 2-4. Search result in BLPC

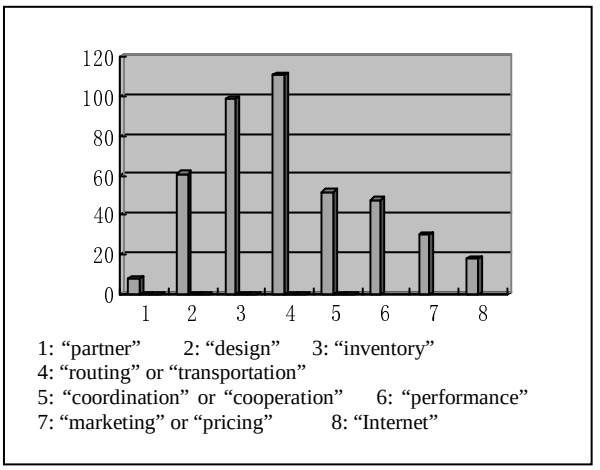


Figure 2-5. Literature distribution for distribution chain

Summing up the articles searched by same key words in three databases, we get Figure 2-5, which can roughly reflect the literature distribution for the study on distribution chain. By this figure, we find that, the most popular issues in distribution chain research are: transportation or routing algorithm for delivering products to customers, inventory control, design of distribution chain, and coordination (or cooperation) in the management of a distribution chain.

As mentioned above, the life cycle of distribution chain is divided into three phases: formation, operation, reconfiguration or extinction phase. Actually, reconfiguring a

distribution chain is similar to forming it, so we categorize all articles searched in three databases into three parts: formation, operation, and extinction. All articles based on the condition that the distribution chain has not been constructed belong to the first category. The articles that are based on a formed distribution chain and mainly concentrate on the operation of a distribution chain belong to the second category. Other articles mainly concentrate on the destruction of a distribution chain belong to the third one. Next, let's take a quick look at the articles searched by different key words, and then identify which category they belong to.

(1) "*partner*". Articles searched by key word "partner" (actually, the key words are "supply chain" AND "distribution chain" AND "partner") mainly illustrate the procedures or methods on how to select a partner. Obviously, such articles belong to the formation category.

(2) "*design*". Such articles mainly address the methods on how to determine the structure of a distribution chain. They belong to the formation category too.

(3) "*inventory*". Such articles illustrate the methods on how to determine the inventory control policy, reorder point and ordering quantity, etc. Obviously, these articles are based on the condition that the distribution chain has been formed, and so they belong to the operation category.

(4) "*transportation*" or "*routing*". Such articles provide methods to solve the routing problems when delivering products to wholesalers, retailers and customers. Same as articles in "inventory", they belong to operation category.

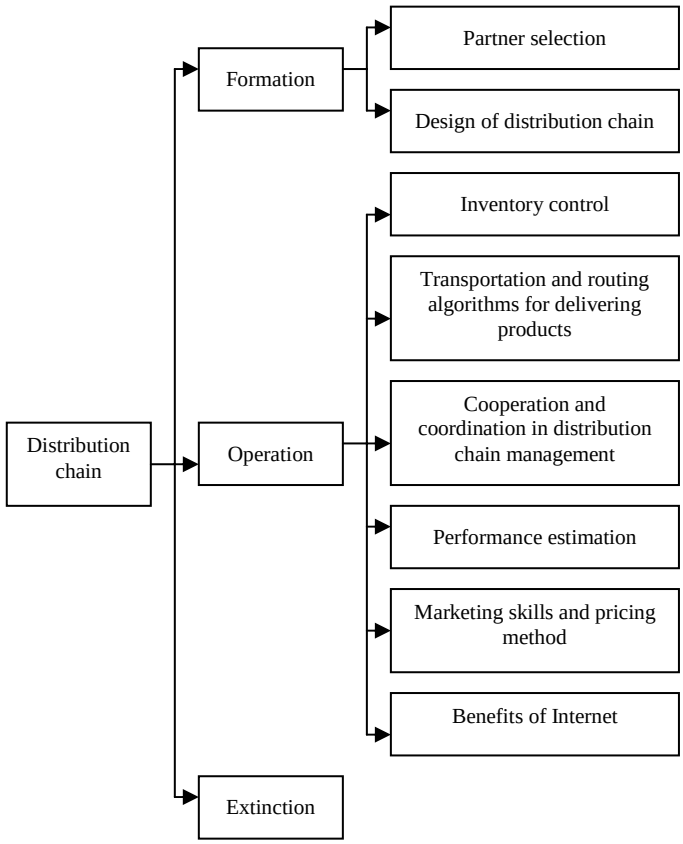


Figure 2-6. Research fields in distribution chain

(5) "*coordination*" or "*cooperation*". Such articles concern the cooperation between different facilities in a distribution chain, and coordination between information flow and material flow, etc. They belong to the operation category.

(6) "*performance*". Such articles mainly address the performance measures for a distribution chain, and methods to estimate these measures. As the performance estimation models mainly reflect the operational aspect of a distribution chain, they are put into the operation category.

(7) "*marketing*" or "*pricing*". Such articles mainly illustrate the marketing skills and pricing methods when operating a distribution chain, they belong to the operation category.

(8) "*Internet*". Such articles mainly illustrate the influence and benefits brought by Internet when operating a distribution chain, so they belong to operation category.

No article is found for the destruction of a distribution chain.

Summarizing the categorizing results mentioned above, we get Figure 2-6, which roughly depicts the current research fields on distribution chain.

2.4 The State of the Art on Distribution Chain Research

In the previous subsection, the main research fields in distribution chain are identified, and numbers of articles on different fields are listed. Now, let's begin to analyze the research state in this area, and illustrate what have been done by these articles. It is impossible to illustrate all papers for all research areas in this literature review. For the limited space, here we only list the typical articles to demonstrate the main achievements in each field.

2.4.1 Formation

(1) Partner selection.

Before selecting partners, we need to evaluate all the possible ones. When evaluating a partner, first, we need to determine the factors needed to be considered when evaluating it. The normally considered factors are: site characteristics, cost, traffic access, market opportunity, and quality of living and local incentives [Min et al., 1999]. Cavusgil et al. [1995] provided criteria especially for evaluating foreign distributors. They categorized the criteria as financial and company strengths, product factors, marketing skills, commitment and facilitating factors. In practice, what kind of factors to be considered heavily depends on the goal for the host enterprise to construct its distribution chain.

Methods on how to evaluate partners have been studied for decades, and different approaches have been developed. Houshyar et al. [1992] gave an 8-step supplier evaluation model. The 8-step procedure is given as follows:

- Define the critical factors, objective factors, and subjective factors.
- Evaluate the critical factor measures.
- Evaluate the objective factor measures.
- Determine the subjective factor weights.
- Determine the supplier weight.
- Evaluate the subjective factor measures.
- Determine objective factor decision weight.
- Calculate supplier performance measure.

Similar descriptive model may also be found in [Motwani et al., 1999, Chick et al., 2000, etc.]

To evaluate a partner quantitatively, some other approaches were developed. Cavusgil et al. [1995] used expert system to evaluate a foreign distributor; Min et al. [1999] applied AHP (Analytical Hierarchy Process) to assess a domestic partner, etc.

For the existing evaluation methods, the final result is given by scoring method, i.e. each category is assigned a weight of importance, and each factor in this category is scored by expertise. Summing up these weighted scores, the final score for this partner is gotten as the indication of its performance. Obviously, such methods are largely affected by subjective judgement, and it is difficult to computerize them.

(2) Design of Distribution Chain

Almost all of the existing design methodologies view the design of a distribution chain as an optimization process, and the most popular optimization method is MIP [Carlos et al., 1997]. Researchers used MIP to minimize cost [Brown et al., 1987, Cohen and Moon, 1991, Cole, 1995, Jayaraman, 1998, etc.] or maximize profit [Cohen and Lee, 1989, Chen et al., 1997, etc.] for multicommodity distribution chain, and then determine its configuration and corresponding parameters.

Besides MIP, other optimization methods are also used for distribution chain design. Berry et al. [1998] used genetic algorithm to optimize the topology of distribution network. Anthony [2000] used simulated annealing method to optimize the structure of a distribution chain, and he claimed that the simulated annealing method behaves better than MIP. About the research on distribution chain design, further literature review will be given in Chapter 3.

2.4.2 Operation

(1) Inventory control

The basic inventory control policies are: (*R, S*) and (*s, S*) policy. For (*R, S*) policy, a replenishment order is placed to raise the inventory position to *S* for every period of *R* unit time. For (*s, S*) policy, the order is placed to raise the inventory position to *S* once

the at hand stock is less than or equal to *s* [Henk, 1994]. A lot of methods have been developed to determine these parameters by minimizing the inventory carrying cost or satisfying the given fill rate. Diks [1998] developed a set of formulae to calculate the parameters for (*R, S*) policy by minimizing inventory carrying cost. Heijden [1997] put forward a new rationing rule (ration limited capacity to a set of warehouses) for the (*R, S*) control policy. And then based on this rationing rule, an optimization model was founded to calculate control parameters by satisfying the fill rate constraint [Heijden, 1999, 2000]. Besides (*R, S*) policy, (*s, S*) was also well studied. Silver [1985] (and Sabri, [2000], etc.) used safety stock to determine the reorder point *s*, and then determined the order up to level *S* by minimizing inventory carrying cost. Other optimization model for (*s, S*) policy can be found in [Henk, 1994] and [Ganeshan, 1999].

(2) Transportation and routing algorithms.

In this field, TSP (Travelling Salesman Problem) plays an important role [Tayur et al., 1999]. The purpose for TSP is to find shortest path given the visiting constraints for the salesman (or the vehicle). Combinatorial optimization algorithm can be used to solve such kind of problems [Kreyszig, 1999]. When the vehicle capacity is considered, the problem becomes CVRP (Capacitated Vehicle Routing Problem). A lot of heuristics has been proposed for the CVRP. These heuristics may be categorized into: Constructive method [Paessens, 1988, etc.], Route First-Cluster Second method [Haimovich et al., 1985, etc.], Cluster First-Route Second method [Noon et al., 1991, etc.], and Incomplete Optimization method [Fisher, 1994].

Genetic algorithm is another effective approach for solving the vehicle routing and scheduling problem [Park, 2001]. For example, Gabbert et al. [1991] presented a genetic algorithm approach to learning low-cost routes and schedules for a large rail freight transportation network; Cheng et al. [1996] proposed a hybrid genetic algorithm to solve the fuzzy vehicle routing and scheduling problem, etc.

Some researchers realized that, the transportation planning and inventory control need to be integrated, rather than operated separately [Chandra, 1993, etc.]. Dempster et al. [2000] (and other researchers, such as Murthy et al. [2001], etc.) used IP (Integer Programming) method to plan the inventory-transportation system. In these articles, following objective functions are normally used: minimizing cost, minimizing total distance covered or maximizing total volume delivered. Obviously, for some situations (such as in distribution chain planning, etc), it is absolutely necessary to consider transportation planning and inventory control simultaneously.

(3) Coordination (or cooperation) in distribution chain management

Weng [1999] (and Gavirneni et al., [1999], etc.) compared the performance of a distribution network in the presence and absence of coordination, and pointed out that, the coordination is important to achieve joint profit, especially when the demand is sensitive to price. There are mainly two types of coordination in managing a

distribution chain: information coordination in managing multi-echelon inventory systems [Gavirneni, 2001], and coordination between transportation and inventory control [Geunes, 2001]. In these articles, models and formulae have been developed to determine the coordination parameters.

(4) Performance estimation.

The following performance measures of a distribution chain are normally defined and estimated in existing articles: profit, cost, customer service, flexibility, quality, asset utilization, fill rate and lead time [Viswanadham, 1997, Jayashankar, 1998, Beamon, 1999, Solvang, 2001]. The main methods to estimate these performance measures are: mathematical method [Beamon, 1999], Fuzzy Logic [Solvang, 2001], simulation based method [Alfieri, 1997, Reis, 2001, Gjerdrum, 2001]. As stated in [Solvang, 2001], performance estimation is becoming one of the major research areas in supply chain (distribution chain) management. For this research field, further literature review will be given in Chapter 7.

(5) Marketing skills and pricing methods

Carter et al. [2002] (and Min, et al. [2000]) studied the marketing skills and purchasing social responsibility (PSR), and concluded that, PSR has direct and positive impact on supplier performance.

Pricing is another important issue in managing a distribution chain. Singh [1997] developed an IT support generic model to help decision makers in determining the price for their product, and claimed that the resulted price enables the firm to meet all its costs and make a profit whilst meeting its longer term strategic goals. Other pricing model may be found in [Nagle, 1987] and [Singh, 1996].

When there are several companies in a distribution chain, these companies negotiate to determine the transfer price, i.e. the price for products shipped between primary and secondary companies, or secondary and tertiary companies. Gjerdrum [2001] developed an IP model to determine the transfer price by maximizing the joint profit. Other transfer pricing model may be found in [Carlos, 2001].

(6) Benefit of Internet

Internet has brought tremendous revolution for our life, including distribution chain management. It may benefit the retailing industry [Rao, 1999], information exchange [Dasgupta et al., 1999], and distribution chain operation [Gavirneni et al., 1999]. In the future, obviously it will play more important role for the integration of different enterprises in a distribution chain.

2.5 Summary

In this chapter, after a simple introduction to supply chain, the formal definition for distribution chain is given, and then the scope, mission and life cycle for a distribution

chain are specified. Then, a literature search is carried out to illustrate the research state on distribution chain management, and main research fields are identified and categorized based on the division of life cycle. The main purpose for this literature review is to identify the main concern for this dissertation.

It is impossible to study all fields in a dissertation. So, we need to select one of them as our main concern. Among these research fields, distribution chain design is mainly used to determine the configuration of a distribution chain, including number and locations of distributors. Obviously, such result may affect the management of a distribution chain for a long term, and once the distribution chain is formed, it is costly, even impossible to change it. Because distribution chain design is so important, a lot of study has been done on it. As indicated in Figure 2-6, it is the third most popular research field in distribution chain management. But, unfortunately, the achievement is not so satisfactory. As mentioned above, almost all of the existing methodologies view the design of distribution chain as an optimization process. Distribution chain is a large system. To design such a large system, a lot of factors, including qualitative and logic factors, must be considered. Obviously, such factors are difficult to be considered in an optimization model. During forming the optimization model, some less important factors may be neglected. So, the design result must be verified by estimating the performance of the designed distribution chain. Unfortunately, no verification is provided in existing design methodologies. Based on this analysis, we will take distribution chain design as our main concern in this dissertation. In the next chapter, existing methodologies for distribution chain design will be further analyzed, and the structure of our integrated design methodology will be illustrated.

Distribution Chain

URN:NBN:no-3444

CHAPTER 3 STRUCTURE OF THE INTEGRATED METHODOLOGY FOR DISTRIBUTION CHAIN DESIGN

3.1 Introduction

Design exists almost every where in our life, especially in industry area. For example, before developing a new product, it needs to be designed; before implementing a manufacturing system, it also needs to be designed. The quality of design is a vital factor for the success of developing this product, or implementing this manufacturing system.

Same as any other systems, a distribution chain needs to be designed before implemented. Korpela et al. [1999] described distribution chain design as "a strategic level network design problem" (i.e. determining the number and location of wholesalers and retailers strategically), and stated that "the nature of the decision is long-term and the influence of the warehouse location decision on the profitability of the company will last for years". Baunach et al. [1995] analyzed Germany construction industry, and gave the benefit of distribution chain design in more detail: by opening two main depots and closure of five sub-depots (according to the analysis and design result), their case company can increase turnover by 5%, and the profit by 20%. Other literatures, such as [Carlos et al., 1997], [Escudero et al., 1999], [Lakhal et al., 2001] etc., also stated that the design of distribution chain is a vital step to achieve the success of distribution chain management.

Facing with volatile market and intensive competition, an enterprise always needs to reconsider its distribution strategy. Aronsson [2000] examined the structural changes in distribution, and concluded that best practice enterprises use a time based distribution strategy. This means that the enterprise needs to design and re-design its distribution chain from time to time. This statement illustrates the significance of distribution chain design again.

Compared with its importance, research on distribution chain design is not so satisfactory. In what follows, we will give a literature review on the main existing methodologies for distribution chain design, then illustrate their shortcomings. Based on this analysis, the structure of the integrated design methodology will be put forward.

3.2 Analysis of Existing Methodologies for Distribution Chain Design

3.2.1 Review of existing design methodologies

Probably, [Geoffrion et al., 1974] is the first paper to use MIP (Mixed Integer Programming) in designing a production-distribution system. Since then, mathematical programming, heuristics, simulation annealing, etc. have largely been used to design a

distribution chain (or production-distribution network as called in these papers). In existing methodologies, optimization is the main design approach [Vidal et al., 1997, Sabri et al., 2000]. For an optimization model, following general form is explicitly or implicitly applied:

***Objective function**: Minimize cost = production cost + inventory maintaining cost* (3-1)
+ transportation cost +…… ,
Or: *Maximize profit = all revenues – total cost*
……

Subject to: *Production capacity constraints* (3-2)
Inventory capacity constraints
Transportation capacity constraints
……

In existing design methodologies, following three steps are normally used to realize this general model:

- Setting the design objective(s).
- Building the model.
- And solving the model.

In what follows, we will review existing design methodologies from these three aspects.

(1) Setting design objective(s)

In existing design methodologies, minimizing cost and maximizing profit were popularly set as design objectives [Brown et al., 1987 (minimizing cost), Cohen and Lee, 1989 (maximizing profit), Cohen and Moon, 1991 (minimizing cost), Cole, 1995 (minimizing cost), Chen et al., 1995 (maximizing profit), etc.]. In these design methodologies, the total cost includes production cost, facilities opening cost, inventory maintaining cost, transportation cost, etc., and profit is expressed as total revenue minus total cost.

Korpela et al. [1999] put forward a customer oriented approach to design a warehouse network. They stated that “costs are often used as the major factor…, whereas enough attention is not paid to the various quantitative and qualitative customer service elements”. Based on this analysis, they took maximizing customer satisfaction as their design objective.

Some researchers realized that, solely optimizing a single performance can not fulfil the requirement of production-distribution management, so they began to study multi-objective optimization model. Sabri et al. [2000] provided a multi-objective MIP approach in supply chain design. In this optimization model, cost, customer service level (fill rate) and system flexibility were considered simultaneously in their objective function.

(2) Building the model

As shown in the general form, there are two parts in an optimization model: objective function and constraints. To simplify the design process, some researchers built models for designing multi-commodity, but single-product production-distribution system [Cohen and Lee, 1985, Geotschalckx et al., 1995, Dogan et al., 1999, etc.]. In the objective functions of such models, following basic items were normally considered:

- Revenues at retailers.
- Production cost.
- Inventory opening and maintaining cost.
- Product delivery cost.
- Etc.

For constraints, following types of constraints were normally considered in these models:

- Constraints for customer demand satisfaction.
- Capacity constraints for each facility
- Material flow balance.
- Etc.

For the decision variables, binary variables were used to indicate the selection/rejection of a facility, and continuous variables were used to model the volume of products to be produced at plant, kept at different warehouses, etc.

For some production-distribution systems, single-product is not the case, so Jayaraman [1998] (and Sabri et al. [2000], etc.) extended the model into a multi-product situation. For such kind of models, besides the items considered above, costs for different production methods were also considered. Obviously, this makes the model more complex, but more realistic and useful.

Because of the economic globalization, Hoder et al. [1986] described an international plant location model. Besides the common items mentioned above, exchange rate fluctuation, international interest rates, and other related factors were considered in their model.

In practice, when designing a distribution chain, some factors (such as customer demand, transportation time, etc.) are uncertain. The design methodologies mentioned above neglected the uncertainties of these factors, and took them as constant. Such kind of treatment can simplify the design model, but bring considerable error. To improve the design, some existing design methodologies took demand as stochastic [Escudero, 1999, MirHassani et al., 2000], and ISP (Integer Stochastic Programming) or other stochastic mathematical methods were used to design the distribution chain.

(3) Solving the model

Vidal et al. [1997] claimed that, MIP is the main method in solving the models built above. My search in database ISI releases the same result: about 85% of the existing design methodologies used MIP to solve their models. Some of them also integrated MIP with other techniques. For example, Korpela et al. [1999] used AHP (Analytical Hierarchical Process) to analyze alternative warehouse operators, and then used MIP to maximize the customer satisfaction; Jayaraman [1998] developed an efficient procedure for warehouse network design: a MIP model was used to minimize the cost, and a procedure, called as WARELOG, was used to solve this large scale model. As uncertain demand was considered in [MirHassani et al., 2000], the stochastic form of IP (Integer Programming), ISP was used to solve their model.

Besides MIP, other methods were also used to design a production-distribution system. Berry et al. [1998] used genetic algorithm to optimize the topology of distribution network. In this method, a chromosome represents one topology of the network, and the optimized topology of distribution network is determined by minimizing cost. Anthony [2000] used simulated annealing method to optimize the production-distribution system. The simulated annealing process alters potential configurations to arrive at the final configuration with lowest cost.

After solving the model, values for decision variables are acquired. Then the configuration of the distribution chain (i.e. the numbers and locations of wholesalers and retailers, and assignments of retailers to wholesalers) is determined. That is the design result.

3.2.2 Shortcomings of existing design methodologies

In previous subsection, existing methodologies for distribution chain design are reviewed. The common feature for existing methodologies is that, most of them used optimization approach to design their distribution chains. When formulating their objective functions, revenues and costs (including production cost, inventory maintenance cost, product delivery cost, etc.) were considered. Of course, these items are basic factors that need to be considered when designing a distribution chain. But, distribution chain is a large and complicated system. Just considering these basic factors is far from enough. For example, when selecting a retailer, its marketing environment must be considered. It is unimaginable to select a retailer with bad marketing environment, even if its product delivery cost (or other costs) is low. Unfortunately, such qualitative factors can not be reflected in existing design methodologies. As mentioned previously, formulae (3-1) and (3-2) are used as general form in existing design methodologies. It is difficult to express qualitative or logic factors in such mathematical models. But, without considering these factors, the resulted configuration of the distribution chain can not be ideal, even if the model is claimed to be optimized.

Decision maker's preference is another important issue to be considered when locating retailers, and this preference may be different for different products. For example, for

some products, technical service is important, so the decision makers prefer to select those retailers that can provide such technical service. But, such preference can not be reflected in the existing methodologies too.

Existing design methodologies may encounter difficulty when designing large scale distribution chains. In practice, when an enterprise wants to design or redesign its distribution chain, it faces hundreds, even thousands of possible distributors. If inputting these candidates directly into the existing design methodologies, the resulted optimization model may be very large, even intractable to be solved.

If we set a distributor evaluation and selection module before designing the distribution chain, the problems mentioned above may be solved. By this module, a distributor can be evaluated comprehensively. All factors related to the profit (including those qualitative and logic factors) can be considered, and the decision maker's preference may be reflected in the evaluating process. After evaluating all possible distributors, a set of eligible ones are selected to design a distribution chain. This filter process can reduce the scale of the design model, and make it tractable.

There is another type of shortcoming in existing design methodologies: no verification is set for the design result. As mentioned in previous subsection, after solving the model, design result is obtained. This result is taken as the final design without verification. Distribution chain is a large system. Without verification, it is dangerous to implement the design directly. For example, in existing design methodologies, the static feature of a distribution chain was considered, but the dynamic feature and interaction between different processes were neglected. This raises a question: whether there is conflict when operating the designed distribution chain? The existing design methodologies can not answer this question.

At the same time, existing design methodologies can not guarantee that the performance of the designed distribution chain is satisfactory. The design model is an abstract of reality. During the abstracting process, some less important properties of the system are ignored. It is not guaranteed that the ignored properties are trivial for the performance of resulted distribution chain. This raises a question: although optimization method has been used in the designing process, is the performance satisfactory for the decision makers of the host enterprise? The existing design methodologies can not answer this question too.

All these shortcomings are vital, not trivial for the success of distribution chain management. Based on this analysis, it is necessary to develop a new methodology for distribution chain design.

3.3 Structure of the Integrated Methodology for Distribution Chain Design

Distribution chain is a large system. Without guidance of systematic approach, the result in designing such a large system may be incomplete, or even useless. The general problem-solving framework (as shown in Figure 3-1) has long been recognized as a useful model for structured decision making [Wu, 1994]. In this subsection, first, this general problem-solving framework is simply introduced. Then, with the guidance

of this systematic approach, the structure of the integrated methodology for distribution chain design is illustrated.

3.3.1 General problem solving framework

Wu [1994] illustrated a general problem-solving framework, shown as Figure 3-1. This framework is composed of following blocks:

(1) Analysis of situation. This is similar to answer "where we are now?" in a journey planning process. Analysis of the problematic situation is the first stage for this systematic approach. The aim of this stage is to analyze the existing system and identify the problem to be solved.

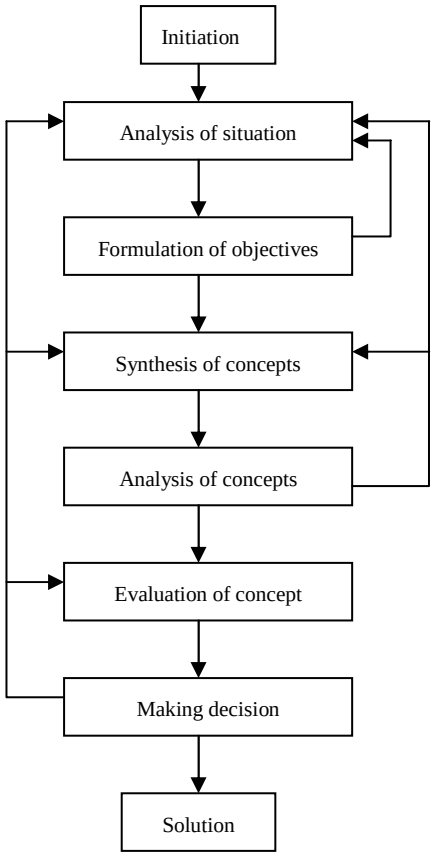


Figure 3-1 General problem solving cycle [Wu, 1994]

(2) Formulation of objectives. Having answered the question "where we are now?" in planning a journey, it is time to answer "where to go?" The goal for this stage is to determine the required performance of the system to be designed, identify objectives and constraints. The combination of first two stages in this general problem-solving model, i.e. analysis of situation and formulation of objectives, has been referred as problem formulation, because these two stages together identify the gap between the present system state and future one.

(3) Synthesis of concepts. After completion of problem formulation, we come to the next phase: problem solving phase, synthesis and analysis of concepts in Figure 3-1 belong to this phase. The first task involved in problem solving is to generate a set of possible alternative routes to the objectives. That is finished by "synthesis of concepts". Synthesis of concepts requires creation of a comprehensive set of alternative solutions. The number of ideas generated should be as great as possible given the time and resource constraints of project.

(4) Analysis of concepts. After generating all possible routes, we come to the second stage of problem solving phase: analysis of concepts. Analysis of concepts can be viewed as a continuous refining process. With this process, the original set of

possibilities is gradually narrowed down, or converged, until one or more solutions are left which satisfy the project objectives.

(5) Evaluation of concepts and making decision. After problem solving process, one or more solutions are determined. The next task is to evaluate the refined solution(s) and then make the final decision, that is the decision making process. During decision making process, the first task is to verify whether the refined solution(s) can fulfill the requirement of system objectives. If it is true, the final decision is made, and the solution is output; otherwise, the problem formulating and solving processes need to be re-considered.

3.3.2 Structure of the integrated methodology for distribution chain design

With the guidance of this general problem solving cycle, the structure of integrated methodology for distribution chain design is developed, as shown in Figure 3-2. This structure is mainly composed of three blocks: problem formulation, system design and making decision. In the first block, the problem is analyzed. A clearly understanding of the problem is the foundation to solve it successfully. In the second block, steps and methods for solving the problem are presented. By applying these procedures and methods, the distribution chain is designed. In the third block, this design result is evaluated, and the final solution is given out. Next, we will explain these three blocks in detail.

(1) Problem formulation

In this phase, the first task is to analyze the present situation of the system. Based on this analysis, the designers need to determine what kind of distribution chain will be designed. To answer this big question, following two sub-questions need to be answered:

- What is the object to be distributed? This is a simple but crucial question, because different object may result in different types of distribution chain. For example, distribution chains used to distribute service or product are different.
- What type of distribution chain will be designed? For the host enterprise, to sell its product, it may directly face the customers, or employ retailers (even wholesalers) to distribute its product. At this point, the designers need to decide which type of distribution chain is appropriate. This may be decided according to the product's characteristics.

After determining the type of a distribution chain, designers need to specify the design scope. Distribution chain is a large system. When designing it, a lot of parameters need to be determined. It is impossible to address all issues in a design methodology. So, before designing it, designers need to narrow the problem, and specify their concerns.

Determining constraints is another issue that must be addressed before beginning the design process. Some basic constraints such as production capacity of the host

enterprise, transportation capacities of warehouses, etc. must be specified before designing a distribution chain.

After analyzing the design object, designers can begin to set the designing objectives. Normally used objectives include minimizing cost, maximizing profit, etc. What kind of objective will be used in designing a distribution chain depends on the state of the enterprise, its present problems, and its goals.

The problem formulation phase is an iterative process: after analyzing the design object, the objectives are set. After setting objectives, the design object may be re-analyzed, and the possibility to realize these objectives is checked. Such iterative process will continue until the design object is clearly understood and the objectives are properly set.

(2) System design

Different from existing design methodologies, a pre-design process (i.e. evaluation of possible distributors) is added into this system design phase. In this pre-design process, all possible distributors (including wholesalers and retailers) are evaluated by an evaluation module. This module is partly similar to "synthesis of concepts" block in Figure 3-1. After this evaluation, a set of eligible distributors are selected as candidates according to pre-defined criteria, and only these candidates are allowed to go into the following design process. Main functions of this evaluation module are depicted as follows (the detail of this module is referred to chapter 5).

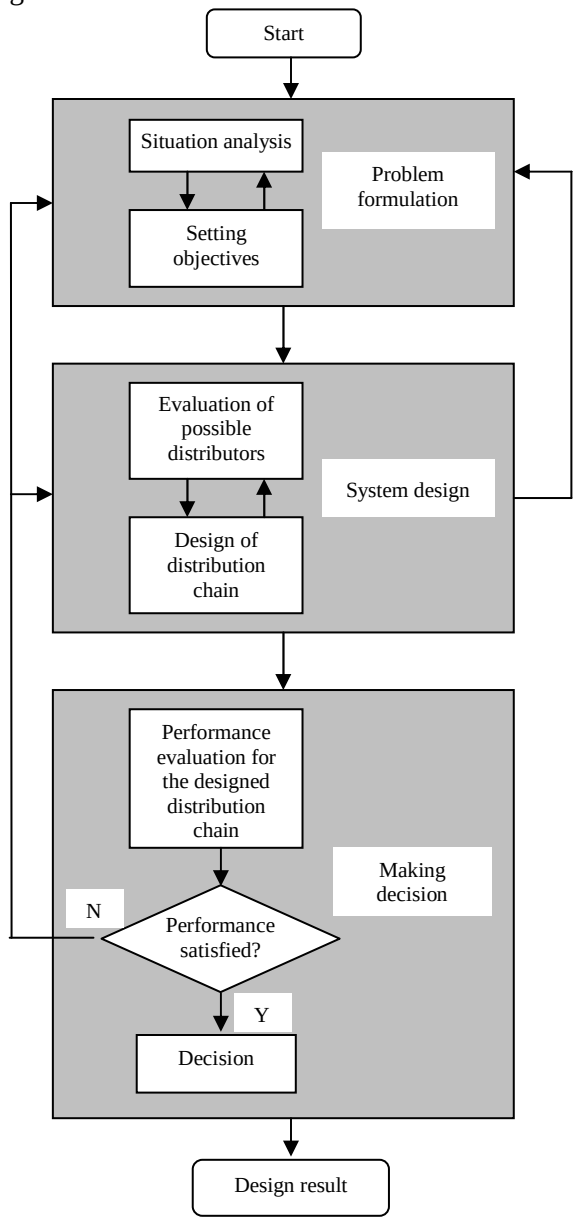


Figure 3-2 Architecture of the new distribution chain design methodology

- It can evaluate a distributor comprehensively. After a systematic analysis, almost all of the factors related to the design objectives, including qualitative even logic factors, are listed hierarchically by AHP (Analytical Hierarchical Process). These factors will be considered in the evaluating process.
- It is possible to integrate decision maker's idea into the evaluating process. The evaluation of a distributor is finished by an integrated FL-ABL approach. Both FL and ABL have capability to integrate semantic expertise and complicated knowledge into the inference process. This makes it possible to consider decision maker's preference in the evaluation module.
- It can reduce the scale of design model remarkably. After evaluating all possible distributors, only a set of them are selected to enter the next process: design of distribution chain. This can reduce scale of the design model, and make it possible to use our design methodology in practice.

After evaluating all possible distributors, a set of selected distributors are input into the next module: design of distribution chain. Detail for the module of distribution chain design is referred to Chapter 6. By this design module, the configuration of a distribution chain is determined, inventory parameters at each node of the distribution chain are optimized, and routes for delivering product between different nodes are identified. All models in this design module are formulated based on the objectives and constraints set in problem formulation phase.

(3) Making decision

After system design phase, the design process for a distribution chain is finished. To verify the design result, a simulation based module is developed to evaluate the performance of this designed distribution chain. The detail of this module is referred to Chapter 7. By this performance evaluation module, following questions can be answered:

- Whether there are conflicts when running this designed distribution chain? As mentioned above, this performance evaluation module is simulation based, so it can reflect the dynamic properties of activities and interaction between them. This makes it possible to find conflicts when implementing the designed distribution chain.
- Whether the performance of this "to be" distribution chain is satisfactory? By running the simulation model, performance measures such as cost, profit, etc. can be estimated. This gives us opportunity to check whether the designed distribution chain is satisfactory.

After performance evaluation, the decision on whether this designed distribution chain can be implemented may be made. If the designed distribution chain can satisfy the objectives set in problem formulation phase, then the design methodology outputs the design result as final solution. If one or more of the performance measures is not satisfactory, we may find the cause by analyzing this simulation model, and then re-consider the corresponding factors in the former pre-design and design modules.

Three modules, namely, distributor evaluation module, design module and performance evaluation module, form the core of this integrated design methodology. The role of each module and the design process of a distribution chain can roughly be shown as Figure 3-3. Distributor evaluation can be viewed as pre-design of a distribution chain. To some extent, design of a distribution chain can be viewed as selection of possible distributors, and this pre-design module provides candidates for the selection process. Performance evaluation can be viewed as post-design of a distribution chain. It provides opportunity for us to verify the design result.

As mentioned in previous subsection, in existing methodologies, a distribution chain is designed by three steps: setting design objective(s), building the model and solving the model. Comparing this design procedure with Figure 3-2, we find that, two modules are added into the integrated methodology, i.e. distributor evaluation module and performance evaluation module. In our integrated design methodology, these two modules are indispensable.

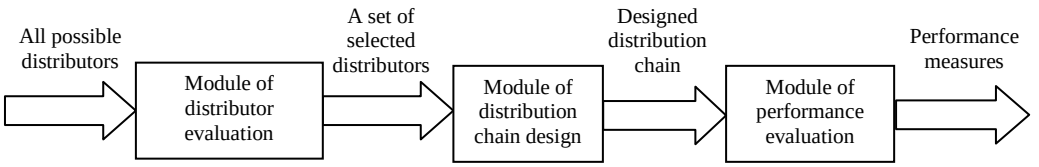


Figure 3-3 The design process for a distribution chain

3.4 Summary

In this chapter, first, a literature review is carried out on the existing methodologies for distribution chain design. Based on the analysis of these existing methodologies, a new methodology, integrated methodology, is put forth for the design of distribution chain, and then its structure is illustrated, as shown in Figure 3-2. This is a key figure in this dissertation. With the guidance of this structure, this integrated methodology is developed, and each module will be illustrated in the later chapters of this dissertation. Based on this structure, the framework of this dissertation is shown as Figure 3-4.

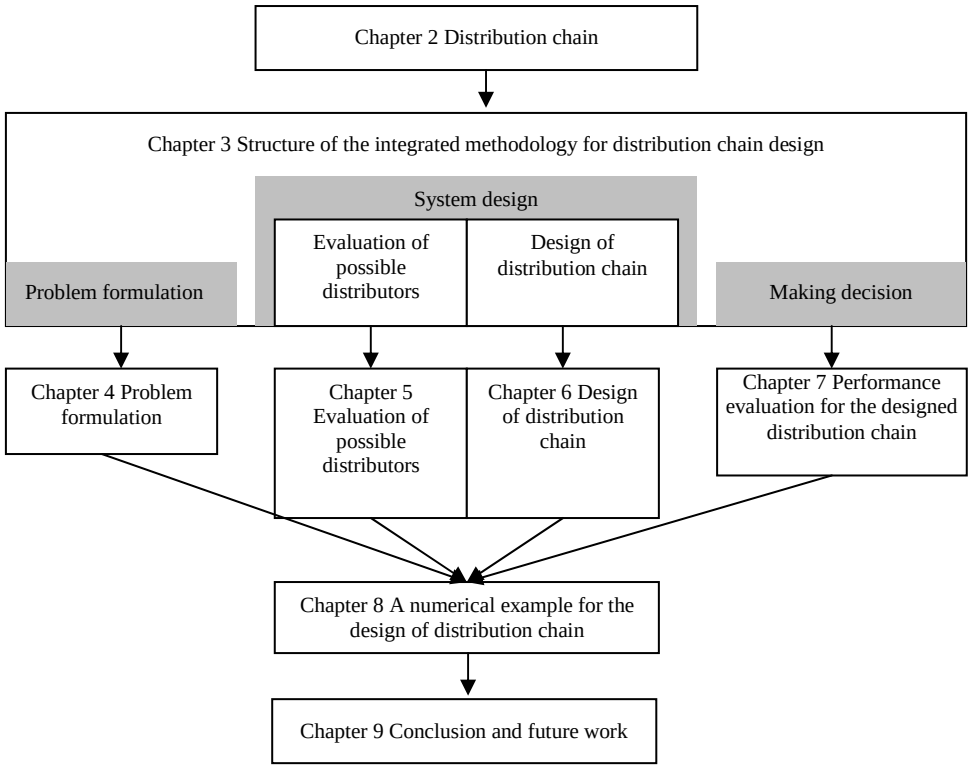


Figure 3-4 The framework of this dissertation

CHAPTER 4 PROBLEM FORMULATION

As shown in Figure 3-2, the first stage in designing a distribution chain is problem formulation. A good understanding of the facing problem is the prerequisite to solve it successfully. As mentioned in the previous chapter, there are two processes in this phase: situation analysis and setting objectives. Next, we will illustrate these two processes in detail.

4.1 Design Object and Scope

These two issues are determined by situation analysis. In this process, based on the analysis of host enterprise's present situation, following questions are answered: what type of distribution chain will be designed (design object), and what to be determined by this design (design scope).

(1) What to be distributed

Before determining the type of distribution chain, we need to answer: what is the object to be distributed? Normally, there are two kinds of objects: physically visible object: products and physically invisible object: service. As manufacture is the main part in industry area, the former one is selected, i.e. we will design a distribution chain which is used to distribute products. In modern industry, mass production is still one of the main manners in producing products, so we assume that the production mode for the host enterprise is mass production.

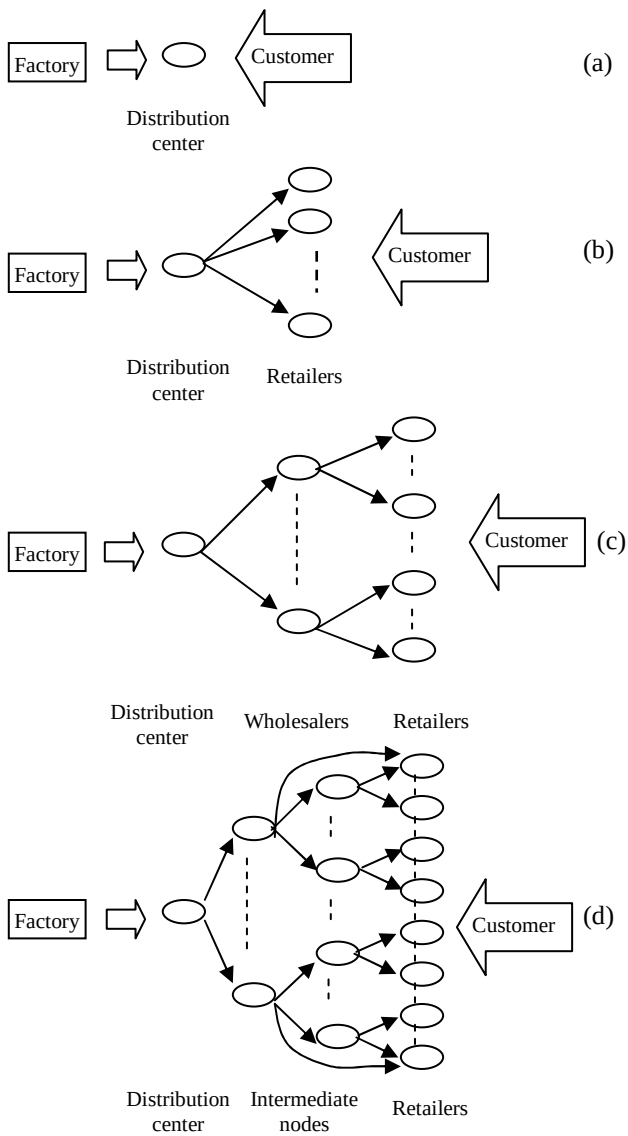


Figure 4-1 General distribution chain types

(2) Type of the distribution chain

For a distribution chain used to distribute products, generally, there are following four types (as shown in Figure 4-1):

- The enterprise directly distributes its products, as shown in Figure 4-1(a).
- The enterprise employs retailers to distribute its products, as shown in Figure 4-1(b).
- The enterprise adds a tier of wholesalers between its distribution center and retailers, as shown in Figure 4-1(c).
- The enterprise adds two or more tiers of wholesalers between its distribution center and retailers. Some times, a wholesaler may go around the descendent wholesaler, and directly reaches the retailer(s), as shown in Figure 4-1(d).

For an enterprise with mass production, normally, it does not distribute its products by itself, so the first type is out of our consideration. To achieve scale effects for small customers, normally, host enterprises set regional distribution centres at different customer zones, and these centres act as wholesalers for those zones, so the second one (direct distribution) is not so popular, and it is out of our consideration too. The third one is a general model of distribution chain [Heijden, 2000]. Moreover, by studying this common and basic form, the general principle for designing a distribution chain can be turned out, and this principle may be extended to design those complicated distribution chains (such as the forth type as shown in Figure 4-1(d)). The general design principle turned out here may also be applied for direct distribution. For example, we can view a wholesaler in Figure 4-1(c) as a distribution centre, then the relationship between this wholesaler and retailers connected to it can be an example of direct distribution.

To simplify the design object and focus on the design principle itself, in this dissertation, we will take the third type as design object, i.e. we will design a distribution chain with one distribution center, one tier of wholesalers and a set of retailers. For the designs of more complex distribution chains (e.g. the distribution chains with more tiers of wholesalers, or distribution chains implementing several distribution strategies), we leave them as future work. In this basic form, we assume that the host enterprise will build distribution center by itself. Here, wholesalers are referred to those regional distribution centres which belong to the host enterprise, so they will also be built by the host enterprise itself. Retailers are selected from existing ones. These assumptions accord with reality.

(3) Design perspective: process design

Before specifying design scope, we need to determine the design perspective. According to [Aronsson 2000], there are three perspectives in designing a supply chain (or distribution chain): process design, function design and organization design. Process design is mainly concerned with the consecutive order and delivery cycles,

including their lead times, storage point locations, activities inside each cycle and their integration. Function design determines functions for each facility, resources needed for each function, and relationship between different functions. Organization design is concerned with how the logistics is organized within and between companies, including the description of how the responsibility for logistics activities is organized in the company, which logistics activities are performed within the company, and which are outsourced, etc. Process design is the basic component for the entire design of a distribution chain. Only when the process design is determined, the functions for each facility can be specified, and the organization can be designed. For its significance, in this dissertation, we will concentrate on process design of a distribution chain.

(4) Design scope

In this dissertation, we will concentrate on following issues in designing a distribution chain:

- The configuration of the distribution chain, including number and locations of retailers and wholesalers, and the assignment of retailers to wholesalers.
- Inventory control policy and parameters at each node of the distribution chain.
- Routes for vehicles to deliver products between different nodes (i.e. from distribution center to wholesalers, and from wholesalers to retailers).

(5) Other issues

Constraints such as production capacity, transportation capacity, etc. are specified according to the present situation of the host enterprise. To concentrate on the basic issues in designing a distribution chain, we do not consider exchange rate fluctuation, international interest rate, etc. in this dissertation.

According to above analysis, the delimitation of our integrated methodology for distribution chain design is formed and shown in Figure 4-2. The shaded part is our concern.

Object to be distributed	Scale of enterprise	Production mode	Design perspective	Customers
	Small	Per unit production	Function design	Few customers
Products	Medium to large	Mass production	Process design	Many customers
Service			Organization design	

Figure 4-2 Delimitation of the integrated methodology for distribution chain design

4.2 Design Objective

Following three objectives are widely applied in designing a distribution chain (or supply chain):

(1) Minimizing cost. Each activity inside a distribution chain causes cost. By minimizing the sum of these costs, activities may be organized properly.
(2) Maximizing profit. There is no doubt that the objective for an enterprise is to make profit. Without being profitable, there is no meaning for an enterprise to survive. To some extent, maximizing profit also means minimizing cost, so this objective can partly cover the above one.
(3) Maximizing customer satisfaction. In recent years, this objective has been attracting attention of researchers. Customer satisfaction is a complex issue, and it is not easy to be described as profit or cost. Korpela et al. [1999] deducted customer satisfaction from three aspects: reliability, flexibility and customer's logistics costs.

For an enterprise, maximizing profit and satisfying customer requirements are basic objectives, and they are mutually affected. Making money is the ultimate goal for an enterprise, but if it only considers its profit, and pays no attention on customer requirements, it will lose the market share and then profit eventually. Based on this point, in this dissertation, we set the objective for designing a distribution chain as: maximizing profit subject to satisfying customer requirements. Generally, maximizing profit is the ultimate objective of an enterprise, and satisfying customer requirements is for its long term goal to win the competition. Next, we will discuss these two issues in detail.

(1) Maximizing profit

According to the formula:

$$Profit = price \times sale - cost$$

Maximizing profit means maximizing price, volume of sale and minimizing cost. None of the three factors is isolated, and the relationship between them is complicated. Next we will discuss a little about these three factors.

Price is a crucial parameter in managing a distribution chain. Higher price means more revenue, but unreasonably high price may cause the loss of market share, then lower revenue. Singh et al. [1997] stated the pricing principle as: "the prices and resulted sale volumes should enable the firm to meet all its costs and make a profit whilst meeting its longer term strategic goals of capturing or retaining appropriate levels of market share".

Volume of sale is related to many factors such as market environment, price, product quality, customer service, etc. To have better market environment is one of sub-

objectives to manage a distribution chain. It is unimaginable for an enterprise to locate its retailers where only a few of its products are needed. So, when the enterprise selects its retailers, it needs carefully to consider the present situation and market potential for retailers. The detail for retailer selection is referred to Chapter 5. As mentioned above, high price is normally a negative factor for volume of sale. Good product quality and customer service are positive factors to volume of sale, but both of them mean more cost. The enterprise needs to make a compromise and select an appropriate standard of quality and level of customer service.

Cost is mostly related to the inside activities such as purchasing raw material, production process, maintaining inventory, delivering products, etc. According to [Themido et al., 2000], every activity costs a set of resources, including labour, equipment, materials, etc. Given standard of quality and level of customer service, an enterprise needs to optimize its production and distribution planning, and improve its management to reduce cost.

(2) Satisfying customer requirements

Satisfying customer requirements is closely related to some performance measures such as fill rate, flexibility, etc. Next, the relationship between customer satisfaction and these performance measures will be discussed.

Mobråtan [1996] (reference from [Solvang, 2000]) stated that, the customer will be satisfied when following six Rs are present:

- the Right volume of the Right bundle of products and services
- to the Right place
- at Right time
- in Right quality and
- at Right price.

The six Rs can partly be indicated by fill rate (which is defined as the fraction of demand satisfied from stock on hand [Heijden, 1999]). Obviously, raising inventory level can increase fill rate, but this will cause the increase of cost for maintaining inventory. The challenge for decision makers is to find the balance point between satisfying customer requirements and reducing cost.

Solvang [2000] defined the flexibility of supply chain as: “the ability of a supply chain to satisfy dynamic customer requirements by handling environmental uncertainties with profitability”. In the current buyer-market, to win the drastic competition, enterprise needs to improve its flexibility to properly handle the urgent deliveries, special requests, etc. Of course, this will also cause the increase of cost, so the decision maker is facing same problem: balance between improving flexibility and reducing cost.

4.3 Summary

In this chapter, following two questions are answered:

(1) What kind of distribution chain will be designed by the methodology developed in this dissertation? Figure 4-2 gives the answer, and describes different facets of such a distribution chain.
(2) What is the objective to design a distribution chain? In this dissertation, we specify the objective as: maximizing profit subject to satisfying customer requirements. Here, customer satisfaction may be indicated by some performance measures such as fill rate, flexibility, etc.

After answering these two questions, the exact designing process for a distribution chain can begin.

CHAPTER 5 EVALUATION OF POSSIBLE DISTRIBUTORS

According to Figure 3-2, after formulating the problem, we come to the system design phase. Two modules are involved in this phase: evaluation of possible distributors and design of distribution chain. The first module will be finished here, i.e., in this chapter we will develop a module to evaluate all possible distributors, and then select a set of eligible ones to design the distribution chain.

5.1 Introduction

When a host enterprise wants to design its distribution chain, it faces a lot of possible distributors. The number of possible distributors is so large, that it is difficult to begin the designing process at once. So, the host enterprise needs a distributor evaluation and selection module to filter all the possible distributors, and select a set of eligible ones to design its distribution chain. Of course, the method used here may also be applied in partner selection for an enterprise.

The research on partner evaluation and selection has received considerable attention of academicians and practitioners over the last several decades. Next, we will introduce two typical distributor selection modules to illustrate the main achievement in this area.

Cavusgil et al. [1995] developed an expert system for the selection of a foreign distributor. In this paper, a distributor is evaluated from five aspects, namely, financial & company strength, product factors, marketing skills, commitment and facilitating factors, then, an expert system is used to evaluate a distributor. Actually, the approach is a scoring method: every aspect is given a weight to express its importance and a score to express its priority. The final evaluation of a distributor is achieved by summing up the weighted scores.

Min et al. [1999] developed a module for the relocation of a hybrid manufacturing/distribution facility. In this module, six location categories are considered when evaluating a hybrid manufacturing/distribution facility, i.e. site characteristics, cost, traffic access, market opportunity, quality of living and local incentives. Then, AHP (Analytic Hierarchical Process) is used to list criteria for all location categories. The final evaluation of a distributor is also given out by summing up the weighted scores for all criteria.

For existing modules, a distributor is normally evaluated by following process: identify a set of factors, and evaluate it by scoring method. By analyzing these existing modules, following three questions are raised:

- Are these factors complete when evaluating a distributor?
- How to acquire information from possible distributors? The possible distributors are geographically distributed. This makes the information acquisition difficult. But the method to acquire information from distributors is not touched in the existing modules.

- Whether the scoring method, which largely depends on the subjective judgment (expertise), and can not be realized by software, is appropriate when evaluating a distributor?

To answer these questions, in this chapter, a new evaluation module is developed, by which a distributor is evaluated according to following steps:

- Identify the factor set which needs to be considered when evaluating a distributor.
- Collect data from possible distributors by a mobile agent based information acquisition system
- Evaluate a distributor quantitatively by an integrated FL-ABL approach.

These three steps will be illustrated in the following sections.

5.2 Factor Set

5.2.1 Literature review

As shown in Figure 5-1, Min et al. [1999] considered six categories in their evaluation module. This module mainly considered the external condition (environment) of a distributor, but ignored the internal factors, such as the facility for maintaining inventory, communication system, financial state of the company, etc. Obviously, this factor set is not complete.

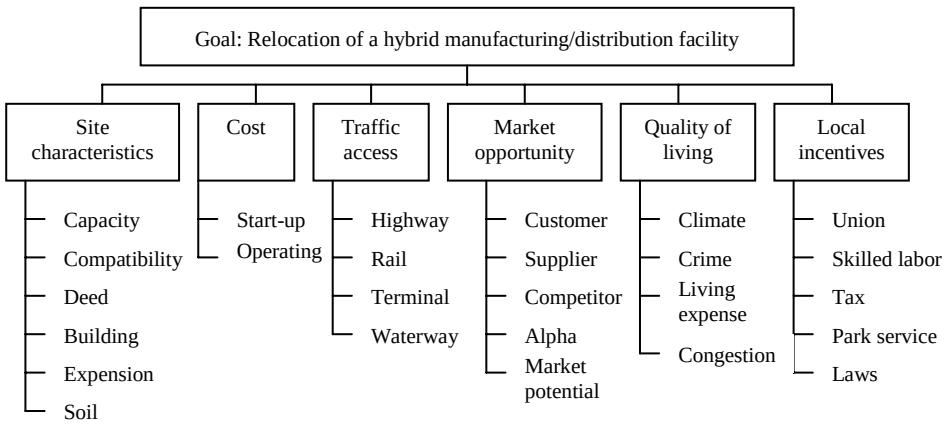


Figure 5-1 Hierarchical representation of the relocation of a hybrid manufacturing distribution facility

Cavusgil et al. [1995] evaluated a foreign distributor from five aspects: financial & company strength, product factors, marketing skills, commitment and facilitating factors. For every aspect, there are several factors to be considered, as shown in Figure 5-2. This model pays more attention on software of a company, but limited attention on the hardware and environment of a company, so the factor set is not complete too.

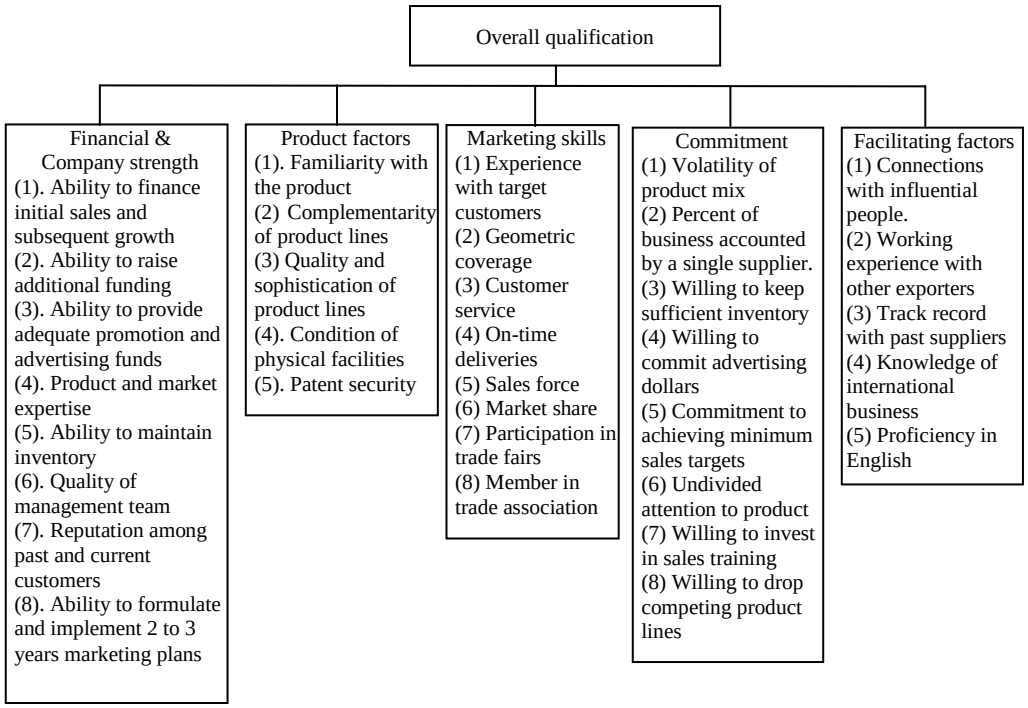


Figure 5-2 Criteria for evaluating foreign distributors

As there was no systematic analysis in the above literatures, the resulted factor set is not complete. Incomplete factor set may cause inaccurate, even wrong evaluation. Next, we try to determine a relatively complete factor set by systematic analysis.

5.2.2 A relatively complete factor set for evaluating a distributor

All factors related to the design objective set in Chapter 4 need to be considered when evaluating a distributor. These factors can be divided into two parts: internal factors and external factors. A distributor may be a firm or other kind of economic entity. Internal factors are referred to the factors that are associated with the internal operations of a firm, e.g. maintaining inventory, transporting materials etc. External factors are referred to the factors that are associated with the interactions between the firm and its surrounding marketing environment. Both will be discussed next.

5.2.2.1 Internal factors

Like any other economic entities, the internal status of a firm can be evaluated from two points of view: hardware and software. Normally, hardware is referred to those physically visible objects, such as infrastructure, human resource, etc. Software is referred to physically invisible factors, such as management, commitment, etc. Based on this analysis, all internal factors can be categorized into hardware factors and software factors.

Evaluation of Possible Distributors

Hardware factors. The hardware of a firm is mainly composed of three parts: infrastructure, human resource and financial capability. As this firm will possibly act as a distributor, its main activities are maintaining inventory, delivering product, and communication. So, the infrastructure we care about includes inventory carrying facilities, transportation facilities, and communication system. In what follows, all hardware factors will be introduced individually.

- *Inventory carrying facilities*. An inventory carrying facility is evaluated from three aspects: its capacity, cost, and reliability. Inventory carrying capacity can be indicated by the floor space of the firm. Cost is measured by the carrying cost per unit floor space. During running in a firm, a part of the floor space may be broken. If less part of the floor space is broken, we say that the facility is more reliable; vice versa. So the reliability of inventory carrying facility is indicated by the percentage of broken floor space during a given period (e.g. one year).
- *Transportation facilities*. Here, the transportation facilities are mainly referred to the facilities used for delivering product. A transportation facility can also be evaluated from three aspects: capacity, cost, and reliability. Transportation capacity is indicated by the firm's throughput. Cost is represented by the product delivery cost per day. During delivering products, some of the transportation facilities (e.g. vehicles) may be broken. If fewer facilities are broken, we say that the transportation system is more reliable; vice versa. So the reliability of transportation facilities is also indicated by the percentage of broken facilities during a given period.
- *Communication system*. The communication system for a firm is evaluated from two aspects: communication methods and cost. The normal communication methods are telephone, fax, Intranet and Internet. Cost is measured by the communication cost per unit time (e.g. per month) for a given workload.
- *Human resource*. Human resource is one of the most important resources for a distributor. The human resource for a firm can be evaluated from two aspects: quantity and quality. The number of employees is associated with the scale of a firm and cost for salary; quality of employees is indicated by the percentage of educated (or trained) workers.
- *Financial capability*. Financial capability is another hardware factor for a firm. As the firm will act as a distributor, three types of financial capability are mainly concerned: capability to finance sale, capability for additional funding and capability for funding advertisement.

Software factors. A firm possessing good hardware can not guarantee successful operation. The software of this firm is equivalently important as its hardware. Management is the basic type of software factor for a firm. As the firm will act as a distributor for a given product, its commitment to the host enterprise, its familiarity and technical support to the product also need to be evaluated. In what follows, we illustrate these software factors individually.

- *Management*. The management for a firm can be evaluated from three aspects: the firm's efficiency, management cost, and safety. The efficiency of a firm can be reflected by the throughput per employee. For management costs, we mainly care

about transaction costs here. Safety can be indicated by the number of accidents for a given period. As the firm is used to distribute a special product, the percentage of damaged product is also considered as a type of safety.

- *Commitment*. If a firm does not commit to the host enterprise, it can not distribute the product efficiently even if it has this capability. For a distributor, normally, three types of commitment are evaluated: willing to keep appropriate inventory, willing to provide advertisement fund, and willing to invest in training employees.
- *Product factors*. When the host enterprise wants to employ a firm to distribute its product, it hopes that the firm is familiar with the product, or has experience in selling similar products. For some products, producer needs to provide service after selling. To reduce supporting cost, the host enterprise hopes that the firm has technical capability to provide such services. Product factors mentioned here are concerned with these items.

5.2.2.2 External factors

For a distributor, two kinds of external factors need to be considered: its location and marketing environment. The former one is related with the product delivery, and the later one is associated with product sale.

Location factors. Following two factors are normally viewed as location factors:

- *Unit product delivery cost for the firm*. This factor includes two sub-factors: distance between the host enterprise and the firm, and the freight balance for that firm. In order to reduce product delivery cost, the host enterprise hopes to select those firms with lower unit product delivery cost.
- *Traffic access to the firm*. Normally, there are four kinds of possible traffic access to a firm: highway, waterway, railway and aircraft. Which one is preferred depends on the product property, and this preference will be reflected in the inference rules used when evaluating this possible distributor.

Marketing environment. As the firm will act as a distributor for a given product, its marketing environment is vital for the success of distribution chain management. For a marketing environment, following issues are normally evaluated:

- *Buying power*. It is indicated by the net personal income (= gross personal income – personal taxes – non-tax payment) [Min et al, 1999]. Buying power is a measure of market ability to buy. Host enterprise will select those firms located in the region with higher buying power.
- *Geographical coverage*. It is expressed as the number of possible customers covered by the firm. Obviously, larger geometric coverage means larger market potential. Market potential is a crucial factor for long term success of the distribution chain management.
- *Market share of the firm.* Here, it is referred to the general market share of the firm in its customer zone. This is a synthetic reflection of reputation, service, etc. for the firm. In Chapter 6, we will talk about market share again. At that time, it

will be referred to the market share only for the products to be distributed. By this market share, the exact customer demand at this retailer can be calculated.

- *Number of competitors*. Competitors are referred to those firms that sell the same product as the one produced by the host enterprise. Obviously, this is a negative factor for the enterprise to sell its product. The more competitors are there in the region, the stronger the competition will be when the enterprise initiates its sale.
- *Tax reduction*. It can be viewed as an incentive factor from the local government. Tax reduction can directly reduce cost for the enterprise, and to some extent, it also means the law support to the product to be distributed.
- *Number of skilled labors*. If the region where the firm is located can provide enough skilled labor, the firm may employ high quality workers. This is important for the further development of this firm.

Now, about 30 factors are identified to be considered when evaluating a distributor. As stated in [Min et al., 1999], AHP is an effective tool for dealing with decisions involving a large number of factors with different scales. Obviously, it is the appropriate tool to organize all these factors. The organized factors are shown in Figure 5-3. Each factor listed at the rightmost tier of Figure 5-3 can be indicated by a parameter. In next section, we will illustrate how to acquire values on these parameters from possible distributors.

5.3 Acquiring Information from Possible Distributors

In the previous subsection, factors needed to be considered when evaluating a possible distributor are identified. Before beginning the evaluation process, parameters for these factors need to be acquired from possible distributors. All possible distributors are distributed in geography, and both the number of possible distributors and the scale of information needed for one possible distributor are large. This raises a question: how to acquire the large amount of information from distributors efficiently and economically? In this subsection, we will mainly illustrate a mobile agent based information acquisition system, and then a prototype is given to show the designing principle.

5.3.1 Introduction

There are several ways for an enterprise to acquire information from distributors. For example, it can send a man (a real agent) to collect data, or alternatively, it can acquire information by telephone, fax etc. Internet provides another alternative for acquiring information from possible distributors. For traditional information acquisition methods (e.g. telephone, fax, etc.), it is unnecessary to explain them here. In what follows, we will mainly illustrate how to acquire information via Internet.

To acquire information via Internet is not a new idea. The normally used methods can be divided into asynchronous (the message sender proceeds to next task without waiting for the reply from the message receiver) and synchronous (not proceed until receiving the reply) approaches. Among asynchronous messaging methods, E-mail is the most familiar one for us. As it is easy to be used, here, we do not mention any more on it.

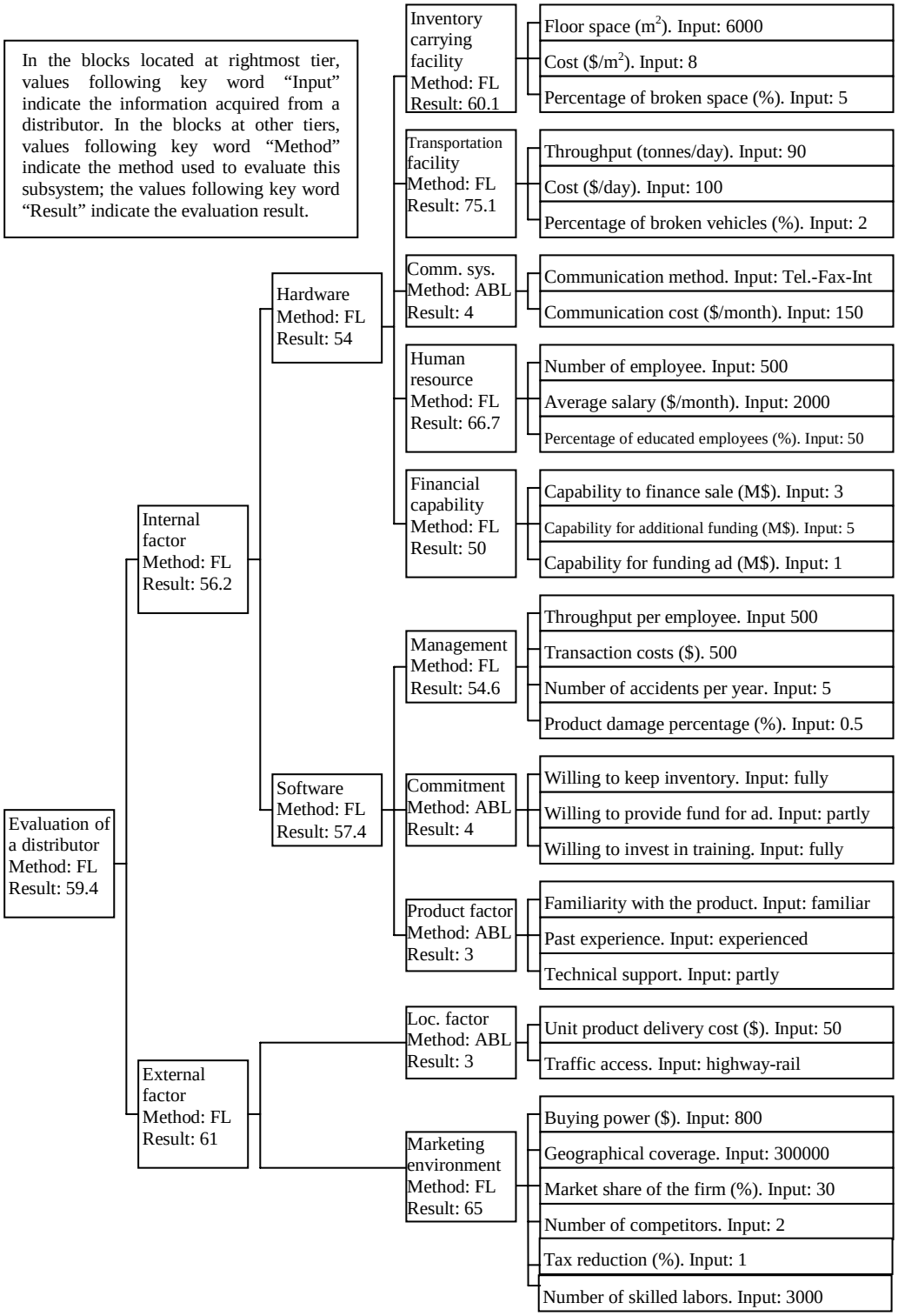


Figure 5-3 A hierarchical representation of factor set for evaluating a possible distributor, and the evaluation result for one distributor.

Evaluation of Possible Distributors

For synchronous messaging methods such as Sockets, HTTP, CORBA/IIOP, etc., most of them take request-reply model, i.e. the host enterprise sends a request to query information to a potential distributor, and then the distributor replies, such process proceeds until the host enterprise gets all information needed, as shown in Figure 5-4(a). In practice, the host enterprise needs not only to acquire information from distributors, but also possibly to discuss, even negotiate with the distributor, so the interaction between them may be tremendous, and the cost for Internet's band-width is large.

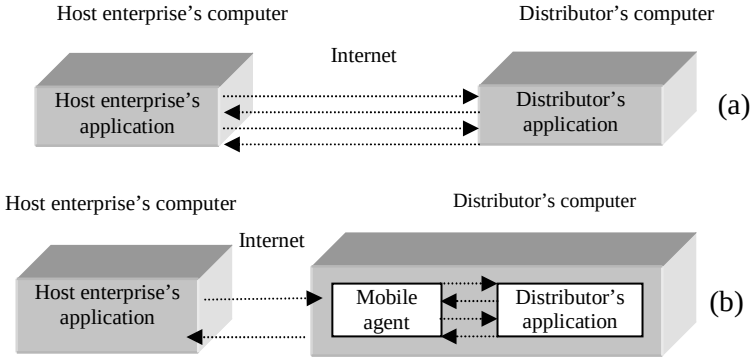


Figure 5-4 Interaction between host enterprise and distributor
(a): request-reply model. (b): mobile agent model

Mobile agent changes the way of interaction, as shown in Figure 5-4(b). This technology allows an agent in the form of program code, data, and execution state to be packaged into a message and sent across the network to a remote computer [Dagupta, et al., 1999]. So the interaction mainly takes place inside the distributor's computer. Given a "thin" agent, mobile agent can remarkably reduce the cost of band width, and provide reliable message delivery and delivery confirmation by its security management system. In this dissertation, we will use mobile agent as an alternative method to acquire information from potential distributors.

Several programming languages such as Java, C++, etc., can be used to develop a mobile agent. Among them, Java provides following unique features:

- *Platform independency*. Java is designed according to "writing once, run anywhere" rule, i.e., the application developed for a specific platform, say UNIX, can also run on other platforms such as Windows NT or Mac OS. As the mobile agent designed to acquire information from distributors will run on remote computers, which are possibly based on different operating system, this property is crucial for us.
- *Security*. With several layers of security control protection against malicious code, Java is claimed to be one of the most secure languages. Security is always an important issue for a network based application. Any attack caused by malicious code may damage the system, or steal business secrets. So this property is also very important for us.
- *Simplicity*. Java also offers cleaner and simpler code (than C++) and component model. This property makes it easy to develop a large system.

For these reasons, Java based mobile agent is used to develop our information acquisition system. Several Java based agent development systems are available in the market, such as Concordia, Aglet, Voyager, etc. For its simple API (Application Programming Interface) and security management, Concordia is selected as the development platform to design our mobile agent system.

5.3.2 Structure of the mobile agent based information acquisition system

The structure of the mobile agent based information acquisition system is shown as Figure 5-5. Possible distributors are divided into several regions, say *N* regions, according to their geographical location. The host enterprise dispatches one mobile agent for each region. The mobile agents visit every distributor in their corresponding regions, acquire information, and then go back to report all acquired information to the host enterprise's stationary agent. Each distributor has a stationary agent to receive the mobile agent and interact with it. Every agent is actually a program, and all these processes are finished on Internet according to following procedure:

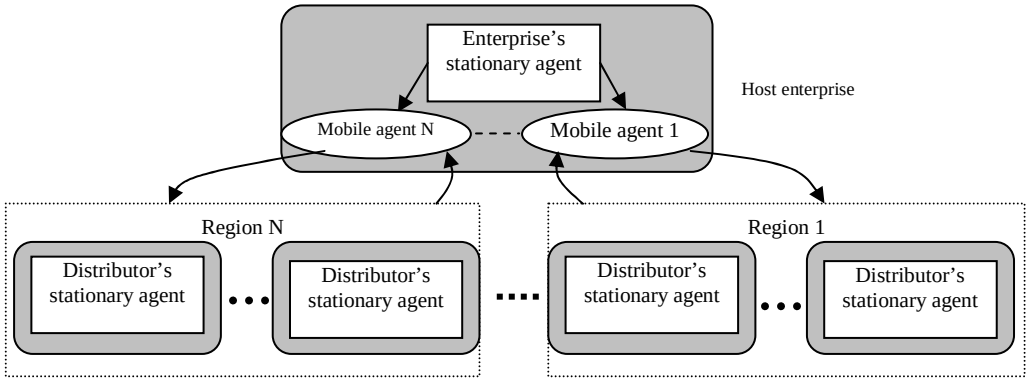


Figure 5-5 Structure of the information acquisition system

(1) Initialization. Before dispatching the mobile agent, the host enterprise needs to prepare:

- *URLs (Uniform Resource Locator) of potential distributors' computers*. As the information acquiring process is finished on Internet, the destinations of mobile agents are expressed as URLs. An enterprise can locate the URLs of potential distributors by search engine or other approaches.
- *Variable table*. That is used to express what kind of information to be acquired by the mobile agent. Actually, the items in this table are the factors shown at the rightmost tier of Figure 5-3.

(2) Host enterprise dispatches mobile agents to distributors. After determining where to go and what to be acquired for mobile agents, the host enterprise stationary agent creates the mobile agents, and dispatches them to potential distributors by the tool provided in Concordia package.

(3) Mobile agents acquire information from distributors. After dispatched, the mobile agents begin to travel on Internet, arrive at destinations and execute pre-defined methods to acquire information according to the prepared variable table. At that time, the mobile agent may do some initial analysis on the information, interact or negotiate with the distributor's stationary agent.

(4) Mobile agents report information to host enterprise's stationary agent. After visiting all distributors, the mobile agents travel back to the host enterprise, execute pre-defined method to report the acquired information to the stationary agent.

(5) Host enterprise's stationary agent stores acquired information into its database. The host enterprise's stationary agent receives all mobile agents, and stores the acquired information into its database for future use.

5.3.3 Testing prototype

To illustrate the design principle, we realized a testing prototype with one host enterprise's stationary agent, a mobile agent and several distributors' stationary agents in our laboratory, which can be sketched as Figure 5-6. This prototype is mainly composed of three modules: host enterprise's stationary agent, mobile agent and distributor's stationary agent. Next, we will introduce them individually.

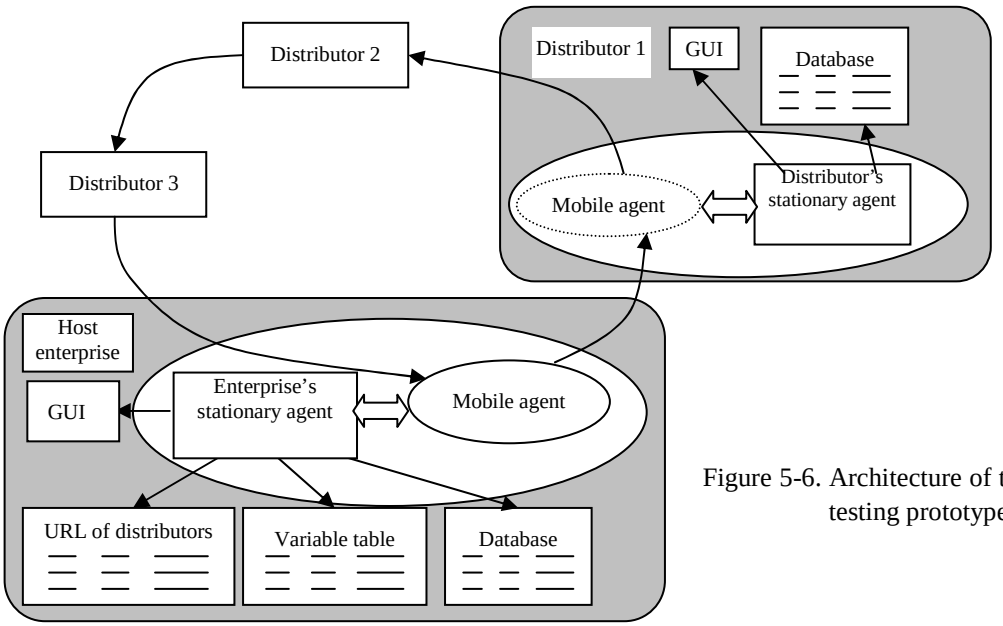


Figure 5-6. Architecture of the testing prototype

(1) Host enterprise's stationary agent.

It runs on the host enterprise's computer, and acts as the bridge between mobile agent and other parts of the computer. The main functions for this stationary agent are:

launching mobile agents and managing database. Concordia provides the following three approaches for launching a mobile agent:

- *Agent launch wizard*. This is an interactive process for launching a mobile agent. The wizard prompts operator to specify the mobile agent class file and related classes to be packed with the agent, input the destinations for the mobile agent, and specify method(s) to be executed remotely. Then the agent can be launched immediately.
- *Command line tool*. The launching process is very similar to the one in "Agent Launch Wizard". The only difference is that, here, command line rather than wizard is used.
- *API*. Concordia provides methods for assigning itinerary for mobile agent, determining method(s) to be executed remotely, and launching the mobile agent.

To realize an automatic launching process, in this testing prototype, we use API to launch a mobile agent, and part of source code is shown as Figure 5-7

```
import COM.meitca.concordia.*;                                  // import the Concordia package.

…
Itinerary itinerary = new Itinerary();                          //create an Itinerary object.
itinerary.addDestination(new Destination ("distrbutor_1", "query")); // specify the first destination "distrbutor_1"
                                                                  and method to be executed there named as
                                                                  "query".
itinerary.addDestination(new Destination ("distributor_2", "query")); // specify the second destination and method..
…
agent.setItinerary(itinerary);                                  //set the itinerary for the mobile agent.
…
agent.launch();                                                 //launch the mobile agent
```

Figure 5-7. Part of source code for launching a mobile agent in Concordia

When the mobile agent comes back to host enterprise, it reports the acquired information to enterprise's stationary. Then the stationary agent inserts the reported information into its database. Part of the source code for this function is shown in Figure 5-8.

```
import java.sgl.*;                                              //import the SQL package
…
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver()); //load the driver
Conn=DriverManager.getConnection("jdbc:oracle:thin:@(description=(address=(host=ipto.hin.no)(protocol=tcp)(
port=1521))(connect_data=(sid=stud)))", "homa", "hongze");          //make the connection
stmt = conn.createStatement ();                                 //create the statement
…
stmt.executeUpdate("insert into InvFacilities ("+key") values ("+ value +")"); //insert value into database
…
```

Figure 5-8. Part of source code for inserting value into database

Alternatively, we can also put this part of code in mobile agent, but for the following two reasons, we put them in the stationary agent:

Evaluation of Possible Distributors

- To make the mobile agent as "thin" as possible. As the mobile agent will travel on Internet, this arrangement can reduce the Internet cost.
- Some sources for the database management, such as Driver, etc., are platform dependent. When the mobile agent comes back to the enterprise's computer or visits a distributor's computer, maybe, it does not know what kind of Driver is installed on that machine, so it is better to make this part of code as local, not global.

(2) Mobile agent.

Three functions are designed for the mobile agent: accessing database, parsing XML (eXtensible Markup Language) documents, and extracting information from command line. Next, let's discuss these three functions in detail.

Accessing database. Traditionally, database is the main method for an enterprise to manage its information, so, accessing database is the basic function for a mobile agent. If the needed information is stored in distributor's database, this function can be used to acquire it. For the reasons mentioned above, the function itself is installed on the Concordia server of the distributor's computer, and expressed as a service for mobile agent. When the mobile agent visits this distributor, it passes variable table to the service, and calls the service to acquire information. As the source code for this function is similar to the one shown in Figure 5-8, for brevity, we do not list it here.

Parsing XML document. Similar to HTML (HyperText Markup Language), XML is developed for Web application. The main difference between XML and HTML is that: in HTML, tag set is fixed, and it is originally designed to present information on screen for people; in XML, the tag set is flexible and extensible (as its name implies), and the new rules can be created, agreed upon, and specified in the accompanying DTD (Data Type Definition) file. So XML is more powerful to treat data, and it can be used conveniently to exchange data between Web applications. As stated in [Ray, 2001], XML is becoming the main format for common data exchange between databases, or message exchange on Internet. For this reason, the function for parsing XML document is also developed in the mobile agent. If the needed information is stored in an XML document on distributor's computer, this function can be called to parse the document and extract information from the parsed XML document.

A sample of XML document is shown as Figure 5-9(a), which is used to express the information of an inventory carrying facility. Its accompanying DTD file is shown as Figure 5-9(b).

```
<?xml version="1.0"?>
<!DOCTYPE facForInvData SYSTEM
"facForInvData.dtd">
<facForInvData>
  <key>floorSpace</key>
  <value>5000</value>
  <key>cost</key>
  <value>10</value>
  <key>reliability</key>
  <value>2</value>
</facForInvData>
```

(a).

```
<!ELEMENT FacForInvData(key,
value)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

(b).

Figure 5-9 An example of XML document *facForInvData.xml*, and its accompanying DTD file *facForInvData.dtd*

Both the XML file and DTD file are stored in distributor's computer. The task for mobile agent is to read and parse the XML file, then extract data from it. XML for Java provides SAX (Simple API for XML) parser and DOM (Document Object Model) parser. SAX parser is selected when the document is very large, and only a few elements are needed. On the other hand, if the document is not large, and most of the information is needed, DOM parser is used. Obviously, DOM parser is more appropriate for our application. The source code to parse a XML document by DOM parser is shown as Figure 5-10.

```
import com.ibm.xml.parser.Parser;  //import the DOM Parser from XML for
Java package
import java.io.FileInputStream;
import java.io.InputStream;
import org.w3c.dom.Document;
…
String myfile = "facForInvData.xml";  //specify the XML file
facForInvData.xml
InputStream is = new FileInputStream(myfile);  //create an InputStream
object to open the file
Parser parser = new Parser(myfile);  //create an object of DOM Parser
Document doc = parser.readStream(is); //read the file and parse it
…
```

Figure 5-10 Sample of source code to parse a XML document: *facForInvData.xml* by DOM parser

After parsed, the XML document *facForInvData.xml* becomes a DOM tree structure, as shown in Figure 5-11. The source code shown in Figure 5-12 extracts information from such DOM tree structure, and put them into "Hashtable" as key-value pairs. The mobile agent takes this "Hashtable" back to host enterprise, and reports it to the host enterprise's stationary agent.

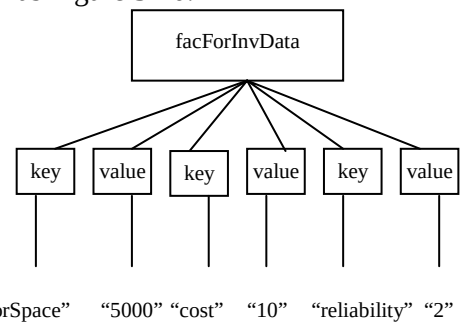


Figure 5-11 DOM tree parsed for file *facForInvData.xml*

```
import java.util.Hashtable;
import org.w3c.dom.*;
…
Hashtable hash = new Hashtable();
String key = null, value = null;
for (Node child = doc.getDocumentElement().getFirstChild();child!=null; child=kvchild.getNextSibling()) {
       if (kvchild instanceof Element) {
          if (kvchild.getNodeName().equals("key")) {
            key = makeChildrenText(kvchild);  //extract “key” value
           }
          else if (kvchild.getNodeName().equals("value")) {
            value = makeChildrenText(kvchild);  //extract “value” value
            if (key!=null) {
              hash.put(key, value);  //put the key-value pair into hash table
              key=null;
            }
           }
         }
     }
…

private static String makeChildrenText(Node node) {
   StringBuffer buffer = new StringBuffer();
   return makeChildrenText1(node, buffer);
  }

private static String makeChildrenText1(Node node, StringBuffer buffer) {
   for (Node ch=node.getFirstChild(); ch!=null; ch=ch.getNextSibling()) {  //scan all its child nodes
     if (ch instanceof Element || ch instanceof EntityReference) {
      buffer.append(makeChildrenText(ch));  //recursively call method makeChildrenText(Node node)
     }
     else if (ch instanceof Text) {
     buffer.append(ch.getNodeValue()); //if it is a text node, just get the value.
     }
   }
   return buffer.toString(); //return the value
  }
}
```

Figure 5-12 Part of source code to extract information from a DOM tree structure and put it into “Hashtable”

Extracting information from command line. It is possible that the acquired information from database and/or XML document is not complete, so the third function, extracting information from command line, is called. In this function, first, a prompt is presented to operator of the computer, asking him/her to enter the value for a variable. Then the system directly reads the value from keyboard, and put them into “Hashtable” as key-value pair. This “Hashtable” will be reported to host enterprise’s stationary agent when the mobile agent travels back to host enterprise. The source code for this function is shown in Figure 5-13(a), and the corresponding running result is shown in Figure 5-13(b).

```
import java.io.*;
import java.util.Hashtable;
…
Hashtable hash = new Hashtable();              //create an object of Hashtable
BufferedReader stdin = new BufferedReader(new
 InputStreamReader(System.in));
                                               //ready to read input from key board
System.out.println("please enter " +key);    //present prompt to operator
string = stdin.readLine();                     //read input from key board
hash.put(key, string);                         //put key-value pair into Hashtable
value=Integer.parseInt((String)hash.get(key));
System.out.println(key+" is "+value);       //print the result to operator for
                                               checking
```

(a)

```
Please enter cost: 10
Cost is 10
```

(b)

Figure 5-13. Part of source code for extracting information from command line.

For limited time, the negotiation function between mobile agent and distributor's stationary agent is not realized in this testing prototype, but it is necessary in practice, it is left as future work.

(3) Distributor's stationary agent.

Similar to enterprise's stationary agent, distributor's stationary agent is also a Concordia server. Besides receiving mobile agent, it also provides the service of accessing its own database. When mobile agent visits this distributor, it invokes this service, acquire the information from distributor's database, and bring back to host enterprise.

Although this testing prototype is realized in our laboratory, in principle, it can be realized in reality between host enterprise and potential distributors which are distributed all over the world, because the prototype only works with the IP (Internet Protocol) addresses of different computers on Internet when the mobile agent visits different distributors. The only requirement for distributors is that, they must run Concordia server, and install needed services on their computer.

In this section, a mobile agent based information acquisition system is developed to collect data from possible distributors. Three functions are developed for the mobile agent: accessing database, parsing XML documents, and extracting information from command line. If the mobile agent is authorized to access distributor's database, the first function is activated; if the possible distributor puts some information on its web site in XML form, then the second function is called. The mobile agent can also interact with the operator of the distributor via the third function. If none of above situations is applicable, or the information acquired by mobile agent is not complete, other methods such as telephone, fax, E-mail can be used to collect data from possible distributors.

5.4 Distributor Evaluation

After acquiring the needed information from possible distributors, we come to the third stage of distributor evaluation module: evaluating the distributors. In this subsection, first, the existing evaluation methods are analyzed. Then, after a survey of possible methods, our new evaluation method is put forth. At last, a case study is given to illustrate how to use this new method in evaluating a distributor.

5.4.1 Analysis of existing evaluation methods

Cavusgil et al. [1995] developed an expert system, called DISTEVAL, in evaluating a foreign distributor. In this system, each criterion is assigned a weight to express its importance and score to express its priority, and both are determined via the knowledge acquisition process. Based on the sum of weighted scores, the evaluation from DISTEVAL produces scores ranging from 0 to 100 for each of the dimensions, which can be used to interpret the evaluation results qualitatively.

Min et al. [1999] used AHP model to evaluate a hybrid production/distribution facility. In this model, each criterion is also assigned an importance weight and a priority score, which were derived from pairwise comparisons and management team's judgment. At last, the sum of weighted scores produces the synthesized priority for this production/distribution facility.

The core of existing evaluation methods is scoring. Normally, scoring method suffers following shortcomings:

- As both importance weight and priority score are assigned by expert (human being), the result is largely affected by subjective judgement.
- As main parameters are determined subjectively, it is difficult to use computer application to realize the evaluation process.
- The final result is only a linear combination of all criteria. Generally, the relationship between each criterion and the evaluation of a distributor is unnecessary to be linear, so the scoring method may cause error in evaluating a distributor.

As mentioned above, the existing evaluation method (scoring method) can neither reflect the relationship between factors and the evaluation of a distributor properly, nor fulfill our requirement of automatic distributor evaluation, so we need to find a new method for evaluating a distributor.

5.4.2 Requirements for the new method

The requirements for the new method are analyzed from three aspects: input, mapping process and output.

(1) Input

As shown in Figure 5-3, there are four types of input variables for this evaluation module:

- Quantitative variable, such as “floor space”, etc.
- Qualitative-I variable. This kind of variables are qualitative, but they can be indicated by the system perception of a quantitative value. For example, for the variable “reliability of inventory carrying facilities”, it can be indicated by the percentage of broken floor space.
- Qualitative-II variable. This kind of variables is qualitative, and they are difficult or almost impossible to be indicated by quantitative values. For example, for the variable “familiarity with the product”, the possible values for it are: ‘not-familiar’, ‘partly-familiar’, ‘familiar’, etc. It is difficult to indicate it by a quantitative value, but easy by multi-value logic value.
- Logic variable. Some variables are originally logic variable. For example, for the variable “communication-method”, the possible values are: ‘no-telephone_no-fax_no-Internet_no-Intranet’, ……, ‘telephone_fax_Internet_Intranet’, i.e. it can only be expressed by logic variable.

The characteristics of all input variables for this evaluation module are shown in Table 5-1. The new evaluation method must have capability to handle all these four kinds of variables.

Table 5-1 Characteristics of all input variables in the evaluation module

Variable	Character			
	Quantitative	Qualitative-I	Qualitative-II	Logic
Floor space (m^2)	√			
Inventory carrying cost ($/$m^2$)	√			
Reliability of inventory carrying facilities (indicated by percentage of broken space) (%)		√		
Throughput (tonnes/day)	√			
Product delivery cost ($/day)	√			
Reliability of product delivery facilities (indicated by percentage of broken facilities) (%)		√		
Communication method				√
Communication cost ($/month)	√			
Number of employees	√			
Average salary ($/month)	√			
Education status for employees		√		
Ability to finance initial sales (M$)	√			
Ability for additional funding (M$)	√			
Ability for advertising fund (M$)	√			
Efficiency (throughput per employee) (tonnes)	√			
Safety (number of accidents per year)		√		
Transaction costs ($)	√			
Percentage of damaged product (%)	√			
Willing to keep enough inventory			√	
Willing to provide enough fund for advertising			√	
Willing to invest in employee training			√	
Familiarity with the product			√	
Past experience with similar product			√	
Technical support			√	
Unit product delivery cost ($)	√			
Traffic access				√
Buying power ($)	√			
Geographic coverage	√			
Market share (%)	√			
Number of competitors	√			
Tax reduction (%)	√			
Number of skilled labour	√			

(2) Mapping process

The evaluation process can be viewed as a mapping process from all factors to the evaluation of a distributor. This mapping process has the following properties:

- The relationship between factors and evaluation of a distributor is normally not linear.
- Actually, the evaluation process is a human-thinking like process. It is usually involved with expert knowledge, and the expertise is expressed by linguistic or logic rules.

These properties require that the new method can not only reflect the non-linear relationship between input and output, but also integrate the expertise into the mapping process.

(3) Output

The new method needs to output a value, which can be used to reflect the evaluation of a distributor, and compare between different distributors.

5.4.3 A survey of possible methods

As mentioned above, evaluation process can be viewed as a mapping process from all factors to the evaluation of a distributor, so, all methods related to mapping can be viewed as possible ones. Before developing the new evaluation method, a survey of these possible methods is given as below.

(1) Mathematical method

Mathematical function is an intuitive choice for a mapping process, but it suffers following disadvantages: (1) Normally, mathematical function is good at treating quantitative variables, but not at qualitative and logic variables. (2) It is difficult to integrate expertise (linguistic or logic rules) into a mathematical function. So, it is eliminated from our choice scope.

(2) Artificial intelligence methods

The possible artificial intelligence methods are ANN and FL.

ANN. Theoretically, ANN can fulfill almost all the requirements mentioned in subsection 5.4.2, but in practice, it is difficult to realize it. ANN needs a huge number of data to train it. For one distributor, it needs training data to describe all possible states, and the host enterprise needs to get the necessary data from all potential distributors to train the ANN. So, we can imagine how large the scale of data is, and how difficult it will be to acquire these data. At the same time, it is difficult to change

the trained ANN model when the decision makers change their evaluation rules. Based on this analysis, ANN is also eliminated from our choice scope.

FL. For the input variables, FL is good at handling quantitative and qualitative-I variables. For the mapping process, FL can fulfill all the requirements. At the same time, FL can also output a value to indicate the evaluation of a distributor. Especially, in practice, FL is realizable in our module: we only need to define if-then rules to express the expertise. So, FL is one of the possible choices for our evaluation module. The remained problem is: how to handle the qualitative-II and logic variables?

(3) ABL (Array Based Logic)

As the remained problem is related with logic variables, we will explore logic methods. Davidrajuh [2001] stated that, ABL is a promising technology in dealing with large scale logic inference problems. Comparing with other logic inference technologies, ABL possesses following special advantages:

- It can deal with three kinds of variables: nominal variable (Boolean and multi-value logic variable), ordinal variable (e.g. coordinates *[2,3]*, *[3,4]*, etc.), and interval variable (e.g. interval like 'greater than *50* and less than *100*', etc.).
- For other logic inference technologies, if there are M variables and N values for each variable, there will be M^N combinations for the system, this is called 'combination explosion'. The exponential growth of combinations with increasing number of variables makes the inference process intractable. ABL solves this problem by compressing M^N subspaces into linear $M \times N$ representations, and the resulted system is complete and compact. That is why it is good at dealing with large inference problems.

In distributor evaluation module, there are a lot of variables, and these variables may be nominal, ordinal, or interval. Obviously, for such inference module, ABL is the appropriate technology. Especially, the advantages mentioned above can complement the disadvantages of FL in dealing with input variables for an evaluation module. For the mapping process, ABL uses logic rules (premises) to express expertise, and it can output logic value to indicate the evaluation of a system. So ABL also locates within our choice scope.

The survey of possible methods is summarized by Table 5-2.

Table 5-2 Summary of the possible methods

Methods	Mathematical function	Artificial Neural Network	Fuzzy Logic	Array Based Logic
Fulfilling requirements for input variables	Not	Partly	Partly	Partly
Fulfilling requirements for mapping process	Not	Fully	Fully	Fully
Fulfilling requirements for output variables	Not	Fully	Fully	Fully
Whether realizable in practice	Not	Not	Yes	Yes

By this survey, following conclusions are achieved: (1) none of methods can fully fulfill the requirements described in subsection 5.4.2. And (2) Both FL and ABL can fulfill a part of the requirements, and they are mutually complement. The second conclusion gives us a hint: whether we can integrate FL and ABL to evaluate a distributor? Next, we will simply introduce these two technologies, and then illustrate the new evaluation method based on this introduction.

5.4.4 Simple introduction to FL and ABL

5.4.4.1 FL

Fuzzy sets theory was developed by Lotfi Zadeh in early 1960s [Zadeh, 1965], and the application of FL for industrial control was first demonstrated by E.H.Mamdani in 1974 [Mamdani, 1974]. Since then, FL has largely been applied in automatic control, inference engine and other areas. Next, the basic concepts, composition of a fuzzy inference module, and realization tool for FL are simply introduced.

(1) Basic concepts in FL

Fuzzy sets. Fuzzy sets are defined versus crisp sets. For crisp sets, the element either belongs to or not belongs to a set totally. For fuzzy sets, it permits the element to belong to a set partially, and the degree of membership is indicated by a number between 0 and 1. If X is the universe discourse and x is a particular element of X, then a fuzzy set A defined on X can be written as:

$$A = \{(x, \mu_A(x))\}$$

Where $x \in X$, $\mu_A(x)$ is the membership function of x.

Linguistic variable. It is a variable whose arguments are represented by fuzzy sets. For example, *floorSpace* is a linguistic variable (as shown in Figure 5-14), its arguments are fuzzy sets: *small*, *medium* and *large*.

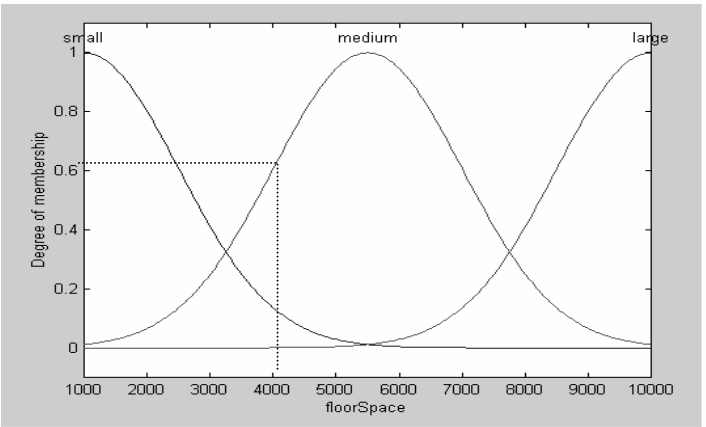


Figure 5-14 Membership function for linguistic variable *floorSpace*

Fuzzy algorithm. Fuzzy algorithm is a procedure for performing a task formulated as a collection of fuzzy

if/then rules [Tsoukalas, 1997]. A fuzzy if/then rule looks like:

IF *floorSpace* is *large*, THEN *invFacility* is *selected*

Where *floorSpace* and *invFacility* are linguistic variables, *large* and *selected* are values for corresponding variables.

(2) Fuzzy inference module

A Fuzzy inference module is composed of three parts: fuzzification, inference and defuzzification (as shown in Figure 5-15).

Fuzzification. Fuzzification is used to convert a crisp value into fuzzy value. The degree of membership is found by the intersection point of an input value on the horizontal axis with the curve defining membership functions. For example, for linguistic variable *floorSpace* mentioned above, when the input value is *4050*, the value for the variable is *medium*, and the corresponding degree of membership is *0.61* (as shown in Figure 5-14).

Inference. Fuzzy inference refers to the procedure to evaluate the linguistic output value based on inputs and fuzzy if/then rules. The commonly used procedures are GMP (Generalized Modus Ponens) and GMT (Generalized Modus Tollens) [Tsoukalas, 1997]. For example, a simple system for selection of inventory carrying facility can be depicted as:

IF *floorSpace* is *large*, THEN *invFacility* is *selected* ELSE
IF *floorSpace* is *medium* OR *small*, THEN *invFacility* is *rejected*

Given the fuzzy value for *floorSpace*, the output value for *invFacility* can be computed.

Defuzzification. Defuzzification is the process to convert fuzzy value back to crisp value. After computing the output (fuzzy value) by combination of input fuzzy value and if/then rules, we need to change the fuzzy value into crisp value to understand its meaning. The commonly used defuzzification methods are COA (Center of Area), COS (Center of Sums) and MOM (Mean of Maxima).

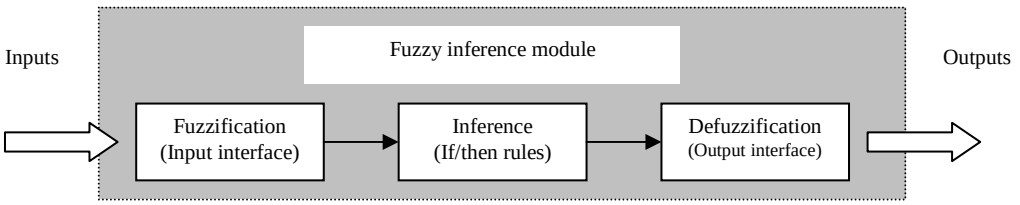


Figure 5-15 Composition of a fuzzy inference module

(3) Realization tool

The realization tool for FL, Fuzzy Logic Toolbox, has been developed in MATLAB environment. Fuzzy Logic Toolbox is a collection of functions that allow the developer to create, edit and implement the fuzzy inference system. The tool box provides three categories of tools:

- *Command line functions*. They are made up of functions that can be called from the command line. Actually, most of these functions are MATLAB M-files or series of MATLAB statements that implement specialized fuzzy logic algorithms.
- *Graphical, interactive tools*. These tools allow the developer to access the functions through GUI (Graphical User Interface). The GUI based tools also provide an environment for fuzzy inference system design, analysis and implementation.
- *Simulink blocks and examples*. This category of tools is a set of blocks for use with Simulink simulation software. It is specially designed for high speed fuzzy logic inference in the Simulink environment.

The first one, command line functions, will be used to realize the fuzzy inference process in our distributor evaluation module.

5.4.4.2 ABL

ABL is first illustrated in [Møller, 1995], which is founded on a geometrical representation of logic in terms of nested data arrays. Same as FL, we will also introduce ABL from three aspects: basic concepts, modeling procedure, and realization tool.

(1) Basic concepts

Domain. In ABL, the domain of a variable is an ordered finite set with *n* unique items to represent all possible values for this variable.

System. A system can be viewed as a set of interconnected premises. A premise is a logic expression of the inference rule. To some extent, a premise is similar to an if/then rule in FL. For example, a premise may be:

Temperature is greater than *100℃* IMPLIES *light* needs to be switched *on*

(2) Modeling procedure.

In ABL, there are three steps to realize an inference system: define the global domain, model the system, and interact with the environment.

Define the global domain. In this step, the domains of all variables for the system (including input and output variables) are defined. For example, in a temperature-light system, there are two variables: *temperature* and *light*. The possible value for

temperature ranges in interval *[50, 250]*, and the possible states of *light* are: *'on'* and *'off'*, so the global domain for this system is defined as:

temperature	[50, 250]	*light*	{*'on', 'off'*}

Model the system. In this step, a set of premises are defined according to inference rules (or expertise), and then these premises are connected by appropriate logic operators such as *AND*, *OR*, etc. Which connector is used depends on the system's property. The connected premises form an ABL inference module (as shown in Figure 5-16). This step is similar to defining if/then rules in FL inference system. After modeling the system, it is ready to get outputs (i.e. evaluation results) by presenting inputs onto the inference module.

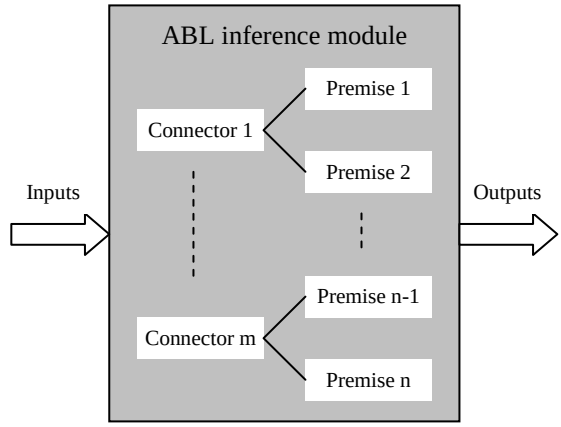


Figure 5-16 An ABL inference module

Interact with environment. Here, the so called environment provides inputs for the inference module, and the interaction with environment can be viewed as the process of presenting inputs onto the inference module. As mentioned above, after presenting inputs onto the inference module, outputs are obtained.

(3) Realization tool

In [Møller, 1995], ABL is realized by APL (A Programming Language). As it is difficult to integrate APL with other tools, Davidrajuh [2001] developed ABL's realization tool, SABL (Structural Array Based Logic), in MATLAB environment. The main functions in SABL package is listed in Table 5-3.

Table 5-3 The main functions in SABL package

Function	Operation
Basic operations	
disjunct	OR operator
conjunct	AND operator
dimp	Direct implication
bimp	Binary implication (equivalence)
Operations for interval variable	
interval	Create an interval variable
union	Union (combination) of intervals
Deductions	
fuse	Remove superfluous axes (variables)
state	Find state of a system (or outputs) for a given input
Utility fuctions	
element	Create a logic variable (or primitive logic element)
assign	Assign new value to a multi-valued logic variable
domain	Assign new domain to an interval variable
join	Combines relations (premiese) through common variables
print	Print out a relation (variables, premises, combined system)

The normal procedure to realize an ABL inference module by this package is:

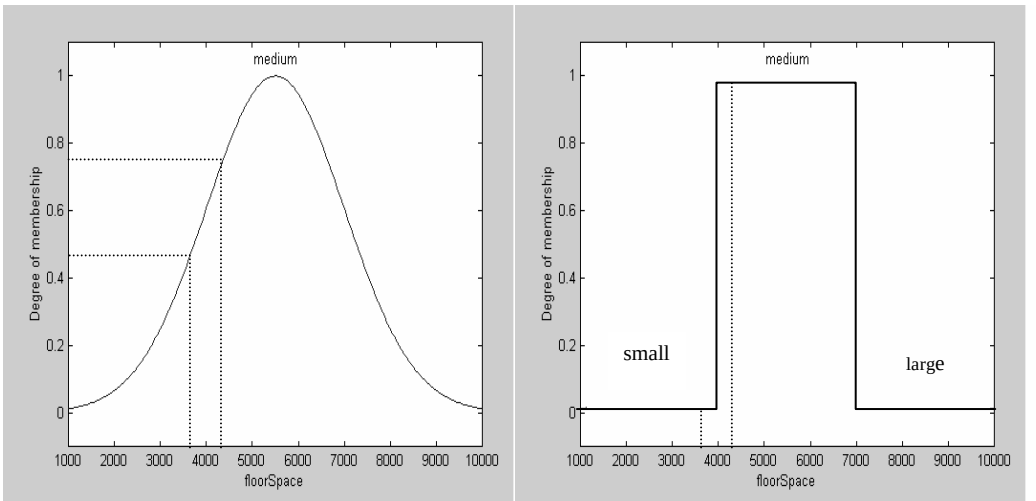
- Define the domain for each variable by *element* function
- Define premises by basic operations *disjunct, conjunct, interval, union,* etc.
- *join* premises to form the inference module
- *fuse* (remove) the common variables
- *assign* values to each input variable to form input vector
- presenting the input vector onto the inference module to compute the output vector by *state* function
- *print* the output vector.

5.4.5 The new evaluation method: integration of FL and ABL

(1) Why we need to integrate FL and ABL

As mentioned in subsection 5.4.3, ABL is an appropriate technology to deal with inference systems with large number of variables, and these variables can be nominal, ordinal, and interval scale. So, theoretically, ABL is capable to treat all input variables and finish the inference process for the distributor evaluation module. But, in ABL, the treatment for some input variables, especially for quantitative variables, is imprecise. This can be illustrated by following example.

Figure 5-17 shows the different treatment of input variable *floorSpace* in FL and ABL. In FL, a crisp value belongs to a set with a degree of membership. For example, when *floorSpace* is *3700*, it belongs to set *medium* with a degree of *0.45*, when *floorSpace* is *4300*, it belongs to set *medium* with a degree of *0.75*. In ABL, a value only belongs to, or does not belong to a set entirely, i.e. the answer can only be 'YES' or 'NO'. For



FL ABL

Figure 5-17. Different treatment of input variable *floorSpace* in FL and ABL

example, in ABL, when *floorSpace* is *3700*, it is *small*, but when *floorSpace* is *4300*, it becomes *medium*! Comparing these two ways, it is obvious that the treatment of input variable *floorSpace* by FL is more precise.

By this analysis, following conclusions are achieved: it is more precise to treat quantitative and qualitative-I input variables by FL, and treat qualitative-II and logic variables by ABL. That is why we have to integrate FL and ABL in evaluating a distributor.

(2) How to integrate FL and ABL

After deciding to integrate FL and ABL in the new evaluation module, a question is raised immediately: how to integrate them? Before answering this question, first, we illustrate the evaluating process for a distributor.

In Figure 5-3, a tree structure is used to depict the distributor evaluation module. We call a branch in this tree structure as a subsystem. For example, “Inventory carrying facility” and its descendent three factors, “Floor space”, “Cost”, and “Percentage of broken space” form a subsystem. When evaluating a subsystem, either FL or ABL is selected according to following rules:

- For the bottom subsystems (located at the rightmost tier of Figure 5-3), ABL is used if there are qualitative-II and/or logic input variables in this subsystem; otherwise, FL is used.
- For all other subsystems, to evaluate a distributor more precisely, FL is used.

When evaluating a distributor, a bottom-up approach is applied, i.e., first the bottom subsystems are evaluated, then the upper subsystems are evaluated according to Figure 5-3. Such process proceeds until the upmost one, “Evaluation of a distributor”, is evaluated. For example, to evaluate the software factor of a possible distributor, first, its three descendent subsystems, “Management”, “Commitment”, and “Product factor”, are evaluated, and then FL is used to evaluate “Software” itself by these three evaluation results.

As both FL and ABL are applied in one evaluation module, following problems are raised. If ABL is used in the lower subsystem (e.g. evaluation of “Commitment”), and FL is used in the upper subsystem (e.g. evaluation of “Software”), then the evaluation result of the lower subsystem (a logic variable) needs to be fuzzified when used as an input in the FL inference module. We call this as the interface from ABL to FL. On the other hand, if FL is used in lower subsystem, and ABL is used in upper subsystem, then the output of lower subsystem (a crisp value) will also be converted into a logic value when it is used as an input in the ABL inference module. This is called interface from FL to ABL. Following rules are used to solve these problems.

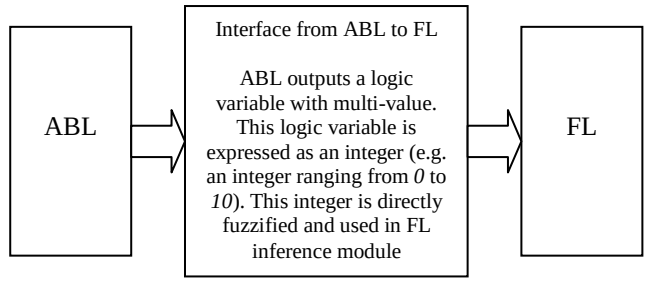


Figure 5-18 Interface from ABL to FL

Interface from ABL to FL. This interface is shown as Figure 5-18. When ABL is used for lower subsystem, and FL is used for upper subsystem, we stipulate that ABL outputs a logic variable with multi-value. This logic variable is expressed as an integer to indicate the evaluation result. When used in FL inference module, this integer is fuzzified directly. For example, as there is logic input variable (communication method) when evaluating "Communication system", ABL is used for this subsystem. We can express the evaluation result of communication system by an integer ranging from *0* to *10* to indicate its satisfaction level (how many levels it is divided into depends on the requirement of precision. More levels mean higher precision, but more complicated inference module). According to the method selection rule mentioned above, FL will be used to evaluate "Hardware", so, we need to fuzzify this integer. Approximately, this integer can be viewed as a quantitative variable, and fuzzified directly (as shown in Figure 5-19).

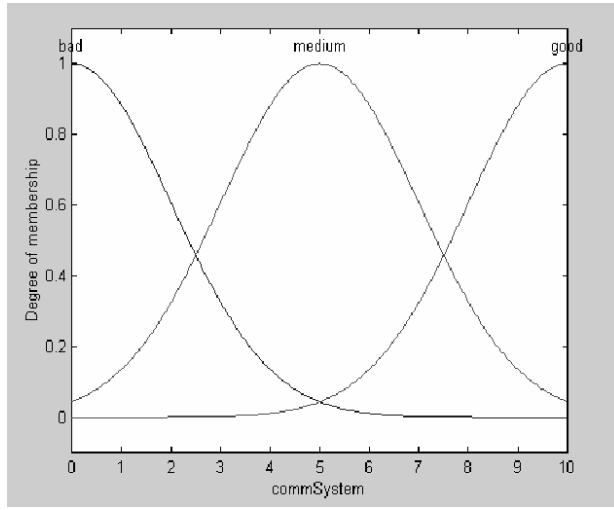


Figure 5-19 Fuzzification of variable c*ommSystem*

Interface from FL to ABL. When FL is used for lower subsystem, and ABL is used for upper subsystem, we need to convert the crisp value into a logic value. This conversion can be simply finished by the way shown at the right part of Figure 5-17. Of course, this treatment will cause imprecision. Fortunately, it is unnecessary to convert a crisp variable into logic one in this evaluation module.

After solving interface problem between ABL and FL, the new evaluation method, i.e. integration of ABL and FL, is developed. For this new method, we summarize it as follows:

- In this new method, both FL and ABL are used in evaluating a distributor.
- When evaluating a subsystem, either ABL or FL is selected according to following rules: for bottom subsystems, ABL is selected if there is qualitative-II

and/or logic input variables in this subsystem; otherwise, FL is used. For other subsystems, FL is selected.

- When evaluating a distributor, a bottom-up approach is applied according to Figure 5-3. First, all bottom subsystems are evaluated, so we can get the evaluation results for factors in the second rightmost tier. Based on these evaluation results for the factors in the second rightmost tier, the factors in the third rightmost tier are evaluated, ……. This evaluation process proceeds until the leftmost one, "Evaluation of a distributor", is evaluated.
- The interface problem from ABL to FL is solved according to the rule shown in Figure 5-18.

As indicated in subsection 5.4.4, the software tools for both FL and ABL are developed in MATLAB environment. This makes it easy to develop computer application for the new evaluation method. Next, we use a case study to illustrate how to apply this new method in evaluating a distributor.

5.4.6 Case study: evaluating a distributor

In this subsection, an example is used to illustrate how to evaluate a distributor in practice. For brevity, here we use the evaluation of "Hardware" to illustrate the whole evaluating process. According to the rules mentioned above, before evaluating "Hardware" itself, its five descendent factors, "Inventory carrying facility", "Transportation facility", "Communication system", "Human factor", and "Financial factor" need to be evaluated first.

(1) Evaluation of "Inventory carrying facility"

"Inventory carrying facility" (*invFacility)* is determined by three variables: "Floor space" (*floorSpace)*, "Cost" (*costInv*) and "Percentage of broken space" (*relia*). All inputs for this evaluation model are quantitative. According to the rule mentioned above, FL is used here. As shown in Figure 5-20, this evaluating process is finished by following steps.

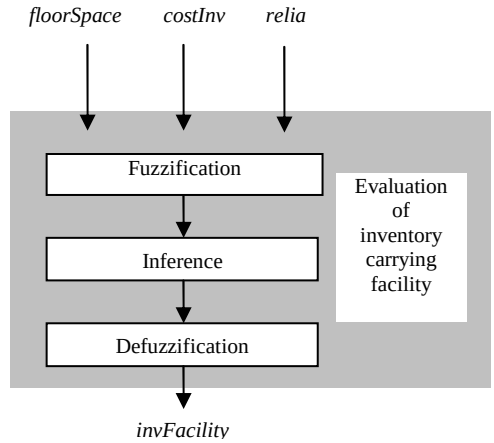


Figure 5-20 Model for evaluation of "Inventory carrying facility"

Fuzzification. Here, Gauss membership function is used to fuzzify *floorSpace* and *costInv*, and trapezoid membership function is used to fuzzify *relia*. The corresponding results are shown in Figure 5-21(a), Figure 5-21(b), and Figure 5-21(c). Output variable *invFacility* is also fuzzified by Gauss membership function (as shown in Figure 5-21 (d)).

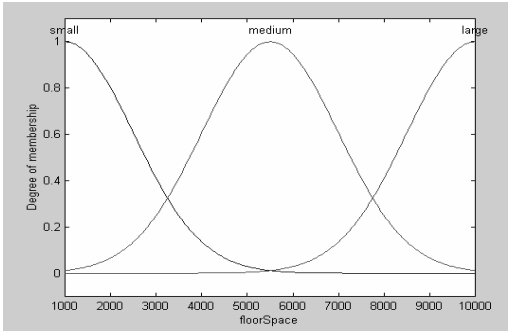


Figure 5-21 (a). Membership functions for input variable *floorSpace*

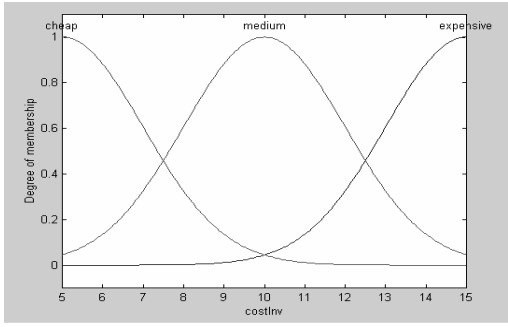


Figure 5-21 (b). Membership functions for input variable *costInv*

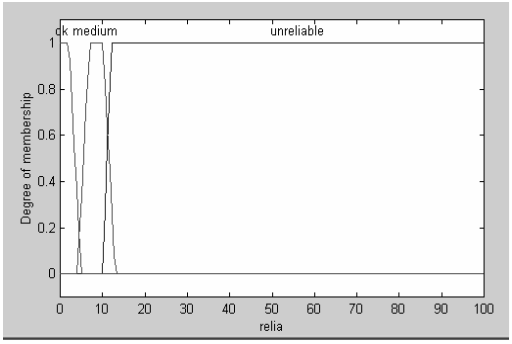


Figure 5-21 (c). Membership functions for input variable *relia*

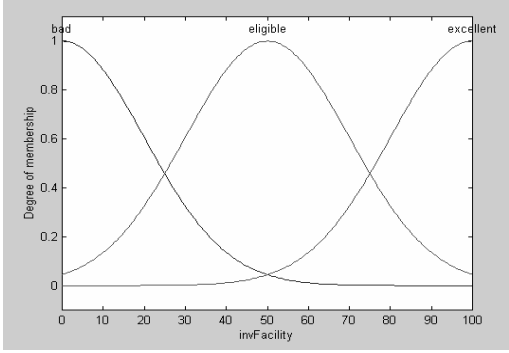


Figure 5-21 (d). Membership functions for output variable *invFacility*

Inference. Inference process is executed based on a set of predefined if-then rules. These if-then rules reflect the decision maker's distributor selection preference. So the inference process can successfully integrate decision maker's role into the evaluation process. For this subsystem, there are three input variables, and for each variable, there are three linguistic values, so a complete inference system needs 3×3×3=27 rules. The detail is referred to rule list in A-1 of Appendix A. An example for these if-then rules is shown as:

IF *floorSpace* is large AND *costInv* is medium AND *relia* is medium, THEN *invFacility* is eligible.

Defuzzification. The result calculated by inference process is fuzzy value. To understand its meaning, this fuzzy value needs to be converted into crisp value. This process is called defuzzification. Here, COA is selected as the defuzzification method.

This inference model has been realized by Fuzzy Logic Toolbox in MATLAB, and the corresponding program is shown in A-1 of Appendix A. A numerical example for the evaluation result is shown in Table 5-4.

Table 5-4 Numerical examples for the evaluation results in evaluating "Inventory carrying facility", "Transportation facility", "Human resource", "Financial factor", "Communication system", and "Hardware".

Inventory carrying facility		Transportation facility		Human factor	
Inputs	Output	Inputs	Output	Inputs	Output
6000	60.1	90	75.1	500	66.7
8		100		2000	
5		2		50	
Financial factor		Communication system		Hardware	
Inputs	Output	Inputs	Output	Inputs	Output
3	50	telephone-fax-Internet	4	60.1	54
5				75.1	
				4	
1		150		66.7	
				50	

(2) Evaluation of "Transportation facility", "Human resource", "Financial factor"

All input variables for these three evaluation models are quantitative or qualitative-I variables. According to the method selection rule mentioned above, FL is used to evaluate them. As the evaluating processes are similar to the one in evaluating "Inventory carrying facility", we do not illustrate them in detail here. These inference models have been realized by Fuzzy Logic Toolbox in MATLAB, and the corresponding program is shown in A-2, A-3, and A-4 of Appendix A. The corresponding numerical examples for evaluation results are shown in Table 5-4.

(3) Evaluation of "Communication system"

"Communication system" (*commSystem*) is determined by two variables: "Communication cost" (*commCost*) and "Communication method" (*commMethod*). *CommCost* is a quantitative variable, but *commMethod* is a logic variable, so ABL is used to finish the evaluation process. As mentioned in subsection 5.4.4.2, this evaluation is carried out according to following procedure.

Define the global domain. In this subsystem, there are three variables: *commMethod*, *commCost*, and *commSystem*. For *commMethod*, there are 7 possible values which are listed in Table 5-5. For *commCost*, three values are used to describe it: *'high'*, *'fair'*, and *'low'*. For the output variable *commSystem*, the evaluation result is divided into five levels, and indexed by integer *1* to *5*. The global domain of this subsystem is shown as Table 5-5.

Table 5-5 The global domain of subsystem "Communication system"

CommCost	*{'high', 'fair', 'low'}*
CommMethod	*{'telephone', 'fax', 'Internet', 'telephon_fax', 'telephon_Internet', 'fax_Internet', 'telephon_fax_Internet', 'telepnon_fax_Internet_Intranet'}*
commSystem	*{'1', '2', '3', '4', '5'}*

Evaluation of Possible Distributors

In SABL, the domain of a variable is defined by function *element*. For example, the domain of *CommCost* is defined by:

communicationCost=element('n', {'high', 'fair', 'low'}, {}, 'Communication Cost');

Model the system. Modeling the system means defining premises and connecting them by appropriate logic operators. A sample premise is shown as:

commCost is *'fair'* AND *commMethod* is (*'telephon_Internet'* OR *'telephon_fax_Internet'*) IMPLIES *commSystem* is *'4'*.

In SABL, this premise is expressed by following statements:

```
x12 = assign(communicationCost, {'fair'});
x22 = assign(communicationMethod, {'telephon_Internet', 'telephon_fax_Internet'});
y4 = assign(commSystem, {'4'});
y=conjunct(x12, x22);
Premise_4=bimp(y, y4);
```

The full premise set is shown in A-5 of Appendix A. Same as in FL inference process, such premises can reflect the decision maker's opinion, so ABL can also integrate the decision maker's role into the design process.

Interact with environment. This interaction process can be expressed as: forming the input vector, presenting this input vector to the system modeled above, and then computing the output vector. This output vector is the evaluation result.

In SABL, this process is accomplished by function *state*, e.g.:

output_SV = state(input_SV, System);

Here, *input_SV* is the input vector, *System* is the inference system formed by connected premises, and *output_SV* is the output vector.

This inference model has been realized by SABL in MATLAB, and the corresponding program is shown in A-5 of Appendix A. A numerical example for the evaluation result is shown in Table 5-4.

(4) Evaluation of "Hardware".

After evaluating its five descendent branches, it is time to evaluate "Hardware" itself. According to the method selection rules mentioned in subsection 5.4.5, FL is used to evaluate it. The evaluation process is same as the one in evaluating "Inventory carrying facility" except following two points:

- As FL is used here, the evaluation model is similar to Figure 5-20. The only difference is that, in this model, there are five rather than three input variables.
- As ABL is used to evaluate "Communication system", one input for this model, *commSystem*, is a multi-value logic variable, and it is expressed by an integer ranging from *1* to *5*. Now, this input variable needs to be fuzzified. As mentioned in subsection 5.4.5, we can view it as a quantitative variable, and fuzzify it directly. The Fuzzification result is shown in Figure 5-22.

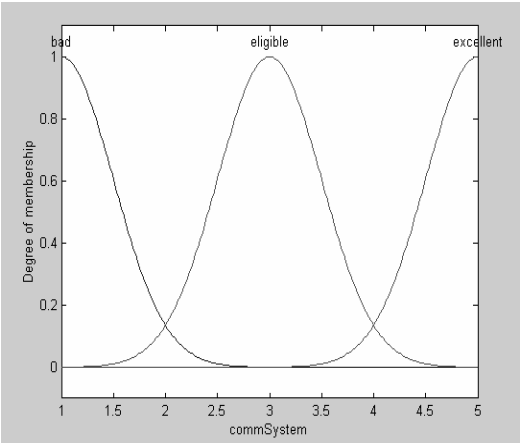


Figure 5-22 Fuzzification of variable *commSystem*

After fuzzifying *commSystem*, all the remained evaluating process is same as the one in evaluating "Inventory carrying facility". This inference model has been realized by Fuzzy Logic Toolbox in MATLAB, and the corresponding program is shown in A-6 of Appendix A. A numerical example for the evaluation result is shown in Table 5-4.

Same as the evaluation process for "Hardware", the factor "Software" of the firm can be evaluated. Based on the evaluation results for "Hardware" and "Software", the factor "Internal factor" of the firm is evaluated. At last, the final evaluation result for the firm, "Evaluation of a distributor", is obtained by the evaluation results for factors "Internal factor" and "External factor".

Based on this evaluating procedure, we evaluated a distributor, and the result is shown in Figure 5-3. The input values for this evaluation module are shown at the rightmost tier of Figure 5-3, and indicated by "Input" in each block. In the blocks at other tiers, the method (FL or ABL) used for evaluating the corresponding subsystems is indicated by keyword "Method", and the evaluation result for this subsystem is indicated by keyword "Result". By the evaluation result, this distributor can be compared with other possible ones, and the selection/rejection decision may be made.

5.5 Summary

In this chapter, a distributor evaluation module is developed according to following steps:

- The factors to be considered when evaluating a distributor are determined, and listed in hierarchical form by AHP. By systematic analysis, this factor set is relatively complete.
- A mobile agent based information acquisition system is developed to collect data from possible distributors. As the information is acquired via Internet, this approach is economical and efficient. If, due to some reasons, the mobile agent

approach is not applicable for some distributors, or the information acquired by mobile agent is not complete, other methods such as telephone, fax, E-mail can be used to collect data from these distributors. Actually, for this mobile agent system, to collect data is just its initial mission. In the future, it may act as an information infrastructure for the integration of entire distribution chain. From this point of view, this research is still valuable even if some of its functions can not be applied in reality.

- FL and ABL are integrated to evaluate a distributor quantitatively. This method can deal with quantitative, qualitative and logic input variables, and the evaluation does reflect the decision makers' preference.

After evaluating all potential distributors, some eligible ones are selected. Based on the selected retailers, some locations are identified as possible places to build wholesalers. All these selected retailers and possible locations for building wholesalers act as candidates to design a distribution chain. These candidates will be input into the design module to be illustrated in next chapter.

CHAPTER 6 DESIGN OF DISTRIBUTION CHAIN

According to Figure 3-2, after evaluating all possible distributors and selecting the eligible ones, we come to the second module of system design phase: design of distribution chain. This module can be viewed as the core of this integrated design methodology. In this chapter, a set of models, formulae and algorithms will be provided to design a distribution chain.

6.1 Introduction

As mentioned in Chapter 3, in existing design methodologies, formulae *(3-1)* and *(3-2)* were widely used as general form to design a distribution chain (or supply chain). In these formulae, two types of parameters were determined inaccurately: customer demands at retailers and operation related parameters. Detail is given below.

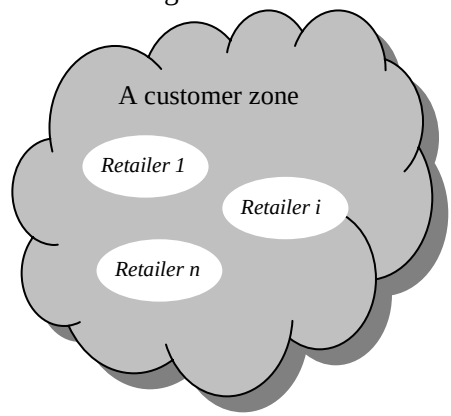


Figure 6-1 A customer zone with several retailers that sell the same product

First, let's talk about customer demands at retailers. As shown in Figure 6-1, in a customer zone, possibly, there are several retailers that sell the same product as the one to be distributed by this distribution chain. Normally, we only select one of them as our retailer in this zone. Assume that *Retailer 1* is the candidate selected by the evaluation module mentioned in previous chapter. By the acquired information, we can only approximately estimate the total demand for this customer zone. To calculate the exact demand at *Retailer 1*, we must know its market share for this product in its customer zone. Especially, this market share may be different for different marketing variables such as price, customer service, etc. In designing a distribution chain, customer demands are important parameters. They are related with revenue, product delivery cost, etc. Unfortunately, in existing design methodologies, market share was not estimated, and customer demands were normally taken as constant with respect to marketing variables. That is why we say that the customer demands were not determined accurately in existing methodologies.

Operation related parameters are another type of parameters which were not determined accurately in existing design methodologies. As mentioned in chapter 3, product delivery cost and inventory carrying cost are main costs considered in designing a distribution chain. To calculate product delivery cost, two parameters are used: delivery cost per unit product and volume of product to be delivered. To calculate inventory carrying cost, we need to know the average inventory level and unit holding cost. In reality, delivery cost per unit product and average inventory level are operation related parameters, i.e. different operation modes may result in different value for them. For example, when different routes are selected to deliver product, the

delivery cost per unit product may be different; average inventory level may also be different if taking different inventory control parameters. In existing design methodologies, to simplify the design process, these parameters were taken as constant regardless different operation modes, and they were determined by experience. Obviously, this simplification may cause remarkable error in designing a distribution chain.

To determine these two types of parameters more precisely, we employ the flow chart shown in Figure 6-2 to guide the design process of a distribution chain. According to Figure 3-2, all contents in this chapter belong to the module of distribution chain design (the second module of system design phase). This module can be divided into two sub-modules: market share estimation and design.

In sub-module of market share estimation, first, marketing variables are determined, then the retailer's market share for the product to be distributed is estimated. Based on this estimated market share, the customer demand at this retailer can be calculated. The main purpose to develop this sub-module is to determine customer demands more precisely.

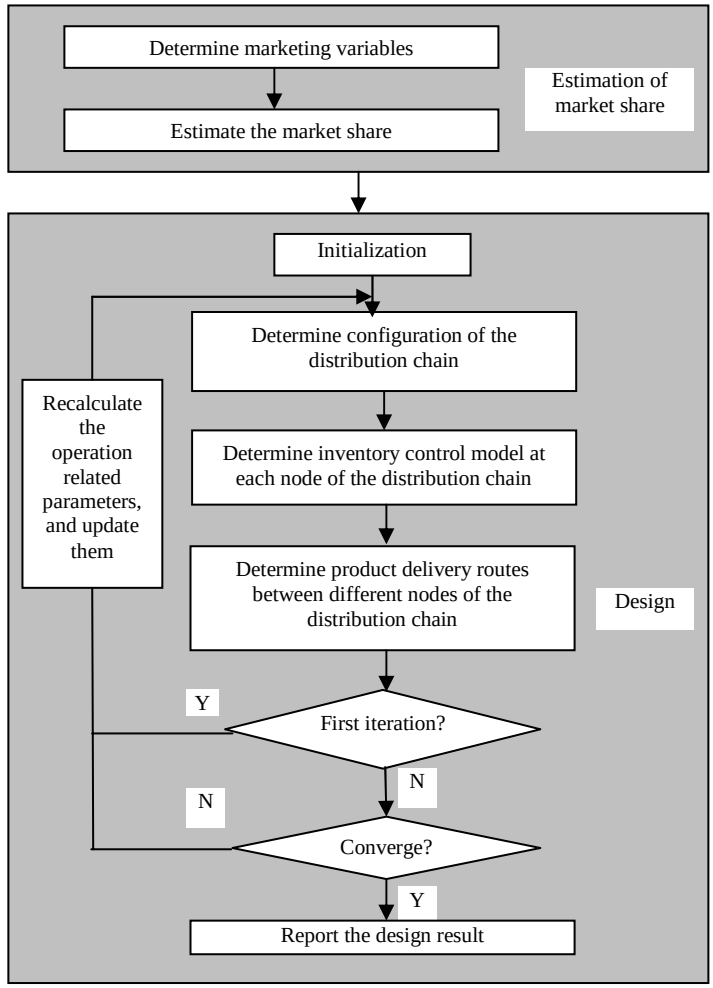


Figure 6-2 Flow chart for the module of distribution chain design

Now, let's talk about operation related parameters. These parameters will be used in the model of determining configuration of a distribution chain (including numbers and locations of wholesalers and retailers, and assignments of retailers to wholesalers), but they can only be calculated after the inventory model and product delivery routes are determined, and these two models depend on the configuration of distribution chain. This means that, these parameters need to be used in determining configuration, but

they can only be calculated after configuration is determined. To solve this conflict and determine operation related parameters more precisely, in sub-module of design, an iterative process is used to design a distribution chain. First, in the "Initialization" block, all parameters (including operation related parameters) are specified by experience. Then, based on these initial values, the configuration of distribution chain is determined. Given this distribution chain configuration, the inventory model at each node of the distribution chain is optimized, and routes for delivering product between different nodes are identified. After determining inventory model at each node and product delivery routes between different nodes, all operation related parameters are re-calculated and updated. Based on these newly updated parameters, configuration of the distribution chain is re-determined, and inventory models and product delivery routes are re-optimized. This iterative process proceeds until the design process converges, i.e. there is no significant difference between successive configurations of the distribution chain. The main purpose for applying this iterative design process is to determine the operation related parameters more precisely.

The flow chart shown in Figure 6-2 illustrates the sequence on how to apply models developed in this chapter in designing a distribution chain. Based on this flow chart, this chapter is organized as follows. In section 6.2, an ANN model is developed to estimate a retailer's market share for the product to be distributed in its customer zone. This model can reflect the relationship between the firm's distribution strategy and its market response, and the estimated market share can be used to calculate the customer demand at this retailer. Based on the calculated customer demands, in section 6.3, configuration of the distribution chain is determined by a MIP model. Given this distribution chain configuration, the inventory model at each node is determined in section 6.4, and routes for delivering product between different nodes are identified in section 6.5. To realize automatic design of distribution chain, all models, formulae and algorithms used in this chapter have been implemented by computer applications.

6.2 Market Share Estimation

Given a competitive market environment, a firm's distribution strategy may be indicated by marketing mix variables such as price, customer service level, etc. The market response to these variables can be expressed by market share [Lau et al., 1997]. In this section, after a review and analysis of existing models for market share estimation, ANN is selected to estimate a firm's market share in its customer zone.

Note that, the market share talked here is different from the one talked in section 5.2. In section 5.2, we took a retailer's general market share as an argument to evaluate its marketing environment. That market share can be viewed as the reflection of a retailer's competitive power. In this section, as we will calculate the exact customer demand at this retailer, so we only care about the retailer's market share for the product to be distributed, rather than its general market share. As the product is to be distributed, we can only forecast its market share based on the marketing variables.

6.2.1 Review and analysis of existing models for estimating market share

Improving market share is one of main concerns for marketers, so the estimation of market share has been attracting the attention of researchers and practitioners for decades. Next, after analyzing the existing models for market share estimation, our new estimation model is put forward.

The simplest model in estimating market share may be naive or linear model [Huff, 1964]. A naive model can be shown as follows:

$$s_{it} = \alpha_i + \beta_i s_{it-1} + \varepsilon_{it} \qquad (6-1)$$

And a linear model can be:

$$s_{it} = \alpha_i + \sum_{k=1}^{K} \beta_{ki} X_{kit} + \varepsilon_{it} \qquad (6-2)$$

Where, s_{it}: market share of brand i in period t.
α_i: brand i's constant component.
β_{ki}: brand i's effect of kth marketing instrument.
X_{kit}: the predictor of variable k for brand i in period t.
ε_{it}: the stochastic error of brand i in period t.

Both naive and linear models take linear function to estimate market share. Such kind of approximation may make the model simple, but the estimated result is inaccurate, because the relationship between market share and marketing mix variables is unnecessarily to be linear. To overcome this problem, other models, such as MCI (Multiplicative Competitive Interaction) and MNL (Multinomial Logit) models are developed. The general form of a MCI model may be depicted as [Jain et al., 1979]:

$$P_{ik} = \frac{\prod_{e=1}^{E} A_{ike}^{\beta_e}}{\sum_{i=1}^{n} \prod_{e=1}^{E} A_{ike}^{\beta_e}} \qquad (6-3)$$

Where, P_{ik}: Probability of customers (market share) in market zone k shopping at store i,
e: identifier of retailer store attributes, $e=1, \ldots, E$.
A_{ike}: variable describing the eth attribute of store i serving zone k.
β_e: attraction parameter associated with attribute e.

This MCI model is calibrated by following procedure: first collect data on historical buying patterns of randomly selected customers at each facility, and then calibrate the model's parameters by least square or other methods. The vital drawback for MCI

model is that, models built for one geographic region may not hold in others, and it is almost impossible to build an individual MCI model for each potential facility location.

An MNL model may be shown as [Lau et al., 1997]:

$$P_{ijk} = \frac{e^{\partial_{ik} + \sum_{s=1}^{S} \beta_{sjk} X_{sjk}}}{1 + e^{\partial_{jk} + \sum_{s=1}^{S} \beta_{sjk} X_{sjk}}} \qquad (6-4)$$

Where, P_{ijk}: estimated market share in market zone k for product family j when served from facility i.

s: an index representing the performance level on a particular attribute (such as price or a customer service element) that is offered from facility i to customers of product family j in market zone k.

β_{sjk}: beta coefficient describing the customer zone k's attractiveness to attribute s when purchasing product family j from facility i.

X_{sjk}: binary variable describing presence or absence of attribute s when product j is supplied to market zone k from facility i.

α_{jk}: logit model intercept for purchases of product family j from market zone k at facility i.

The calibration process for MNL is similar to that of MCI. As there is a binary variable X_{sjk} in MNL to indicate the presence or absence of an attribute, MNL model is flexible to model the different condition for different geographic regions, so the resulted MNL model may be a general model for different customer zones. At the same time, the market share estimated by MNL model is bounded asymptotically by zero and one, this insures that the market share estimates will take on feasible values, and never be negative or exceed 100%.

Based on MNL model, Lau et al. [1997] developed a SMNL (Switching Multinomial Logit) model, which can be viewed as a piecewise linear approximation to MNL model. By SMNL, the points where consumer response to price (or other variables) changes is nonlinear are identified, and these points are crucial in making decisions.

For all these estimation models, there is a common feature: the function form is pre-defined, and only the parameters in the pre-defined functions are determined during the calibration process. For example, linear function is pre-defined in naive and linear models, and exponential function is pre-defined in MCI and MNL models, etc. Actually, the relationship between market share and marketing mix variables is very complicated, and we do not know which function form is appropriate to reflect the case. If this relationship is approximated by one function form, the approximation may bring remarkable error in the estimation result. To estimate market share more precisely, a new method needs to be developed.

As mentioned in [Tsoukalas, 1997], ANN is a good pattern recognizer, and it learns by examples. The main reason that we can not precisely describe the relationship between market share and marketing mix variables is that, we do not know its pattern. But, by the past marketing experience, we have enough historical examples to describe this relationship. Obviously, ANN is the appropriate technology to work here: it can recognize the pattern of the relationship between market share and marketing mix variables by learning these examples, and then estimate market share by this recognized pattern. Based on this analysis, we will use ANN to establish a model for market share estimation. Next, after a simple introduction to ANN, we will use two steps to establish the model for market share estimation: identifying marketing mix variables and determining the ANN model.

6.2.2 Simple introduction to ANN

Since first put forth in [Mcculloch et al., 1943], ANN has largely been applied in production scheduling and process control, system identification, inspection and forecasting, etc. An ANN model can be defined as a data processing system consisting of a large number of simple, highly interconnected processing elements (artificial neurons) in an architecture inspired by the structure of cerebral cortex of the brain [Tsoukalas, et al., 1997]. Some basic concepts and formulae in ANN are introduced next.

Artificial neuron. Figure 6-3 shows a schematic representation of an artificial neuron, in which there are two functions: sum of weighted inputs and activation function. The first function simply aggregates the weighted inputs and yields a quantity I_j:

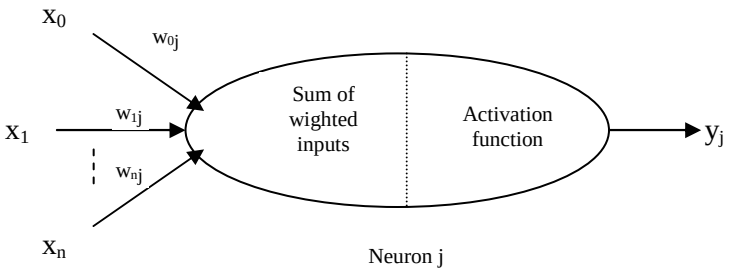


Figure 6-3 Schematic representation of an artificial neuron

$$I_j = \sum_{i=1}^{n} w_{ij} x_i \qquad (6-5)$$

The second part is a filter, usually called as the activation function, through which the combined signal flows. The normal formulae used as activation function are Linear function, Threshold function, Signum function and Logistic function, shown as below:

$$\phi(I) = \alpha I \qquad \text{Linear function}$$

$$\phi(I) = \begin{cases} 1 & I > T \\ 0 & I \le T \end{cases} \qquad \text{Threshold function}$$

$$\phi(I) = \begin{cases} +1 & I > T \\ -1 & I \le T \end{cases} \qquad \text{Signum function}$$

$$\phi(I) = \frac{1}{1+e^{-\alpha I}} \qquad \text{Logistic function}$$

Linear function makes the neuron reflect the sum of weighted inputs linearly, and be able to output any value. For the Threshold function, the information is passed only when the output *I* of the first part of artificial neuron exceeds threshold *T*, and output is restricted among *0* and *1*. Signum function passes negative information when this output is less than threshold *T*, and positive information when it is greater than *T*, this function restrict output value between *-1* and *+1*. Contrast to these two functions, Logistic function is a continuous one that varies gradually between two asymptotic values, typically *0* and *+1*. Parameter α is used to control the varying speed, or shape of the curve. These non-linear transfer functions give an ANN capability to learn or simulate both linear and non-linear relationships between inputs and outputs.

Neural network. As defined above, an ANN is an interconnected network of such neurons introduced previously. A typical neural network is fully connected, i.e. there is a connection between each of the neurons in any given layer with each of neurons in the next layer. When there are no lateral connections between neurons in a given layer, and none back to previous layers, the network is said to be feedforward. Figure 6-4 shows a fully connected and feedforward neural network. For each connection, there is a weight w_{ij} (as shown in Figure 6-3) assigned to it.

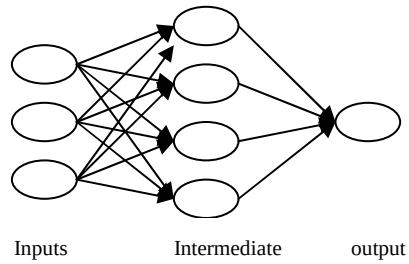


Figure 6-4 A fully connected, feedforward neural network

Training. Training is the process of adapting the connection weights in an artificial neural network to produce the desired output vector in response to a stimulus presented to the input buffer [Tsoukalas et al., 1997]. A supervised training process can be depicted as Figure 6-5. There are mainly four types of supervised training algorithms [Patterson, 1996]:

- Hebbian training. In this algorithm, a connection weight is incremented if both the input and desired output are large.
- Competitive training. In competitive training, the weights are adjusted to favor neurons that initially respond most strongly to given input stimuli. Neurons in a given layer compete to represent an input pattern, but only a single unit wins.

- Stochastic training. It adjusts weights in a probabilistic manner. Stochastic training is also called as Boltzmann training.
- Error correction training. It takes place when there is error (i.e. there is difference between desired output and the actual output), and its goal is to minimize this error.

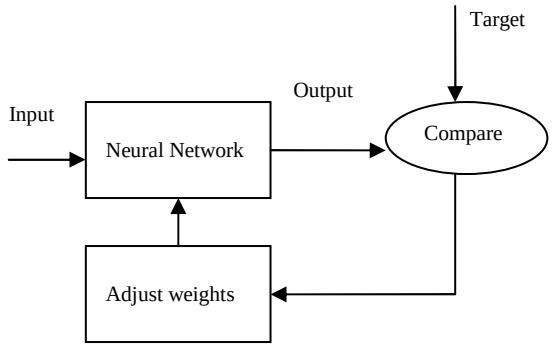


Figure 6-5 Supervised training process

By far, the most common method among error correction training algorithm is referred as back propagation, which is a 2-phase, gradient descent optimization algorithm. Initially, weights are randomly assigned to each connection, and then back propagation method is used to minimize the sum of squared errors between the actual outputs and those estimated by the network. During the optimization process, weights are adjusted iteratively until the sum of squared errors is small enough. Once trained, the ANN can be used to estimate outputs by inputs, which is referred as recall process (for a full description of ANN, see [Tsoukalas et al., 1997]).

6.2.3 Identifying marketing mix variables

After introducing the method for estimating market share, it is time to identify main marketing mix variables that determine the firm's market share for the product to be distributed. As a firm runs in a competitive environment, the marketing variables come from two sources: the firm itself (internal factors) and its marketing environment (external factors). For the firm itself, the main factors that have influence on market share include product price, customer service, product feature, and advertise expenditure. For its marketing environment, such factors include number of competitors, the competitive power of all competitors, and average product price for them. Note that, in section 5.2, we talked about internal and external factors, but they are different from the ones talked here. In section 5.2, as we wanted to evaluate a firm comprehensively, all internal and external factors related to the design objectives were considered there. But here, as we want to estimate market share for a product, so we only consider those internal and external factors which are related with market share.

Internal factors. Such factors have determinative function on the firm's market share, and they may be decided by the decision makers. Such factors include:

(1) Product price. Price for the selling product has crucial effect on the firm's market share for this product. Normally, high price yields low market share, and vice versa. For the firm, its ultimate goal is to make profit, and high price is a possible way to make more profit. But high price may result in low market share, and finally reduce profit for the firm, so decision makers need to make compromise between the positive and negative effects of price.

(2) Customer service. Besides low price, improving customer service is another way to increase market share. But, higher customer service level means more cost for the firm, so, same as price, the decision maker also needs to make compromise and keep an appropriate customer service level for customers. Normally, a firm uses following variables to indicate customer service level:

- *Fill rate.* Fill rate is defined as the fraction of demand satisfied from the stock on hand. High fill rate can earn good reputation for the firm, and then result in high market share. On the other hand, as customer demand is a random variable, high fill rate may mean more safety stock, high inventory level and more inventory maintenance cost for the firm, this is the negative effect of high fill rate.
- *Flexibility.* Flexibility of a firm is defined as its capability to satisfy dynamic customer requirements by handling environmental uncertainty with profitability [Slolvang, 2001]. It can be indicated by the percentage of slack throughput for a warehouse. Similar to fill rate, high flexibility can also earn goodwill, and then bring high market share for the firm. But higher flexibility means more production and transportation capacity, and then more opening cost for a facility. Unnecessary high flexibility may result in waste of source.
- *Technical support.* For some products, the firm needs to provide after selling technical service for customers. Obviously, providing nice technical support may increase market share for the firm, but cause extra cost.

(3) Product features. They include quality, package, and other features of the product. These features, especially the quality of a product, remarkably influence the firm's market share for this product. As they are mainly determined during the production, rather than distribution process, we will not explain them here. In the ANN model mentioned next, a score ranging from *0* to *100* is used to indicate the product features.

(4) Advertising expenditure. Excellent advertisement is another way to increase market share, but advertising expenditure adds cost for the firm.

External factors. We view all competitors as an entity. The factors that have influence on the firm's market share include:

(1) Number of competitors in this customer zone. Normally, the more competitors are there in a customer zone, the stronger the competition will be, and the resulted market share for the firm may be lower, so this is a negative factor for market share.

(2) Competition power for all competitors. We use an enterprise's global market share (i.e. its market share all over the world) to indicate its competition power. So, the competition power for all competitors is indicated by the sum of their global market shares. Obviously, this is a negative factor for the firm's market share.

(3) Average product price of all competitors. We use the following formula to calculate this variable:

$$P_a = \frac{\sum_{i=1}^{n} P_i S_i}{\sum_{i=1}^{n} S_i} \qquad (6-6)$$

Where, P_a: Average product price of all competitors.

P_i: product price of competitor i, $i = 1, \ldots, n$.

S_i: local market share (a competitor's market share in this customer zone) for competitor i.

This variable is the counterpart of product price for the firm. Its effect on the firm's market share is also negative.

All the variables identified above and their effect on the firm's market share is summarized as Table 6-1.

Table 6-1 Marketing mix variables and their effect on the firm's market share

Variable		Effect
Product price for the firm ($)		Negative
Customer service	Fill rate (%)	Positive
	Flexibility (%)	Positive
	Technical support (%)	Positive
Product features		positive
Advertising expenditure ($)		positive
Number of competitors in the customer zone		negative
Competitive power for all competitors (%)		negative
Average product price for all competitors ($)		negative

6.2.4 Determining the ANN model

There are two issues to be addressed when determining an ANN model: its structure and parameters in training process. A multi-layer fully connected feedforward neural network is applied in this market share estimation model. For its structure, we need to determine the number of layers, and number of nodes in each layer. For training process parameters, we need to determine learning rate and momentum. Next, the rules on how to determine these parameters are illustrated individually.

Structure of the ANN model. The ability of neural network to model input-output patterns is directly related to its structure. Patterson [1996] suggested that: to model interaction and non-linearity, the typical network consists of a single intermediate layer between input and output layers. When discontinuities are present in the data, two intermediate layers are desirable. For our model, a single intermediate layer neural network is selected, i.e. there will be three layers in this ANN model: input layer, one intermediate layer and output layer.

For the input layer, as nine marketing mix variables are identified above, there will be nine nodes in this layer. For the output layer, only one node is needed as there is only one output variable. For intermediate layer, too many nodes may make the model over-fit the training data; on the other hand, too little nodes may limit its modeling ability. Hornik et al. [1989] suggested that the number of nodes in intermediate layer ranges from $(2N+P)^{1.2}$ to $(2N+1)$, where N is the number of nodes for input layer, and P is number of nodes for output layer. We determine the number of intermediate nodes as *25*.

For a function approximation (or regression) ANN, the typical activation function for neurons in intermediate layer is Logistic function, and for output layer, linear function is the appropriate one. To speed the training process, a bias is added to each neuron. This bias can produce an effect equivalent to offsetting the origin of activation function, and then may result in more rapid training.

Parameters in training process. Back propagation is taken as the training method for this ANN model. The training process can be shown by the following formula:

$$\Delta w_{pq.k}(N+1) = -\eta_{pq}\delta_{pq.k}\phi_{p.j} \qquad (6-7)$$

Where, $\Delta w_{pq.k}(N+1)$: increment of the weight between node p (in jth layer) and node q (in kth layer) in $(N+1)$th iteration.

η_{pq}: training (or learning) rate for weight between node p and q.

$\delta_{pq.k}$: is defined as $2\alpha(T_q-\phi_{q.k})\ \phi_{q.k}(1-\phi_{q.k})$, where α is the constant in Logistic function, T_q is the target output for node q (in kth layer), $\phi_{q.k}$ is the output for node q.

$\phi_{p.j}$: the output for node p in jth layer.

For this training process, the parameter to be determined is training rate η (e.g. η_{pq}). According to [Tsoukalas 1997], η must range between *0* and *1*. If η is large (e.g. *0.8* or thereabouts), the weight vector will take relatively large step and find the minimum faster. However, if the input data patterns are not highly compacted around the "ideal" example, this will cause the network to jump wildly each time a new input pattern is presented. If η is small (*0.2* or thereabouts), the learning step will not be so wild, but the network may require longer time to learn the patterns, and the resulted network may be over-fitted. As a compromise, the learning rate for our model is specified as *0.4*.

To reduce the training time, momentum is introduced into the training process. As stated in [Tsoukalas, 1997], momentum is used to keep the training process going in the same general direction analogous to the way that momentum of a moving object behaves. By adding momentum into the training process, the formula shown above becomes:

$$\Delta w_{pq.k}(N+1) = -\eta_{pq}\delta_{pq.k}\phi_{p.j} + \mu\Delta w_{pq.k}(N) \qquad (6-8)$$

Where μ is the momentum coefficient, $\Delta w_{pq.k}$ is the previous (*N*th iteration) adjustment of weight between node *p* in *(k-1)*th layer and node *q* in *k*th layer. As suggested in [Tsoukalas 1997], momentum coefficient in our model is specified as *0.9*.

All parameters determined above can be summarized as Table 6-2. By these parameters, the ANN model is determined.

Table 6-2 Parameters in the ANN model for market share estimation.

Number of layers	3
Number of nodes in input layer	9
Number of nodes in intermediate layer	25
Number of nodes in output layer	1
Activation function for intermediate layer	Logistic function
Activation function for output layer	Linear function
Learning rate	0.4
Momentum coefficient	0.9

6.2.5 Realization of ANN model

An ANN model can be realized by Neural Network Toolbox in MATLAB. The procedure to realize the ANN model determined above is shown as Figure 6-6.

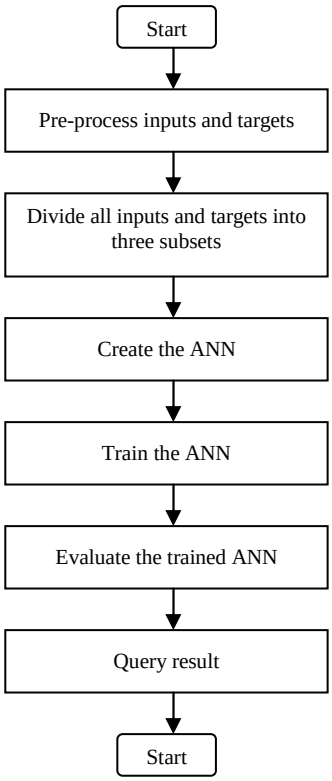


Figure 6-6 Procedure to realize an ANN model in MATLAB

(1) Pre-process inputs and targets. Before applied to estimate market share, the ANN model needs to be trained by a set of inputs and targets (desired outputs). Neural network is sensitive to absolute magnitudes of different variables. For example, in this model, price may range from 1000 to 5000, while fill rate only ranges from 0 to 1, the fluctuations in price will tend to swamp any change in fill rate. To minimize the influence of absolute scale, all inputs in this model are normalized to range from *0* to *1* by function *premnmx* provided by Neural Network Toolbox. The corresponding form is:

[pn, minp, maxp, tn, mint, maxt] = premnmx (p, t).

Where, *p* is input matrix, *t* is target matrix, *pn* and *tn* are normalized input and target matrices respectively.

(2) Divide all inputs and targets into three subsets: two fourths of data for training set, one fourth for validation set, and the rest for test set.

(3) Create the ANN. Function *newff* is used to create and initialize an ANN, the form is:

net=newff(minmax(ptr), [25,1], {'logsig', purelin'}, 'traingdm').

In this function, the first argument is the matrix of minimum and maximum values of input variables. The second argument is used to indicate the number of nodes in intermediate layer (here *25*) and output layer (here *1*). The third argument indicates the activation function for nodes of intermediate layer (here Logistic function, called as *'logsig'* in MATLAB) and output layer (here linear function, called as *'purelin'* in MATLAB. The fourth argument is the training function which will be explained next. Function *newff* automatically call function *init*, which is used to initialize the weights and biases by default functions.

(4) Train the ANN. During training process, weights and biases are iteratively adjusted to minimize the Mean Square Errors (*MSE*), which can be shown as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(t_i - a_i)^2 \qquad (6-9)$$

Where, t_i is the target (desired output) for node i in output layer, i=1, …, n. In this model, $n=1$.

a_i is the actual output for node i in output layer.

As mentioned above, back propagation algorithm is selected to train the neural network. The learning rate can be set by variable *net.trainParam.lr*, and momentum coefficient by *net.trainParam.mc*.

One of the problems that occur during training process is overfitting, i.e., the error on the training set is driven to a very small value, but it becomes very large when new data is presented to the network. The network has memorized the training examples, but it has not generalized to new situation. To improve the generalizing capability, an early stopping technique is used in the training process [Demuth et al., 2001]. In procedure 2, data have been divided into three sets: training set, validation set and test set. In early stopping technique, the training set is used for computing gradient and updating the weights and biases for the network. The error on the validation set is monitored during the training process. The validation error will normally decrease during the initial phase of training. However, when the network begins to overfit the data, error on validation set will begin to rise. When the validation error increases for a specified number of iterations, the training process is stopped. Such training process can be finished by function *train*

[trainedNet, tr]=train(net, ptn, ttn, [], [], v)

Where, *net* is the neural network to be trained, *trainedNet* is the trained network.

ptn and *ttn* are normalized inputs and target in training set.

v is a structure including inputs and targets in validation set.

The fourth and fifth arguments in function *train* are initial input and layer conditions, which are null here.

(5) Evaluate the trained ANN. Test set is used to analyze the performance of the trained network by following commands:

answer=sim(trainedNet, testInput)
[slope, intercept, correlation]=postreg(answer, testTarget)

In these routines, first, the inputs in test set, *testInput*, are presented at the trained network *trainedNet*, the corresponding result, *answer*, is output by function *sim*. Secondly, both the network output and targets (desired outputs) in test set are input into function *postreg*, which analyses the relationship between them. This function returns three parameters: *slope* and *intercept* correspond to the slope and y-intercept of the best linear regression relating targets and network outputs, *correlation* is the correlation coefficient between targets and network outputs. If both *slope* and *correlation* are close to *1*, and *intercept* is close to *0*, then the trained network is satisfied, otherwise the structure of the neural network needs to be adjusted.

(6) Query result. After trained and evaluated, the neural network *trainedNet* can be used to estimate the output (market share) by inputs (marketing mix variables), and it is finished by function *sim*

marketShare=sim(treainedNet, input)

Where *input* is the input presented at the trained neural network *trainedNet*, and *marketShare* is the resulted market share. This estimated market share will be used in the design model developed in next section.

In this ANN based market share estimation model, both internal and external factors for a retailer have been considered. These factors are universal for any retailer, so the model can be applied for all retailers, not only for a particular one. The model is trained by the historical data of the host enterprise, so it can also be applied for the retailers which have no experience of distributing the product of concern. If the host enterprise estimates market share for a new product, and it has no experience in distributing such product, then this model is not applicable.

Besides estimating a firm's market share in its customer zone, the ANN model developed above can also help decision makers to determine values for marketing mix variables. We take price as an example to illustrate the determining process. First, fix other marketing mix variables, and vary price in the possible range, and then present them onto the trained neural network to get the corresponding market share. Then, approximate the relationship between price and market share into a function: *marketShare = f(price)* by least square method. At last, the price can be determined by maximizing *price* $\times$ *f(price)*, which is roughly proportional to profit.

In practice, pricing is a complex process. When determining product price, the decision makers need to consider costs, sales, and the strategic distribution strategy, etc. But these factors are difficult to be expressed by mathematical models, so the process mentioned above can only be viewed as an assistant.

6.3 Determining the Configuration of a Distribution Chain

According to Figure 6-2, after estimating the possible retailers' market shares in their customer zones, it is ready to design a distribution chain. In this design sub-module, after initialization, the first step is to determine the configuration of a distribution chain. In this section, an optimization model will be developed to accomplish this work.

In chapter 5, all possible distributors were evaluated, and a set of eligible ones were selected according to pre-determined criteria. Based on the selected possible distributors, some locations were identified as possible places to build wholesalers. We call the selected possible distributors as retailer candidates (as the distributors will act as retailers), and possible places for building wholesalers as wholesaler candidates.

As mentioned in section 4.1, we will design a distribution chain with one distribution center, one tier of wholesalers and a set of retailers. The abstract structure of such a distribution chain can be shown as Figure 6-7. Based on this abstract structure, the task for determining the configuration of a distribution chain is two-folded:

- Determine the selection/rejection of retailer candidates and wholesaler candidates.
- Determine the assignments of retailers to wholesalers.

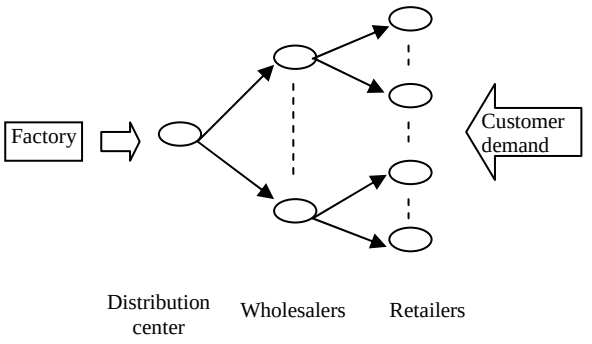


Figure 6-7 Abstract structure of the distribution chain to be designed

Before finding the method to accomplish these tasks, we need to analyze property of the problem we are facing. In mathematics, these tasks can be expressed by binary variables. For example, a binary variable can be used to indicate the selection/rejection of retailer and wholesaler candidates (e.g. *1* indicates selection, and *0* indicates rejection), or assignments of retailers to wholesalers (e.g. *1* if a retailer is assigned to one wholesaler, *0* otherwise). Assume that, we have n retailer candidates and m wholesaler candidates, then there will be $(m+n)$ binary variables to express the selection/rejection of retailer and wholesaler candidates, and $(m \times n)$ binary variables to indicate the assignments of retailers to wholesalers. We can imagine that, how large the problem will be if there are hundreds even thousands of retailer candidates and dozens of wholesaler candidates. As set in section 4-2, in this dissertation, the objective for

designing a distribution chain is to maximize its profit subject to satisfying customer requirements. Obviously, the problem we are facing is an optimization problem with large scale. As stated in [Kreyszig, 1999], MIP (Mixed Integer Programming) is an appropriate optimization method for large scale problems, and the software package to realize it is available in market. Based on this analysis, in this dissertation, MIP will be used to determine the configuration of a distribution chain.

6.3.1 Simple introduction to MIP

The general form of a MIP model may be depicted as follows [Kreyszig, 1999]:

Maximize (or Minimize): $f(x_1,\ldots,x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n$ (6−10)

Subject to

$$a_{11}x_1 + \ldots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + \ldots + a_{2n}x_n = b_2$$
$$\ldots\ldots\ldots$$
$$a_{m1}x_1 + \ldots + a_{mn}x_n = b_m$$
$$x_i \geq 0 \qquad (i = 1,\ldots,n)$$

Where $x_1 \ldots x_n$ are called control variables, $a_{11} \ldots a_{mn}$, $b_1 \ldots b_m$ and $c_1 \ldots c_m$ are parameters. Control variables may be integer or binary.

The most common method to solve a MIP model is Branch and Bound approach (see [Nemhauser et al., 1988] for detail).

6.3.2 MIP optimization model

As shown in formula (6-10), an MIP model is composed of two parts: objective function and constraints. Next, we will illustrate them separately.

6.3.2.1 Objective function

As determined in chapter 4, the objective for our design methodology is to maximize profit subject to satisfying customer service requirements. Customer service requirements are expressed as constraints, so, only maximizing profit is left in the objective function. In a distribution chain, profit can be expressed as sum of revenues minus sum of distribution costs.

Revenue. In a customer zone, the revenue for a retailer can be formulated as:

$$P_i S_i D_i$$

Where P_i: the product price at retailer *i*.

S_i: the market share of retailer *i* in its customer zone. This market share is estimated by the ANN model developed in previous section.

D_i: total customer demand in this customer zone during a planning period. In practice, demand is a random variable. As we are considering long term

demand in this model, we use its average value to represent this random variable. According to [Jayaraman et al., 2001], such simplification is acceptable.

The sum of revenues for all retailers in a distribution chain is expressed as:

$$\sum_{i=1}^{n} u_i P_i S_i D_i$$

Where u_i is the decision variable to indicate retailer i is selected/rejected. If retailer i is selected, $u_i=1$, otherwise $u_i=0$. n is the number of retailer candidates selected by the distributor evaluation module developed in chapter 5.

Cost. In a distribution chain, there are mainly two types of cost: inventory carrying cost at each node and product delivery cost between different nodes. As assumed before, retailers do not belong to the host enterprise. So their inventory carrying cost is not counted here. Next, each part of the cost is explained in detail.

(1) Product delivery cost from wholesalers to retailers. Then sum of all product delivery costs from wholesalers to retailers can be formulated by the following linear function:

$$\sum_{j=1}^{m}\sum_{i=1}^{n} w_{ij} C_{ji} D_i S_i$$

Where, w_{ij}: the decision variable to indicate whether retailer i is assigned to warehouse j. If retailer i is assigned to j, $w_{ij}=1$; otherwise $w_{ij}=0$.
m: number of possible places where wholesalers can be located.
C_{ji}: unit product delivery cost from warehouse j to retailer i.

C_{ji} is an operation related parameter. For example, when a vehicle serves several retailers, the delivery cost per unit product allocated to a retailer largely depends on which route to be taken.

(2) Inventory cost at wholesalers. Inventory cost at a wholesaler is composed of two parts: fixed cost for opening a warehouse and variable cost for carrying the inventory. For the opening cost, most existing design methodologies took it as constant with respect to inventory level ([Chen et al., 1997], [Jayaraman et al., 2001], etc.). In practice, this is not the case. Normally the relationship between the highest inventory level at a wholesaler and the cost to open it can be depicted as Figure 6-8. Especially, such effect may be different for different locations. In our design methodology, this effect is considered,

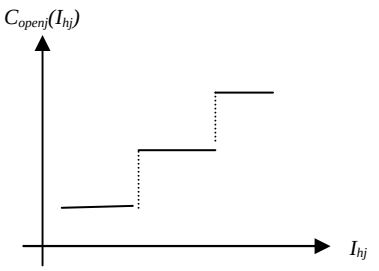


Figure 6-8 Wholesaler j's opening cost function with respect to its possible highest inventory level

and the opening cost for a wholesaler is determined according to its possible highest inventory level. The sum of inventory costs for all wholesalers is formulated as follows:

$$\sum_{j=1}^{m} v_j (C_{openj}(I_{hj}) + I_j C_j T)$$

Where v_j: the decision variable to indicate whether wholesaler j is selected. If selected, $v_j=1$, otherwise $v_j=0$.

$C_{openj}(I_{hj})$: function to indicate the opening cost for wholesaler j. I_{hj} is the possible highest inventory level at it.

I_j: the average inventory level for wholesaler j, it is an operation-related parameter.

C_j: inventory carrying cost per unit product per unit time for wholesaler j.

T: strategic planning period.

(3) Delivery cost from distribution center to wholesalers. This cost can be shown as:

$$\sum_{j=1}^{m} C_{0j} d_{wj}$$

Where C_{0j}: unit product delivery cost from distribution center to warehouse j. It is also an operation-related parameter.

d_{wj}: total demand at wholesaler j during a planning period. Here, d_{wj} can be represented by:

$$d_{wj} = \sum_{i=1}^{n} w_{ij} S_i D_i$$

(4) Inventory cost at distribution center. Same as the inventory cost at wholesalers, inventory cost at distribution centre is also composed of two parts: fixed cost for opening the distribution centre and variable cost for carrying the inventory. This opening cost is also related with the highest inventory level at distribution centre. The total inventory cost at distribution centre can be depicted as:

$$C_{open0}(I_{h0}) + C_0 I_0 T$$

Where $C_{open0}(I_{h0})$: function to indicate the opening cost for distribution centre. I_{h0} is the possible highest inventory level at it.

C_0: inventory carrying cost per unit product per unit time for distribution center *0*.

I_0: average inventory level at distribution center *0*, it is an operation-related parameter.

6.3.2.2 Constraints

Flexibility constraints. For a distribution chain, we mainly care about two types of flexibilities: production volume flexibility and delivery volume flexibility. Production volume flexibility is measured by the percentage of slack production capacity for an enterprise [Slack, 1987]. According to this definition, the production volume flexibility constraint is expressed as:

$$\frac{C_{p\max} - \sum_{i=1}^{n} u_i S_i D_i}{C_{p\max}} \geq \varepsilon_p \qquad (6-11)$$

Where C_{pmax}: the maximum production capacity for the host enterprise, so the numerator represents the slack production capacity.
ε_p: the required production volume flexibility.

At the same time, sum of demands at all retailers must exceed the minimal production capability to ensure enough resource utilization rate, i.e.

$$\sum_{i=1}^{n} u_i S_i D_i \geq C_{p\min} \qquad (6-12)$$

Where, C_{pmin} is the minimal production capability for the host enterprise.

Similar to production volume flexibility, delivery volume flexibility is defined as the percentage of slack throughput of a warehouse. So the delivery volume flexibility constraints for wholesalers are:

$$\frac{C_{t\max\ j} - \sum_{i=1}^{n} w_{ij} S_i D_i}{C_{t\max\ j}} \geq \varepsilon_{tj} \qquad \forall j \qquad (6-13)$$

Where C_{tmaxj}: the maximum throughput for wholesaler j,
ε_{tj}: the required delivery volume flexibility for wholesaler j, which is determined by the decision makers.

Same as production capability, to ensure resource utilization rate, the real throughput for a wholesaler must also exceed its minimal capability, i.e.:

$$\sum_{i=1}^{n} w_{ij} S_i D_i \geq C_{t\min\ j} \qquad \forall j \qquad (6-14)$$

Where, C_{tminj} is the minimal throughput for wholesaler j.

Normal constraints on variables. All variables must be non-negative, and u_i, v_j, w_{ij} are binary variables. For these binary variables, there are other constraints. First, a retailer can only be assigned to a warehouse, i.e.

$$\sum_{j=1}^{m} w_{ij} = u_i \qquad \forall i \qquad (6-15)$$

Secondly, only when both retailer i and warehouse j are selected, this retailer can be assigned to the wholesaler, i.e.

$$\sum_{i=1}^{n} w_{ij} = v_j \qquad (6-16)$$

After identifying the objective function and constraints, the MIP model can be formed as follows (the corresponding notation explanation is shown in Table 6-3):

Maximize

$$profit = \sum_{i=1}^{n} u_i P_i S_i D_i - \sum_{j=1}^{m}\sum_{i=1}^{n} w_{ij} C_{ji} S_i D_i - \sum_{j=1}^{m} v_j (C_{openj}(I_{hj}) + I_j C_j T) \qquad (6-17)$$
$$- \sum_{j=1}^{m} C_{0j} (\sum_{i=1}^{n} w_{ij} S_i D_i) - (C_{open0}(I_{h0}) + C_0 I_0 T)$$

Subject to

$$\frac{C_{p\max} - \sum_{i=1}^{n} u_i S_i D_i}{C_{p\max}} \geq \varepsilon_p \qquad (6-18)$$

$$\sum_{i=1}^{n} u_i S_i D_i \geq C_{p\min} \qquad (6-19)$$

$$\frac{C_{t\max\ j} - \sum_{i=1}^{n} w_{ij} S_i D_i}{C_{t\max\ j}} \geq \varepsilon_{tj} \qquad \forall j \qquad (6-20)$$

$$\sum_{i=1}^{n} w_{ij} S_i D_i \geq C_{t\min\ j} \qquad \forall j \qquad (6-21)$$

$$\sum_{j=1}^{m} w_{ij} = u_i \qquad (6-22)$$

$$\sum_{i=1}^{n} w_{ij} = v_j \qquad (6-23)$$

$$u_i, v_j, w_{ij} = 0 \quad or \quad 1 \qquad \forall i, j \qquad (6-24)$$

Table 6-3 Notation explanation for the MIP optimization model

Variables	Definition
index	
i	Retailer index
j	Wholesaler index
parameters	
n	Number of retailer candidates selected in chapter 5
m	Number of wholesaler candidates selected in chapter 5
P_i	Product price at retailer i
D_i	Total demand in retailer i's customer zone during a planning period
S_i	Market share of retailer i in its customer zone, which is estimated in previous section.
C_{ji}	Unit product delivery cost from wholesaler j to retailer i (operation related parameter)
I_{hj}	Possible highest inventory level at wholesaler j
$C_{openj}(I_{hj})$	Opening cost for wholesaler j. It is a function with respect to I_{hj}. The function may be determined by experience
I_j	Average inventory level for wholesaler j (operation related parameter)
C_j	Inventory carrying cost at wholesaler j per unit product per unit time
T	Planning period
C_{0j}	Unit product delivery cost from central depot to wholesaler j (operation related parameter).
I_{h0}	Possible highest inventory level at distribution centre *0*
$C_{open0}(I_{h0})$	Opening cost for distribution centre *0*. It is a function with respect to I_{h0}. The function may be determined by experience
C_0	Unit inventory carrying cost at distribution centre *0*
I_0	Average inventory level at central depot (operation related parameter)
C_{pmax}	Maximum production capacity for the host enterprise
C_{pmin}	Minimal production capacity for the host enterprise
ε_p	Required production volume flexibility for the host enterprise
C_{tmax}	Maximal throughput for wholesaler j
C_{tmin}	Minimal throughput for wholesaler j
ε_{tj}	Required delivery volume flexibility for wholesaler j
Decision variables (binary)	
u_i	1 if the retailer i is selected, 0 otherwise
v_j	1 if the wholesaler j is selected, 0 otherwise
w_{ij}	1 if retailer i is assigned to wholesaler j, 0 otherwise

For this MIP model, we give following further explanations:

- In this model, we only consider those retailer candidates and wholesaler candidates that were selected in Chapter 5. n is the number of retailer candidates, and m is the number of wholesaler candidates.
- When calculating revenue at a retailer (say retailer i), we use $S_i \times D_i$ to represent the customer demand at this retailer. Here, D_i is the total customer demand for this customer zone during a planning period, and S_i is the market share of this retailer in its customer zone. The market share is estimated by the model established in previous section.
- After solving the model, we get values for all decision variables, then the configuration of the distribution chain is determined. For example, when u_i $(i=1\sim n)$ are determined, all retailers are located. When v_j $(j=1\sim m)$ are determined, the decision on where to build wholesalers is made. When w_{ij} are determined, the connections between wholesaler and retailers are identified.

Some commercial packages such as LINDO & LINGO, etc. have been developed to solve a MIP model, so we can conveniently use these existing computer applications to solve our model.

6.4 Determining Inventory Control Model at Each Node of the Distribution Chain

Maintaining inventory is one of the main activities in managing a distribution chain. According to Figure 6-2, after determining its configuration, it is time to plan inventory control models for each node of a distribution chain. Note that, although retailers do not belong to the host enterprise, the parameters for their inventory control must be determined so as to found the transportation model from wholesalers to retailers.

According to [Tijms, 1994], there are mainly four types of inventory control models, and the basic forms are: periodic review *(R, S)* model and continuous review (*s, Q*) model. In *(R, S)* model, an order is placed every *R* unit time to raise the inventory level to *S*. In *(s, Q)* model, an order *Q* is placed when at hand inventory is less than or equal to *s*. Due to the random customer demand at retailers, *(R, S)* model may cause following problems: the order may come when it is unnecessary (the at hand inventory level is high enough to meet the demand), or may not come when it is necessary (the at hand inventory level is less than safety stock). Both situations may cause extra cost for the retailer. So, we select *(s, Q)* as our inventory control policy.

For *(s, Q)* model, following formula was normally used to determine parameter *s* [Silver, 1985, Johnson et al., 1996]:

$$s = L + n\sigma \qquad (6-25)$$

Where, *s* is the reorder point (or safety stock)
L is the total demand during lead time.
n is the safety factor, which is determined by subjective judgement.
σ is the standard deviation of demand during lead time.

Parameter *Q* was determined by minimizing inventory carrying cost which is shown as follows:

$$TC = \frac{TD}{Q} \times c_o + (\frac{Q}{2} + n\sigma) \times c_h \qquad (6-26)$$

Where, *TC* is the total inventory carrying cost.
TD is the total demand between successive replenishments.
Q is the ordering quantity for each replenishment.
c_o and c_h are ordering cost and unit holding cost respectively.

In these formulae, when determining *Q*, the demand was assumed implicitly to be linear with respect to time, and so the average inventory level at the retailer was expressed as: $n\sigma+Q/2$. In practice, such assumption may not hold, and it may cause error when determining the inventory model at a retailer. To determine the inventory control parameters more precisely, we developed a simulation-based estimation model shown as follows.

The working process at a node (a retailer, a wholesaler or distribution centre) can be described as: the node faces a random demand process. When the at hand inventory level is less than or equal to *s*, an order *Q* is placed. This order will come after a *lead time*. Our objective is to minimize the inventory carrying cost per unit time (e.g. per day).

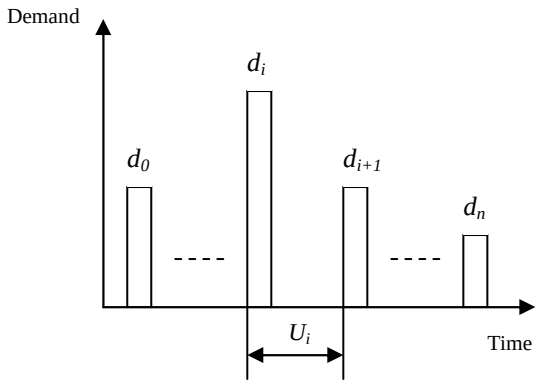


Figure 6-9 Demand process at a node

The random demand process at a node can be shown as Figure 6-9. Here, d_i (or d_{i+1}) represents the volume of an individual demand, U_i represents the inter-arrival time for successive demands. Both volume of individual demand and inter-arrival time for successive demands are assumed to be random variable with normal distribution. At retailers, the parameters (mean and standard deviation) of these two variables are estimated based on historical customer demand. At wholesalers and distribution centre, these parameters are obtained by simulation. After determining the inventory control parameters at retailers and assignments of retailers to wholesalers, the demand process at wholesalers can be simulated; after determining inventory control parameters at wholesalers, the demand process at distribution centre can also be simulated. Based on these simulation models, the parameters mentioned above can be obtained. We have realized these simulation models in MATLAB.

If we view retailers (or wholesalers) as customers for wholesalers (or distribution center), the working process at a wholesaler (or distribution center) is same as the one at a retailer. So we will take a retailer as an example to illustrate the inventory control model for all nodes in a distribution chain.

The inventory carrying cost at a retailer includes holding cost and shortage cost, and the cost function is expressed as:

$$\varphi(N_t, U_t) = \begin{cases} C_h N_t U_t & N_t \geq 0 \\ C_s |N_t| U_t & N_t < 0 \end{cases} \qquad (6-27)$$

Where, U_t is the inter-arrival time for successive demands.

N_t is net inventory. It equals to on hand inventory minus backlogged orders, Both U_t and N_t are random variables.

C_h is holding cost per unit product per unit time. C_s is shortage cost per unit product per unit time.

According to [Reuven, 1998], if a process can be split into several replicas, and for every replica, they have the same initiate state and independent, identical distribution,

then this process can be called Regenerative Process. The inventory carrying process at a node can approximately be viewed as a regenerative process. Every time when the absolute value of net inventory is less than a small number, a new regenerative begins. Assume that the inventory carrying process at a node is simulated within N regenerative cycles, then the inventory carrying cost per unit time can be estimated by following formula:

$$C(s,Q)=\frac{\sum_{i=1}^{N}\sum_{j=1}^{M_i}\left(\sum_{t=0}^{\tau_{ij}-1}\varphi((s+Q-\sum_{t}d_{ijt}),U_{ijt})+K\right)}{\sum_{i=1}^{N}\sum_{j=1}^{M_i}\sum_{t=0}^{\tau_{ij}-1}U_{ijt}} \qquad (6-28)$$

Where,

$C(s, Q)$ is the inventory carrying cost per unit time, which is a function with respect to s and Q.

N is the number of regenerative cycles.

M_i is order times during regenerative cycle i.

τ_{ij} is the number of demands in a replenishment cycle.

$\varphi((s+Q-\sum d_{ijt}), U_{ijt})$ is the cost function defined in formula (6-27). Its value is obtained by simulation. The first argument in this function represents the net inventory.

K is the fixed ordering cost.

Following procedure can be used to realize this simulation model and search the optimal parameter pair (s, Q):

- Generate random samples
  Inter-arrival time: $(u_{1,1}, \ldots\ldots u_{\tau 1,1}, \ldots\ldots, u_{1,N}, \ldots\ldots, u_{\tau N,N})$
  and demand :$(d_{1,1}, \ldots\ldots d_{\tau 1,1}, \ldots\ldots, d_{1,N}, \ldots\ldots, d_{\tau N,N})$ from corresponding PDFs (Probability Density Functions) based on N regenerative cycles.
- Simulate the working process at a retailer, and calculate the corresponding inventory carrying cost by function $\varphi((s+Q-\sum d_{ijt}), U_{ijt})$.
- Determine range for parameter s and Q by experience, and evenly take values from their corresponding ranges to form parameter pairs (s_k,Q_l).
- Estimate cost $C(s_k,Q_l)$ by formula (6-28), find the minimal cost. The parameter pair corresponding to this minimal cost is the optimal one (s^*,Q^*).

We have realized this simulation model in MATLAB, and the corresponding program is shown in B-1 of Appendix B. Next, an example is given to illustrate the utilization of this tool. The parameters for a retailer are shown in Table 6-4.

By the tool developed in MATLAB, the inventory carrying cost for different parameter pairs (s_k, Q_l) is estimated, as shown in Figure 6-10. By this searching result, following conclusions are reached: when *Q=100 (100units)* and *s=50 (100units)*, inventory carrying cost per unit time reaches minimum, and it is *108.67$/day*.

Table 6-4 Parameters for a retailer

Cost	Holding cost coefficient, C_h ($/100units*day)	1	Demand	Mean, μ_d (100units/day)	6.1
	Shortage cost coefficient, C_s ($/100units*day)	10		Variance, σ_d (100units/day)	1
	Reordering cost, C_{order} ($)	1000	Interarrival time	Mean, μ_t (day)	1
Lead time, L (day)		8		Variance, σ_t (day)	0.2
Search range for order up to level, S ($S=s+Q$) (100units)		140~180	Search range for reordering point, s (100units)		30~70

After getting optimal (*s, Q)* for a retailer, its average inventory level can be estimated by this simulation model. For example, the estimated average inventory level for this retailer is *111.2 (100units)*

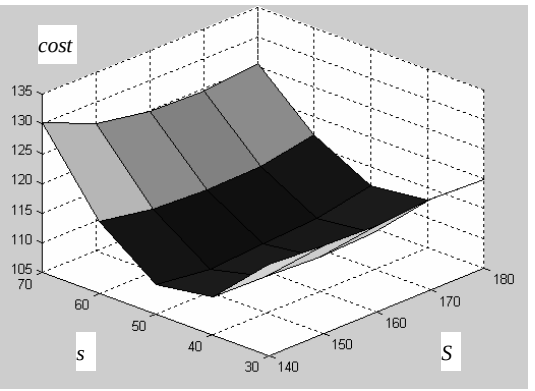


Figure 6-10 Inventory carrying cost for parameter pairs (s_k Q_l) ($Q=S-s$)

Although this model is developed for a retailer, it is applicable for all nodes in a distribution chain, including distribution center and wholesalers. By this model, the optimal parameters for inventory control at each node can be determined, and the average inventory level can be estimated. As mentioned in section 6.1, average inventory level at each node is an operation related parameter. In the iterative design process shown in Figure 6-2, this parameter will be updated when re-determining the configuration of the distribution chain.

6.5 Planning Product Delivery Routes in a Distribution Chain

Delivering product is another type of main activities in distribution chain management. First, let's consider the product delivery routes from wholesalers to retailers. After determining (*s, Q)* for each retailer, it is time to plan the transportation from wholesalers to retailers. Almost all of existing design methodologies consider a simple transportation mode, i.e. one vehicle serves a retailer [Sarbi, 2000, etc.]. In practice, the order quantity by a retailer is normally small, so one vehicle may serve several retailers in one journey. For such situation, following questions are raised: (1) how to cluster retailers; and (2) how to determine routes for vehicles? This is a typical vehicle routing problem.

Genetic algorithms have proven to be a versatile and effective approach for solving the vehicle routing and scheduling problem [Park, 2001]. For example, Gabbert et al. [1991] presented a genetic algorithm approach to learning low-cost routes and schedules for a large rail freight transportation network, Cheng et al. [1996] proposed a hybrid genetic algorithm to solve the fuzzy vehicle routing and scheduling problem, etc.

When planning product delivery routes in a distribution chain, the transportation cost and inventory carrying cost must be considered simultaneously. To reduce transportation cost, we hope that the sum of distances traveled by all vehicles is minimized. To reduce inventory carrying cost, we hope that all retailers are served at the right time (not too late, not too early). The key point is to make compromise between them, and minimize the sum of two types of cost. Unfortunately, in the existing genetic algorithm based routing approaches, normally the transportation aspect was emphasized, but inventory carrying aspect was ignored. For example, when they calculated cost in their objective function, the inventory carrying cost was not considered. Such ignorance may cause terrible mistake when planning product delivery routes in a distribution chain. To overcome this drawback, we will develop a genetic algorithm based model which deals with them simultaneously.

6.5.1 Simple introduction to genetic algorithm

Genetic algorithm was initiated and developed in early 1970s by John Holland, but its applications to practical problems were almost two decades in developing [Lefteri, 1997]. The primary purpose of using genetic algorithm is optimization. The specific nature of the problem to which optimization is applied will determine the type of genetic algorithm used, and especially fitness function. In what follows, some basic concepts in genetic algorithm are introduced.

Chromosome. It is defined as strings of artificial genetic systems which encode solutions for the problem, which are analogous to chromosomes in biological systems.

Mutation. It is the process by which a single component of a chromosome is changed randomly.

Crossover. The chromosomes of two parents are mixed by a process called "crossover", in which two new chromosomes are reproduced, each having some of the characteristics of the two parents. The genetic process evolves by mutation and crossover.

Fitness function. It is the function on which an optimization operates, i.e. seeking its maximum or minimum. In genetic algorithm, "fitness" is the quantity that determines the quality of a chromosome, from which a determination can be made as to whether it is better or worse than other chromosomes in the population.

6.5.2 Genetic algorithm model

A routing problem can be depicted as Figure 6-11. The wholesaler (denoted by *0*) that possesses a set of identical vehicles will serve a set of retailers (denoted by *1… p, …, q…n*). Retailer *i (i=1~n)* will order Q_i units product when its inventory

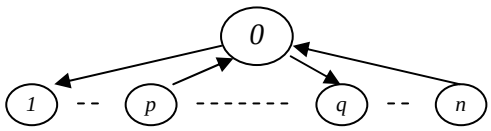


Figure 6-11 Problem illustration

level is less than or equal to s_i (Q_i and s_i were determined in the previous section). The task for us is to route vehicles to different retailers by minimizing the sum of transportation and inventory carrying cost.

Before optimizing the routes by genetic algorithm, its chromosome and fitness function must be specified, and mutation (or crossover) strategy to guide the optimization process must be determined.

6.5.2.1 Chromosome and fitness function

(1) Chromosome.

Similar to [Park, 2001], a chromosome is represented by two strings. The first string includes all retailers except the wholesaler. The second string has the same number of fields as the first one, and it denotes the vehicle numbers assigned to the retailers at the corresponding fields in the first string. Following example represents a routing solution which has 3 vehicles (denoted by *1~3*) and 15 retailers (denoted by *1~15*):

String 1: *2 4 6 8 9 3 12 15 14 11 13 7 1 5 10*
String 2: *1 2 1 3 3 2 1 2 1 3 2 1 1 3 2*

The example shown above represents three routes: *0-2-6-12-14-7-1-0* (served by vehicle *1*), *0-4-3-15-13-10-0* (served by vehicle *2*), and *0-8-9-11-5-0* (served by vehicle *3*), where *0* denotes the wholesaler.

(2) Fitness function

The intuitive objective in planning product delivery routes is minimizing cost, including transportation cost and inventory carrying cost. One problem in applying genetic algorithm is premature selection, i.e. the evolution process may converge to a local optimum. As stated in [Patrick, 1993], it can be as good to be different as it is to be fit, and largely scattered chromosomes can prevent the optimization process from premature selection (or local optimum). To avoid local optimum, we introduce diversity into the fitness function. Here diversity is used to indicate the scattering of chromosomes. Based on this analysis, the fitness function is calculated based on following two objectives: minimizing cost and maximizing diversity. Cost is composed of time related cost, distance related cost and vehicle-renting cost. Next, each part of the fitness function will be explained in detail.

Time related cost. It is related to inventory carrying cost. For a retailer, it hopes to be replenished when its net inventory is zero. If the vehicle arrives late when the retailer's net inventory is negative, the retailer will suffer from inventory shortage cost. If the vehicle arrives early when the retailer's inventory level is positive, the retailer will pay extra inventory carrying cost. As these two costs are related to the time when vehicle arrives at the retailer, we call them as time related cost. Time related cost is depicted as Figure 6-12. Here, N_i represents the net inventory at retailer i when vehicle arrives at it, C_s represents the unit inventory shortage cost, and C_h represents unit inventory holding

cost. If the net inventory is zero when vehicle arrives at the retailer, the time related cost will be zero; if the net inventory is negative, then the time related cost is $-C_sN_i$ (left part of Figure 6-12); if the net inventory is positive, then the time related cost is C_hN_i (right part of Figure 6-12).

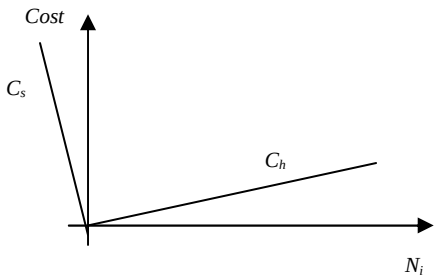


Figure 6-12. Time related cost

Before depicting how to estimate the time related cost, we introduce a concept: inter-service time for a retailer (say retailer *i*). The net inventory process for retailer *i* can be shown as Figure 6-13. At time point *a* and *b*, the retailer is replenished (served) by a vehicle. So, the period between these two successive services is defined as inter-service time for retailer *i*. In a routing solution, every retailer belongs to a route. Here we assume that, retailer *i* belongs to route *j*. As all retailers in a route are served by one vehicle, the inter-service times for all retailers in this route are same. We indicate this common inter-service time as T_j.

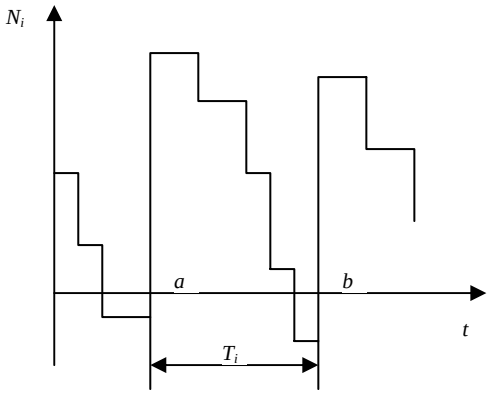


Figure 6-13 Explanation of inter-service time T_i

As the demand at a retailer is random, when vehicle arrives at the retailer, the retailer's net inventory is also random, and it can be calculated by:

$$N_i(T_j) = Q_i - T_j d_i \qquad (6-29)$$

Where, Q_i is the ordering quantity for retailer *i*
T_j is inter-service time for route *j*. Here, we assume retailer *i* belongs to route *j*.
d_i is the per unit time demand at retailer *i*.

Different from the previous section, to simplify the problem, here, we use demand rate (per unit time demand) to represent the demand process at a retailer.

We assume that the demand rate at retailer *i* is a random variable with normal distribution, and the corresponding PDF is:

$$p(d_i) = \frac{1}{\sigma_i\sqrt{2\pi}}\exp(-\frac{1}{2}(\frac{d_i-\mu_i}{\sigma_i})^2) \qquad (6-30)$$

Where, μ_i and σ_i are mean and standard deviation of demand d_i. Then the PDF of net inventory N_i, $p(N_i)$, can be calculated by formula (6-29). Based on $P(N_i)$, the mathematical expectation of time related cost for retailer i , $C_{ti}(T_j)$, can be estimated by following formula:

$$C_{ti}(T_j) = \int_{+\infty}^{0} C_h N_i p(N_i) dN_i + \int_{0}^{-\infty} -C_s N_i p(N_i) dN_i \qquad (6-31)$$

In this formula, the first item represents the extra inventory carrying cost, and the second item represents the inventory shortage cost. According to formula (6-29), time related cost for a retailer is a function with respect to its inter-service time T_j.

For a routing solution, its total time related cost C_t is expressed as:

$$C_t = \sum_i C_{ti}(T_j) \qquad (6-32)$$

Distance related cost. It is proportional to the sum of distances traveled by all vehicles in a routing solution, shown as follows:

$$C_d = c_d \sum_j B_j \qquad (6-33)$$

Where, B_j is the traveling distance for vehicle j.
c_d is unit traveling cost.
C_d is total distance related cost for a routing solution.

Vehicle renting cost. It is proportional to the number of vehicles used in a routing solution, shown as:

$$C_r = c_r n \qquad (6-34)$$

Where, n is the number of vehicles used in a routing solution, c_r is the unit renting cost.

Diversity. Here, diversity can be expressed as the difference between chromosomes. So the diversity for a chromosome is calculated by the sum of inverse squared distances between that chromosome and others in the population. The distance between chromosomes i and j is defined as:

$$d_{ij}^2 = \sum_{k=1}^{m} (V_{ik} - V_{jk})^2 \qquad (6-35)$$

Where, V_{ik} is the vehicle number for retailer k in chromosome i, $k=1\sim m$, m is the number of retailers in a chromosome.
V_{jk} is the vehicle number for retailer k in chromosome j.

So the diversity for a routing solution is:

$$D_d = \sum_i \sum_{j \in N \setminus i} \frac{1}{d_{ij}^2} \qquad (6-36)$$

Where, N is the collection of all chromosomes in the population. So maximizing diversity means minimizing D_d.

Summing up the four parts mentioned above, we get the fitness function:

$$f = w_c(C_t + C_d + C_r) + w_d D_d \qquad (6-37)$$

Where, w_c and w_d are weights for cost and diversity, $w_c+w_d=1$. Both are determined by subjective judgement. The objective for this optimization process is to minimize f.

6.5.2.2 Optimization process

Normally, genetic algorithm uses random mutation and crossover to find the optimal solution. Because of the limited capacity of vehicles and other routing constraints, such random process may cause infeasible solution in this situation. To avoid this problem, we only apply guided mutation in this optimization process, and its flow chart is shown as Figure 6-14. In what follows, each block of this flow chart is illustrated individually.

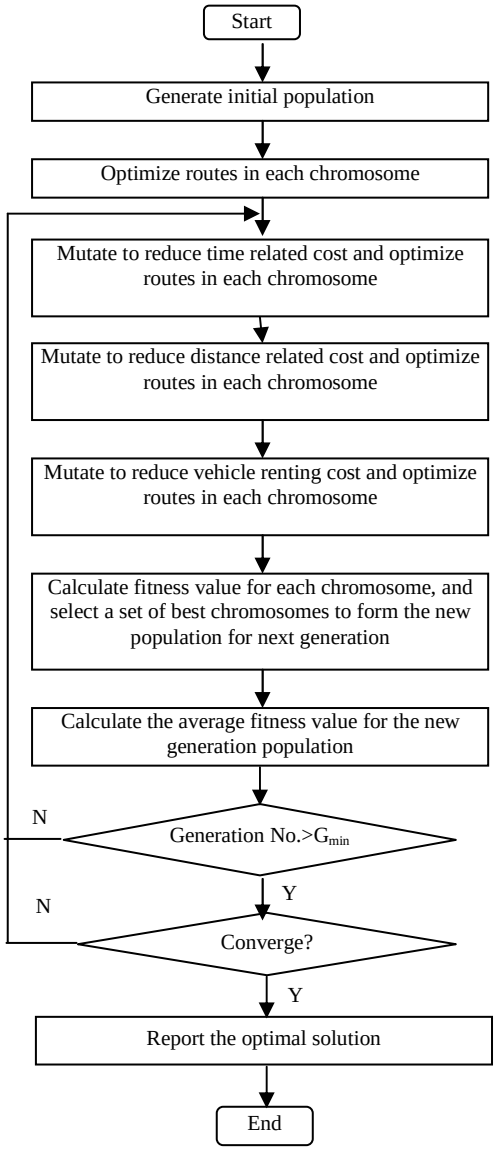


Figure 6-14. Flow chart for optimizing a population

(1) Generate initial population

A population is composed of a set of chromosomes. Because of the capacity constraint for vehicles, we can not generate chromosomes randomly too. In this model, following procedure is used to generate a chromosome. The first string of a chromosome is

generated randomly in order, and it must include all retailers in it. The corresponding second string is constructed by following way: assign vehicle *1* to the first retailer in the first string, then find the geographically closest retailer to latest assigned one, and assign vehicle *1* to it. Such process will continue until the capacity of vehicle *1* is full. Then assign vehicle *2* to the rightmost unassigned retailer, repeat the assigning process as for vehicle *1* to finish the assignment of vehicle *2*, ……. When all retailers in the first string are assigned, this chromosome is constructed. After constructing a set of such genes, a population is formed.

(2) Optimize routes

In the previous step, all retailers in a chromosome are clustered; now let's optimize its routes. According to combinatorial optimization algorithm in [Kreyzig, 1999], the shortest path in a route can be found by following procedure:

Step 1. Label wholesaler with *0*
Step 2. Set *i=0*
Step 3. Among all unlabeled retailers, find the retailer closest to wholesaler (or retailer) *i*, label it with *i+1*, and update *i (i=i+1)*.
Step 4. Go to step 3 until all retailers are labeled.

After all retailers are labeled, the shortest path is found, and the serving sequence (or the route for the vehicle) is: *0-1-2-…-k-0, k* is the number of retailers in this route.

(3) Mutation to reduce time related cost.

For each retailer, there is an ideal inter-service time which minimizes the time related cost for it. For all retailers in a route, as they are served by a vehicle, the actual inter-service time for them is same, so almost none of them will be served according to its own ideal inter-service times. If we can make the ideal inter-service times of retailers in one route as close as possible, the sum of time related costs for this route will be reduced remarkably. The mutation to reduce time related cost is designed according to this idea, and it is finished according to following steps:

Step 1. Select the route with largest time related cost in a chromosome.
Step 2. Select the retailer in this route which satisfies:

$$\max_{i \in N_j} (T_i - T_j)^2$$

Where, T_j is the inter-service time for route j.
N_j is the collection of all retailers in route j.
T_i is the ideal inter-service time for retailer i.
Step 3. Change the route for this retailer to another route which satisfies:

$$\min_{j \in M} (T_i - T_j)^2$$

Where, T_i is the ideal inter-service time for retailer *i* that is the one selected in previous step.

M is the collection of all routes in this chromosome which have enough free capacity to accept retailer *i*. If *M* is empty, give up this mutation.

(4) Mutation to reduce distance related cost.

For a route that has *k* retailers, we can use the combinatorial optimization algorithm (mentioned previously) to determine the service sequence: *0-1-2-…-k-0*. The traveling distance for circle *0-1-…-k-0* is called total traveling distance for this route, and it is used to calculate the distance related cost. The traveling distance from *1* to *k* is called service traveling distance. This mutation is used to reduce service traveling distance, and it is achieved by grouping retailers which are close to each other into a route. The mutation is finished by following steps:

Step 1. Select the route (e.g. route *j*) with largest service traveling distance.
Step 2. Select the retailer *i* (in route *j*) which satisfies:

$$\max_{i \in N_j} \sum_{j \in N_j \setminus i} d_{ij}^2$$

Where d_{ij} is the distance between retailer *i* and *j*

Step 3. Change its route number to another one which is closest to it. If impossible to find a route to accept this retailer, just give up this mutation.

(5) Mutation to reduce vehicle renting cost.

This mutation is used to reduce the number of vehicles used in a chromosome. It is finished according to following steps:

Step 1. Select the route with largest slack vehicle capacity
Step 2. Distribute all retailers in this route into other routes which have enough free capacity to accept one or two of them. After distributing all retailers in this route, the number of vehicles needed for this chromosome is reduced by one. If such distribution is impossible, give up this mutation.

As indicated in Figure 6-14, after one generation, the fitness value for each chromosome is calculated, and the best *m* chromosomes are selected to form the new population for next generation. If the average fitness value for newly formed population converges (i.e. there is no significant change at it), just stop the optimization process, select the best chromosome, and report the solution. Otherwise, continue it.

After this optimization process, the product delivery routes from one wholesaler to all its retailers are identified. The transportation model from wholesalers to retailers can be planned by this genetic algorithm model.

6.5.3 Allocating distance related cost to retailers in a route

As mentioned above, one of the purposes for us to found this optimization model is to calculate the unit product delivering cost for each retailer. But here, a vehicle serves several retailers, so we need to allocate the total distance related cost to all retailers in this route (note here, only distance related cost is related to product delivering cost and needs to be allocated). Here we take a route as example to illustrate the allocation principle.

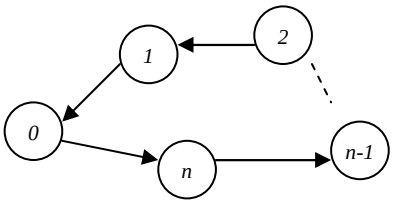


Figure 6-15. A route with *n* retailers and a wholesaler

Figure 6-15 shows a route with *n* retailers (*1~n*) and one wholesaler (*0*). The direct traveling distance from wholesaler *0* to retailer *i* is denoted as d_i. Similar to [Berman et al., 2001], following formulae are provided to allocate the travel distance A_i to retailer *i*:

$$A_i = 2d_i - R_i + \frac{1}{n}d_1 \qquad (6-38)$$

$$R_n = 2\times\frac{n-1}{n}d_n \qquad R_{n-1} = R_n + 2\times\frac{n-2}{n-1}(d_{n-1} - d_n)$$

……

$$R_2 = R_3 + 2\times\frac{1}{2}(d_2 - d_3) \qquad R_1 = R_2$$

So the unit product delivering cost for retailer *i* is

$$C_{ti} = \frac{1}{Q_i}\frac{A_i}{\sum_i A_i}C_{dj} \qquad (6-39)$$

Where, Q_i is the order quantity for retailer *i*.

C_{dj} is the distance related cost for route *j*.

Now, let's consider the product delivery routes from distribution center to wholesalers. Normally, the amount of product demanded by a wholesaler is large. A vehicle can only serve one wholesaler in its journey. So we assume that there is no routing problem in this transportation model. Simply, the unit product delivering cost from distribution center to wholesaler *j* can be calculated by:

$$C_j = C_{l0} + C_{0j} + C_{uj} \qquad (6-40)$$

Where,

C_{l0} is the unit loading cost at distribution center.

C_{0j} is the unit transportation cost from distribution center *0* to wholesaler *j*

C_{uj} is the unit unloading cost at wholesaler *j*

If there is routing problem in this transportation model, models provided in subsection 6.5.2 can be used again to determine routes for vehicles, and formula (6-39) can be used to calculate the delivery cost per unit product from distribution center to wholesalers.

So far, the method for planning product delivery routes in a distribution chain has been developed. If one vehicle serves a set of retailers in its journey, the genetic algorithm based optimization model developed in subsection 6.5.2 is used to identify the product delivery routes, and formula (6-39) is used to calculate the unit product delivery cost for each retailer. If one vehicle only serves a retailer in its journey, there is no routing problem, and formula (6-40) is used to calculate the unit product delivery cost. Next, we give an example to illustrate how to apply this method in practice.

6.5.4 Case study

Assume that we want to plan a route with one wholesaler and fifteen retailers. The parameters for this route are shown in Table 6-5 and Table 6-6. In this route, a vehicle will serve several retaielrs.

Table 6-5 Basic parameters for a route

Retailers and wholesaler	Order up to level S (unit)	Reorder point s (unit)	Mean of demand d (unit/day)	Variance of demand σ (unit)	Position (Km)
Wholesaler 0					[0, 0]
Retailer 1	500	39	30	3	[30, 20]
Retailer 2	1100	112	85	9	[40, 20]
Retailer 3	1800	180	150	10	[50, 10]
Retailer 4	1000	91	67	8	[70, 80]
Retailer 5	2000	221	170	17	[40, 50]
Retailer 6	1600	172	130	14	[80, 100]
Retailer 7	1200	104	80	8	[20, 30]
Retailer 8	1000	78	60	6	[80, 10]
Retailer 9	2000	195	150	15	[20, 50]
Retailer 10	2000	221	170	17	[30, 70]
Retailer 11	1300	117	90	9	[70, 20]
Retailer 12	1100	104	80	8	[20, 10]
Retailer 13	2000	143	110	11	[50, 20]
Retailer 14	1400	169	130	13	[30, 80]
Retailer 15	2200	195	150	15	[20, 100]

Table 6-6 Other parameters for the route

Cost coefficient for early arrival α ($/unit)	Cost coefficient for late arrival β ($/unit)	Cost coefficient for stock out γ ($/unit)	Ideal refill point R (unit)	Capacity for vehicles (unit)
1	0.5	10	3	4500

As one vehicle serves several retailers in its journey, the genetic algorithm model developed in subsection 6.5.2 will be used to solve this routing problem. This model has been realized in MATLAB, and the corresponding program is shown in B-2 of Appendix B. By running this program, the fitness value over 15 generations is shown in Figure 6-16, and the best chromosome is shown as follows:

String 1: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
String 2: 1 2 2 5 2 4 1 3 3 4 3 1 1 4 5

This routing solution can be shown as Figure 6-17.

After optimizing routes, the traveling distance can be allocated to retailers by formula (6-38), and the unit product delivering cost for each retailer can be calculated by formula (6-39).

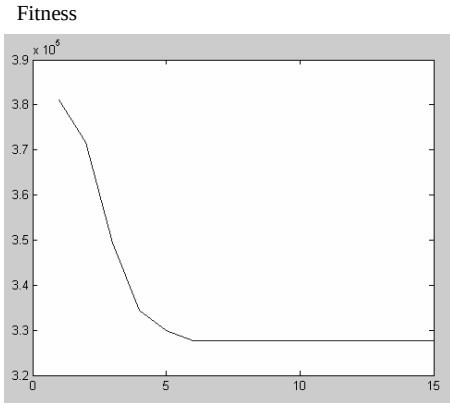


Figure 6-16 Fitness value over 15 generations

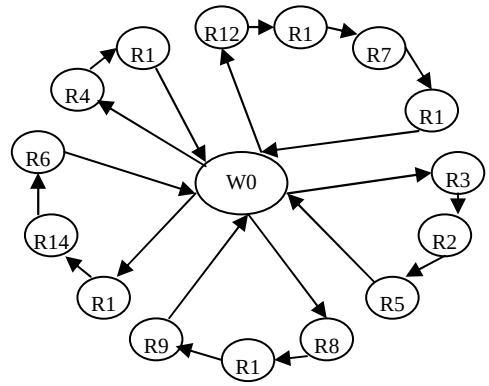


Figure 6-17 Routing solution

6.6 Summary

In this chapter, a set of models, algorithms, and formulae are developed to design a distribution chain, and the procedure to apply them is shown in Figure 6-2. In the first module of Figure 6-2, an ANN model is founded to estimate a retailer's market share for the product to be distributed in its customer zone. Based on the estimated market shares, the exact demands at all retailers are calculated. After determining customer demands at retailers, a distribution chain can be designed according to the procedure shown in the lower part of Figure 6-2. In this iterative process, first, based on given initial values for all parameters, the configuration of a distribution chain is determined by the MIP model developed in section 6.3. Given this distribution chain configuration, the inventory model at each node is optimized by the model developed in section 6.4, and routes for delivering product between different nodes are planned by the model illustrated in section 6.5. After determining inventory model at each node and product delivery routes between different nodes, all operation related parameters are re-calculated and updated. Based on these newly updated parameters, configuration of the distribution chain is re-determined, and inventory models and product delivery routes are re-optimized. This iterative process proceeds until the design process converges, i.e. there is no significant difference between successive configurations of the distribution chain. The design system provided in this chapter can turn out following results:

Design of Distribution Chain

- The configuration of the distribution chain, including number and locations of retailers and wholesalers, and the assignment of retailers to wholesalers.
- Inventory control policy and parameters at each node of the distribution chain.
- Product delivery routes between different nodes (i.e. from distribution center to wholesalers, and from wholesalers to retailers).

All models, formulae and algorithms developed in this chapter have been implemented by computer applications.

So far, the design process of a distribution chain has been finished, and the design requirements set in section 4.1 has been fulfilled. But, without verification, the designed distribution chain can not be implemented. In next chapter, to verify the design results, a simulation based model is founded to evaluate the performance of this designed distribution chain. By this evaluation, the decision makers can decide whether this designed distribution chain is good enough to be implemented.

CHAPTER 7 PERFORMANCE EVALUATION FOR THE DESIGNED DISTRIBUTION CHAIN

According to Figure 3-2, after designing a distribution chain, we need to verify the design results by performance evaluation. In this chapter, a new Petri net form, combinatorial Petri net is put forth. Then, by this newly developed Petri net form, the performance of the designed distribution chain is evaluated.

7.1 Introduction

In previous chapter, the configuration of a distribution chain, inventory control parameters at each node, and product delivering routes between different nodes were determined, i.e. the design process has been finished. As a set of less important factors was ignored during the designing process, there is no guarantee that the designed distribution chain is satisfactory even if optimization approaches have been applied. To verify the design result, in this chapter, the performance of the designed distribution chain is evaluated by a Petri net based model. Of course, the model developed here may also be used to analyze an existing distribution chain.

For a distribution chain, there are mainly two kinds of processes to be modeled: working process (e.g. the product delivering process, inventory maintaining process, etc.) at different facilities, and inferring process (e.g. when to release vehicles to serve different routes, how to ration limited vehicle capacity for different retailers, etc.) at decision making points. Normally, Petri net is good at modeling the working process, but relatively poor at simulating the inferring process. To overcome this drawback, a new Petri net form: combinatorial Petri net is developed by combining ABL with Petri net. Here, ABL is mainly used to simulate the inferring process. By this Petri net form, both working and inferring process in a distribution chain can be simulated properly.

This chapter is organized as follows: in the second section, key performance measures for a distribution chain are identified. Then in the third section, after a literature review, the method to model a distribution chain and evaluate its performance is determined. In the fourth section, the traditional Petri net is simply introduced. In the fifth section, the new Petri net form: combinatorial Petri net is illustrated. Then, in the following section, a simplified distribution chain is modeled by this new Petri net form, and its performance is evaluated. The computer applications to realize this Petri net model and the corresponding evaluation results are illustrated in the seventh section.

7.2 Key Performance Measures for a Distribution Chain

Before developing the model to evaluate a distribution chain's performance, we need to identify what to be evaluated. As distribution chain is a part of a supply chain, the performance measures used for the supply chain can partly be used for a distribution chain. Normally, for a supply chain, cost, productivity, flexibility are viewed as the key performance measures [Beamon, 1999, Solvang et al., 2001]. Besides these, lead time, equipment utilization ratio, etc. were also considered in some literatures [Miltenbueg, 1996, Viswanadham et al., 1997, Venkatesh, 1996]. As mentioned above, the main

purpose to develop this performance evaluation model is to verify the design result, and the objective to design a distribution chain is to maximize profit subject to satisfying customer requirements, so we will mainly evaluate following performance measures for a distribution chain:

(1) Profit. Profit equals total revenue minus total cost. Maximizing profit means maximizing revenue and minimizing cost simultaneously. For a designed distribution chain, profit can be used to indicate whether retailers are located reasonably, and whether the product delivering and inventory maintaining systems are organized properly.

(2) Local revenues and local costs. Local revenues are the revenues at individual retailers. They can be used to indicate the distribution of revenue, and then help decision-makers to further analyze the selection of retailers. Local costs are the costs for individual activities. They can help the decision-makers to analyze whether activities are organized properly.

(3) Utilization ratios for facilities. In a distribution chain, facilities mainly include vehicles for delivering products, buildings and equipment for maintaining inventory, etc. Obviously, more facilities mean more capacity for the distribution chain and better service for customers, but at the same time, it also means more investment. A good distribution chain design must keep a reasonable utilization ratio for facilities.

(4) Flexibility. For a distribution chain, flexibility mainly includes throughput flexibility at warehouses and transportation flexibility at different routes. Both are expressed as the slack capacity. Because of random demands at retailers, flexibility is necessary for emergent requests. Obviously, more flexibility means better service for customers, but more investment for constructing this distribution chain. Similar to utilization ratio, a good design needs to keep a reasonable flexibility for the distribution chain. As its effect is similar to that of facility utilization ratio, we will not evaluate flexibility of a distribution chain explicitly.

(5) Fill rate at retailers. Fill rate at a retailer is defined as the fraction of demand satisfied by at hand inventory. High fill rate may result in high inventory level and then more inventory carrying cost; low fill rate may cause stock out, and then loss of goodwill for the enterprise. The decision-makers need to make compromise between them.

These measures can reflect the main aspects of the performance for a distribution chain. In the following sections, we will illustrate how to evaluate them.

7.3 Method to Evaluate the Performance Measures of a Distribution Chain

In previous section, we have identified what to be measured; now, it is time to determine how to measure. A set of approaches has been developed to evaluate the performance of a distribution chain, including mathematical method, artificial

intelligence, simulation, etc. In mathematical method, some formulae were defined to calculate cost [Sabri et al, 1999], flexibility [Beamon, 1999], etc. In [Solvang, 2001], FL and ANN were used to evaluate the flexibility of a supply chain. Distribution chain is a large and complicated system, performance measures calculated by mathematical method can mainly reflect its static properties, but ignore its dynamic ones and complicated interactions between different facilities. To evaluate a distribution chain more comprehensively, some other approaches were developed.

Themido [2000] used ABC (Activity Based Costing) approach to assess logistic costs. As ABC takes activity as the center, it can partly reflect the dynamic properties of a system. But its implementation requires complete cost information and good experience in cost measurement. Obviously, these requirements are too much for a just designed distribution chain. To model the dynamic properties, some software technologies such as object-oriented software technology [Alfieri et al, 1997], multi-agent system [Swaminathan, 1998, Reis et al., 2001], were also used to simulate the working process of a supply chain, and then evaluate its performance measures. For a simulation model realized by object-oriented software or multi-agent system, the programming load is prohibitive. Especially, when the structure of a distribution chain is changed, the re-programming load is also tremendous. To model a distribution chain more efficiently, new approach needs to be developed.

Petri net is a powerful and efficient modeling approach to solve many problems related to DEDSs (Discrete Event Dynamic Systems) [Murata 1989]. Compared with other approaches, Petri net possesses following advantages:

- It is powerful to describe almost all kinds of activities in any discrete event systems. Distribution chain is a typical discrete event system. Obviously, Petri net is applicable here.
- It has strong mathematical foundation. By analyzing a Petri net model, we may understand the cause for problems like bottlenecks, deadlocks, etc. Such results are important when analyzing a distribution chain.
- It is general and open. It is not difficult to combine Petri net with other technologies. For example, FL has been combined with Petri net to form Fuzzy Petri net. As mentioned above, to simulate the working and inferring process in a distribution chain simultaneously, we need to combine ABL with Petri net. Obviously, this property is important for us.
- A Petri net model may take module structure. This makes it easier to change the model when the system is reconfigured.
- For its graphical representation, it is not difficult to understand a Petri net model.

All these properties are important for us. So, in this dissertation, we will take Petri net as the basic method to model a distribution chain, and then evaluate its performance.

7.4 Simple Introduction to Petri net

Petri net was first developed by C. A. Petri in 1962. Since then, it has been widely used to model and simulate discrete event systems. Figure 7-1 shows a simple Petri net

model. The states of the system are indicated by **places** in Petri net. Each place may contain a dynamically varying number of black dots, which are called **tokens**, as in places *S* and *Aq*. An arbitrary distribution of tokens in the places is called a **marking**. The actions of the system are indicated by rectangles, which are called **transitions** in Petri net. Places and transitions of a Petri net are collectively referred as **nodes**. The Petri net also contains a set of directed arrows as indicated in Figure 7-1, which are called **arcs**. Each arc connects a place with a transition, or a transition with a place. Each arc may have a positive integer attached to it, e.g. *2* is attached to the arc from place *Aq* to transition *T1q*. this integer is called the **arc expression**. A node (place or transition) *x* is called an **input node** of another node *y* when there is an arc from *x* to *y*. Analogously, node *x* is called an **output node** when there is a directed arc from *y* to *x*.

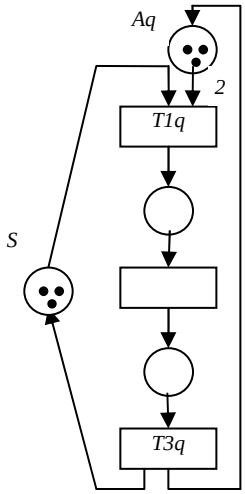


Figure 7-1 A Petri net model

The working process of a Petri net can be explained as follows. Each transition represents a potential action in the model. An action becomes possible when each input place of the transition contains at least the number of tokens prescribed by the arc expression of the corresponding input arc, then we say that the transition is **enabled**. When a transition is enabled, the corresponding action may take place. If this happens, we say that the transition **occurs**. The effect of an occurrence is that, tokens are removed from input places and added to the output places. The number of removed/added tokens is specified by the arc expression of corresponding input/output arc.

To model more complex systems, some high level Petri nets are developed, such as colored Petri net, timed Petri net, etc. In these high level Petri nets, complex arc expressions may be used to describe the condition for working processes, or define the transition delay to describe time needed for an action. But the principle for checking the enabled transition and firing a transition is similar to the ordinary Petri net mentioned above.

7.5 Combinatorial Petri Net

For the existing Petri net forms, when checking whether a transition is enabled, they separately make comparison between arc expressions and their corresponding input places' states, but the relationship between different input places is not considered. They do not have capability to simulate the inferring process for deciding whether or not a transition is enabled. For example, when deciding when to release vehicles to serve retailers, the inventory levels at these retailers need to be considered synthetically, and the decision is made based on a set of pre-defined rules. Existing Petri net forms do not possess capability to simulate this decision making process. To supplement its modeling capability, an inferring technique needs to be combined with Petri net.

As mentioned previously, ABL is a promising inferring technology. It is good at solving large scale inferring problems with quantitative and logic variables. By combining ABL with Petri net, a new Petri net form: combinatorial Petri net is developed. The main difference between traditional Petri net and combinatorial Petri net is that, in traditional Petri net, a transition is enabled when each input place contains more tokens than the number defined by corresponding arc expression. In combinatorial Petri net, all states of the transition's input places are considered synthetically, and an inferring process is used to decide whether this transition is enabled. This inferring process is finished by ABL, and the inferring rules are predefined by decision-makers. Compared with traditional Petri net, there are mainly following two improvements in combinatorial Petri net.

(1) Improvement in enabling rule

The enabling rule for traditional Petri net form is shown in formula (7-2) (mentioned next). It means that, a transition is enabled when the numbers of tokens possessed by its input places are more than or equal to the ones specified by corresponding arc expressions. Actually, this enabling rule can only guarantee that the input places possess enough resource to support the occurrence of this action. But, in reality, holding enough resource can not make sure that the action may happen. Sometimes, when checking whether an action is enabled, the relevant statuses need to be evaluated synthetically, and the decision is made by inference. Obviously, traditional Petri net does not have capability to simulate this inferring process. To simulate this complicated decision making process, we combined ABL with Petri net in the combinatorial Petri net, and ABL is the one that will accomplish this inferring process. In combinatorial Petri net, when forming the Petri net model, a set of inference rules (premises) may be defined for a transition. Then, as mentioned in subsection 5.4.4.2, ABL technology is used to connect these premises by appropriate connector to form an ABL inference module, and this module is represented by a function. Here, we call this function as *ABL* (any other name may also be used). Function *ABL* takes the markings of all input places for a transition as its arguments, and returns a logic value to indicate whether this transition is enabled. Each transition has its own *ABL* function. When checking whether a transition is enabled, the checking process is accomplished by two steps. In the first step, all input places for this transition are checked whether there is enough resource to support the occurrence of this action. If the answer is "yes", then the checking process comes to the second step; otherwise, this transition is not enabled. In the second step, the markings of all input places for this transition are input into its *ABL* function that is formed above. This function will output a logic value to indicate whether this transition is enabled. If no inference rule is defined for a transition, then its *ABL* function automatically returns *1*. In such situation, the enabling rule in combinatorial Petri net is same as the one in traditional Petri net.

(2) Improvement in firing rule

In traditional Petri net, the firing rule is shown in formula (7-3) (mentioned next). When a transition is fired, tokens are removed (or added) in its input (or output) places according to corresponding arc expressions. In reality, sometimes, the result for the

occurrence of an action can not be simply expressed by adding or removing tokens. For example, when we want to evaluate the cost for an activity, some simple calculation needs to be executed. Obviously, traditional Petri net can not fulfill this requirement. To make the Petri net model more general, in combinatorial Petri net, we specify the firing rule as: when a transition is fired, all its input and output arc expressions are directly executed. Such arc expressions may include almost any kind of calculations, including the operation of adding and removing tokens.

To compare combinatorial Petri net with traditional Petri net, the definition, enabling rule and firing rule for both of them are illustrated next.

7.5.1 Traditional Petri net

As combinatorial Petri net is extended from colored Petri net, we take colored Petri net as an example to illustrate the basic concepts for traditional Petri net forms. According to [Jensen, 1992], the definition, enabling rule and firing rule for colored Petri net are given as follows.

Definition. A colored Petri net is defined as:

$$CPN = (\Sigma, P, T, A, N, C, G, E, I) \qquad (7\text{-}1)$$

Where, Σ is a finite set of non-empty types, called color sets here. All token colors for a given place must belong to a specified type. This type is called color set. The set of color sets determines the types, operations, functions that can be used in the net inscriptions.

P is a finite set of places.

T is a finite set of transitions.

A is a finite set of arcs such that: $P \cap T = P \cap A = T \cap A = \Phi$.

N is a node function. It is defined from A into $P \times T \cup T \times P$. This node function maps each arc into a pair where the first element is the source node and the second is the destination node.

C is a color function. It maps each place to a color set.

G is the guard function. It maps each transition into an expression of type Boolean.

E is the arc expression function. It maps each arc into an expression.

I is the initialization function. It is defined from P into closed expressions (an expression without variables is said to be closed expression).

Enabling rule. For a transition, when each of its input places contains at least the tokens to which the corresponding arc expression evaluates, this transition is enabled. A step Y is *enabled* in a marking M iff the following property is satisfied:

$$\forall p \in P: \sum_{(t,b)\in Y} E(p,t)\langle b \rangle \le M(p) \qquad (7-2)$$

Firing rule. When a step is *enabled*, it may occur, and this means that tokens are removed from input places and added to the output places of the occurring transitions. When step Y is enabled in a marking M_1, it changes marking M_1 into marking M_2, defined by:

$$\forall p \in P : M_2(p) = (M_1(p) - \sum_{(t,b)\in Y} E(p,t)\langle b\rangle) + \sum_{(t,b)\in Y} E(t,p)\langle b\rangle \qquad (7-3)$$

In this formula, the first sum is called the removed tokens, while the second one is called the added tokens

7.5.2 Combinatorial Petri net

To some extent, combinatorial Petri net can be viewed as an extension of colored Petri net. In what follows, we use mathematical models and examples to elucidate the definition, enabling rule and firing rule for a combinatorial Petri net.

7.5.2.1 Definition

First, we give the definition of combinatorial Petri net. Some explanations for the individual parts of this definition are given immediately below the definition. Analogous to colored Petri net, a combinatorial Petri net, *ComPN*, is defined as:

$$ComPN = (\sum, P, T, A, N, C, E, I) \qquad (7\text{-}4)$$

Where, $\sum$ is a finite set of non-empty types, called color sets here.
P is a finite set of places.
T is a finite set of transitions.
A is a finite set of arcs such that: $P \cap T = P \cap A = T \cap A = \Phi$.
N is a node function. It is defined from A into $P \times T \cup T \times P$.
C is a color function. It maps each place to a color set.
E is the arc expression function. It is defined from A into expressions.
I is the initialization function, it is defined from P into expressions.
No guard function is defined in combinatorial Petri net.

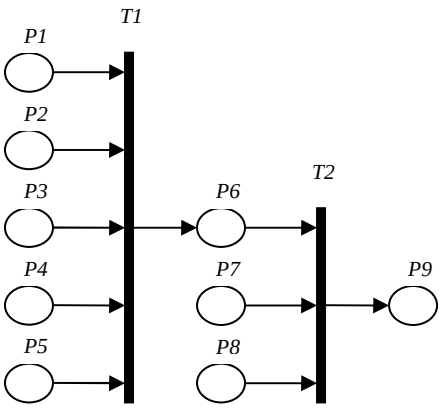
(1) A **color set** is a type. In combinatorial Petri net, a color set may be primitive data type (such as integer, real, etc.) or user defined type (such as structure, etc.). As combinatorial Petri net accepts user defined type to describe the color of a token, this makes it powerful and extensible in modeling complex situations.

(2) A **place** represents a state. In a place, there may be one or more tokens. The color of a token may be described by a primitive data type or a user defined type, and the marking of this place is described by an array of tokens. Figure 7-2 shows a simplified combinatorial Petri net model for vehicle releasing at a wholesaler (the working process of this model will be explained later). In this model, place *P7* indicates the available vehicles at the wholesaler. In this place, one token represents one vehicle. As we only care about the serial number of a vehicle, so the color of a token can be

described by an integer (e.g. *1*) to indicate the vehicle No., and the marking of this place can be described by an array. For example, *P7=[1 2]* means there are two vehicles at the wholesaler, and they are vehicle *1* and vehicle *2*. Place *P9* indicates the loaded vehicles at the wholesaler. A token represents a loaded vehicle, and its color can be described by following structure:

```
struct loadedVehicle {
                        int vehicleNumber;
                        float amount;
                      }
```

There are two members in this structure: the first one indicates the vehicle No., and the second one indicates the amount of product shipped on the vehicle. The marking of this place can be described by an array of this structure to indicate all loaded vehicles at the wholesaler.



Places and transitions:
P1-P4: inventory level at retailer1-4
P5: place to indicate whether route 1 is being served
P6: pool of route(s) that need to be served.
P7: available vehicles at the wholesaler
P8: inventory level at the wholesaler
P9: loaded vehicle at the wholesaler
T1: inferring process for the vehicle releasing of route 1
T2: loading and vehicle releasing process at the wholesaler

E_2 arc expressions:
IA5-1: P5.delRoute(1)=P5.route(1); P5.route(1)=[];
OA1-6: P6.VRI=[P6.VRI T1.abl], P6.route=[P6.route P5.delRoute(1)], P6.amount=[P6.amount SUM(I_1, I_2, I_3, I_4)];
IA6-2: I=MAX(P6.index), P6.delRoute(1)=P6.route(I), P6.route(I)=[ ], P6.delAmount(1)=P6.amount(I), P1.amount(I)=[ ], P6.VRI(I)=[];.
IA7-2: P7.delVehicle(1)=P7.vehicle(1), P7.vehicle(1)=[ ];
IA8-2: P8.inventory(1)=P8.inventory(1)-P6.delAmount(1);
OA2-9: P9.vehicle=[P9.vehicle P7.delVehicle(1)], P9.amount=[P9.amount P6.delAmount(1)];

Note (here and throughout): OA represents output arc expressions, e.g. OA1-6 indicates output arc expressions from transition 1 to place 6; IA represents input arc expressions, e.g. IA5-1 indicates the input arc expressions from place 5 to transition 1. Convention used here is compliant to the one in MATLAB.

Figure 7-2 A simple combinatorial Petri net model

(3) A **transition** represents a possible action. The attribute of a transition is described by following structure:

```
struct transition {
                    int delay;
                    int firingTime
                   int abl;
                   }
```

In this structure, the first member *delay* indicates the time needed to accomplish this action. It can be deterministic or random. The second member *firingTime* indicates when to fire this transition. The third member *abl* is an integer to indicate the inference result. As mentioned above, in combinatorial Petri net, a set of inference rules can be defined for a transition. ABL is used to form a function to accomplish the inferring process. This function takes the current markings of all the transition's input places as its arguments, and returns an integer to indicate the inference result. This result is assigned to *abl.* Detail on this inferring process will be explained later in the enabling rule.

(4) The meanings of **arc**s, **node**s, and **color**s are same as the ones in colored Petri net.

(5) As shown in Figure 7-3, an arc connects a place and a transition. For each arc, we can attach **arc expression**s to it. In combinatorial Petri net, there are two types of arc expressions: E_1 and E_2. E_1 specifies the number of tokens needed for the occurrence of this transition. It will be used in enabling rule. E_2 tells what will happen when the transition is fired. It will be used in firing rule. For an input arc (the arc connecting an input place and a transition), both E_1 and E_2 are attached. For an output arc (the arc connecting a transition and an output place), only E_2 is attached. If no E_1 arc expression is defined for an input arc, we automatically assign default value *1* to it. This means that at least one token is needed from the input place for the occurrence of this transition. If no E_2 is defined for an arc, then nothing will happen for the place when this transition is fired. This means that the occurrence of this transition has no influence on the place. To explain how the arc expressions work in combinatorial Petri net, we take one node from Figure 7-2, and show it in Figure 7-3. In this node, place *P7* represents the available vehicles at the wholesaler, and transition *T2* represents the vehicle loading process at this wholesaler. No E_1 arc expression is defined for this input arc, so we assign *1* to it. This means that, at least one token (vehicle) is needed from place *P7* for the occurrence of *T2*. When *T2* is fired, *E2* arc expression is executed directly. The result is: the first token in place P7 is deleted by *P7.vehicle(1)=[]*, and the information on this deleted token is temporarily stored in *P7.delVehicle(1)* by *P7.delVehicle(1)=P7.vehicle(1).*

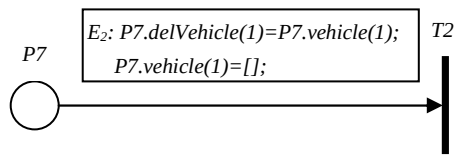


Figure 7-3 A node in combinatorial Petri net

(6) In the **initialization function**, both number of tokens for each place and color for each token must be specified.

7.5.2.2 Enabling rule

In combinatorial Petri net, two steps are used to check whether a transition is enabled. In the first step, all the input places for the transition are checked whether they have enough token(s) to support the occurrence of this transition. If each input place of the transition contains at least the number of tokens prescribed by the *E1* expression of the corresponding input arc, then the transition is **token-enabled**. A token-enabled transition is qualified to enter the second checking process. If a transition is not token-enabled, it will not be enabled. As mentioned above, a set of inference rules can be defined for a transition. In the second step, ABL is used to form a function, *ABL()*, based on this set of inference rules. *ABL()* takes the current markings of the input places of the transition as its arguments, and returns an integer to indicate whether or not this transition can be enabled. If the value returned by *ABL()* is non-zero, then the transition is **enabled**; otherwise, it is not. If no inference rule is defined for a transition, the function *ABL()* for this transition automatically returns *1*. In such situation, the enabling rule for combinatorial Petri net is same as the one for colored Petri net. In what follows, we use an example to illustrate how the enabling rule works in combinatorial Petri net.

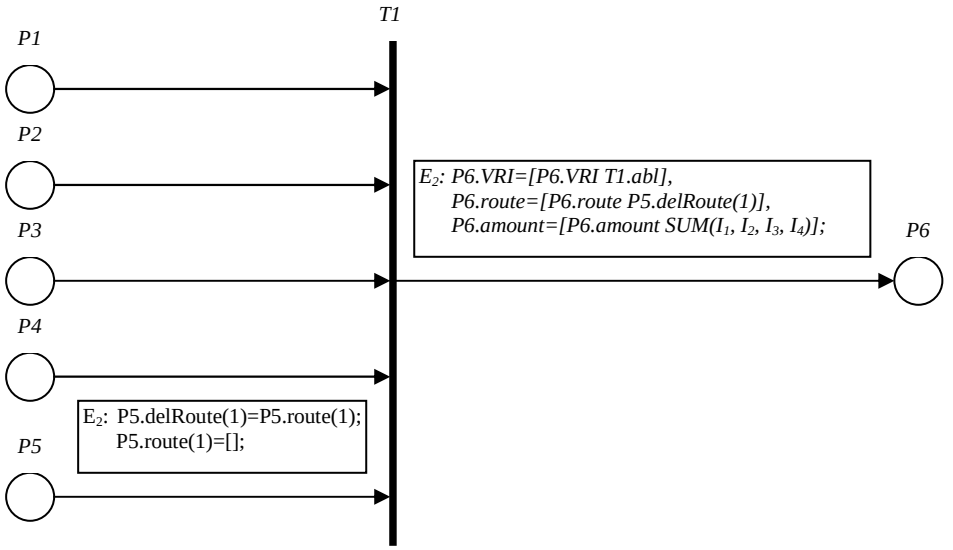


Figure 7-4 A transition with its input and output places

We take transition *T1* and its input and output places from Figure 7-2, and show them in Figure 7-4. In this model, transition *T1* simulates the decision-making process on whether or not to release a vehicle to serve a route (say route *1*). *T1* has five input places: *P1-P4* represent the inventory states of retailer R_1-R_4 (retailers in route *1*), *P5* indicates whether the route is being served. If it is being served, *P5* is empty; otherwise, a token exists in *P5*, and the color of this token indicates the route number. Transition *T1* has one output place: *P6*, which represents the decision. The color of token(s) in *P6*

tells which route needs to be served, and how much product needs to be loaded on the vehicle. Next, we use this example to illustrate how enabling rule works in combinatorial Petri net.

As mentioned above, there are two steps in checking whether a transition can be enabled. In the first step, all input places for the transition are checked whether they have enough token(s) to support the occurrence of the transition. For transition *T1*, there are five input places: *P1-P5*, and no *E1* arc expression is defined for their corresponding input arcs. According to the rule stated in the definition of arc expression, default value *1* will be taken for these *E1* arc expressions. It means that, if there is at least one token in places *P1-P5*, transition *T1* will be token-enabled. Place *P1-P4* indicates the inventory state of retailer R_1-R_4, so there is always one token in each of them. Therefore place *P5* can decide whether *T1* can be token-enabled. *P5* indicates whether the route is being served: if it is not being served, there is one token in place *P5*, and its color is the route number; otherwise, place *P5* is empty. So, the meaning of this checking process is: if the route is not being served, we will always check whether it needs to be served. Here, we assume that route *1* is not being served, then, place *P5* is not empty, and transition *T1* is token-enabled. A token-enabled transition is qualified to enter the second step of the checking process.

In the second step, ABL will be used to accomplish the inferring process, and the inferring result decides whether or not this transition is enabled. In Figure 7-4, transition *T1* simulates the decision making process on whether to release a vehicle to serve a route. In reality, such decision is made based on following premises:

I_1-L_1>0 AND I_2-L_2>0 AND I_3- L_3>0 AND I_4-L_4>0 IMPLIES *VRI is 0*
I_1-L_1<0 AND I_2-L_2>0 AND I_3- L_3>0 AND I_4-L_4>0 IMPLIES *VRI is 1*
……
I_1-L_1<0 AND I_2-L_2<0 AND I_3- L_3>0 AND I_4-L_4>0 IMPLIES *VRI is 2*
……
I_1-L_1<0 AND I_2-L_2<0 AND I_3- L_3<0 AND I_4-L_4>0 IMPLIES *VRI is 3*
……
I_1-L_1<0 AND I_2-L_2<0 AND I_3- L_3<0 AND I_4-L_4<0 IMPLIES *VRI is 4*

Where,

I_i: the inventory level for retailer *i*.

L_i: the expectative lead time demand for retailer *i*. So I_i-L_i is the expectative inventory level when the vehicle arrives at retailer *i*.

VRI: the abbreviation of Vehicle Releasing Indicator. *VRI* is an integer ranging from 0 to 4. It is used to indicate the degree of urgency for a route to be served. *VRI=0* means the route is unnecessary to be served; *VRI=1 ~ 4* means the route needs to be served, and the larger the value is, the more urgently the route needs to be served.

Here, we only give these premises as an example. In practice, what kind of premises to be used depends on the decision maker's opinion. At the left side of these premises, L_i

(i=1~4) are parameters given by experience, and I_i *(i=1~4)* are variables indicating the present inventory level for retailers. At the right side, *VRI* represents the inferring result. These premises form an inference system. Every time when we present values of variables I_i onto the system, it will return a *VRI* value to indicate whether this route needs to be served. This is just like the mapping process of a function, so we call this inference system as a function, and express it as:

$$VRI=ABL(I_1, I_2, I_3, I_4) \quad (7\text{-}5)$$

Here, we only borrow the concept of function to express an inference system. Of course, the function shown in formula (7-5) is different from normal mathematical functions, as we can not realize an inference system by mathematical operators. In what follows, we will illustrate how to get this function (inference system) by ABL. As mentioned in subsection 5.4.4.2, there are three steps in forming an inference system by ABL: define the global domain, model the system, and interact with the environment.

Define the global domain. In this inference system, there are five variables: I_1~I_4 and *VRI*. For the inventory level of retailer *i*, we divide its possible range into two intervals: $[I_{mini}, L_i]$ and $(L_i, I_{maxi}]$. For the output variable *VRI*, we use an integer ranging from *0* to *4* to express it. So, the global domain of this inference system can be shown as Table 7-1.

Table 7-1 Global domain for the inference system of vehicle releasing

I_1	$\{[I_{min1}, L_1], (L_1, I_{max1}]\}$	I_2	$\{[I_{min2}, L_2], (L_2, I_{max2}]\}$
I_3	$\{[I_{min3}, L_3], (L_3, I_{max3}]\}$	I_4	$\{[I_{min4}, L_4], (L_4, I_{max4}]\}$
VRI	*{'0', '1', '2', '3', '4'}*		

In SABL, the domain of a variable is defined by function *element*. For example, the domain of variable R_1 can be defined by following statements:

lowerIntervalR1 = interval ('ge', I_{min1}, 'le', L_1);
upperIntervalR1 = interval ('gt', L_1, 'le', I_{max1});
I_1=element ('n', {lowerIntervalR1, upperIntervalR1}, {}, 'Inventory of retailer R1');

Model the system. Modeling the system means defining premises and connecting them by appropriate logic operators. The premises for this inference system are shown above. All these premises can be modeled by SABL. For example, the first premise can be expressed by following statements:

```
x11=assign (I1, upperIntervalR1);
x12=assign (I2, upperIntervalR2);
x13=assign (I3, upperIntervalR3);
x14=assign (I4, upperIntervalR4);
y1 = assign (VRI, {'0'});
y12=conjunct (x11, x12);
y34=conjunct (x13, x24);
```

y1234=conjunct (y12, y34);
premise_1=bimp (y1234, y1)

All premises are connected by logic operator AND. In SABL, this is realized by function *conjunct*. For example, premise *1* and premise *2* can be connected by:

Premise_12 = conjunct (Premise_1, Premise_2);

This connected system is realized by program in SABL. Conceptually, it can be viewed as a function shown in formula (7-5). Every time when we input values of I_1~I_4 into this function, the resulted *VRI* is obtained. Note here, this function is a program rather than a set of mathematical operations.

Interact with environment. This interaction process can be expressed as: forming the input vector, presenting this input vector to the system modeled above, and then computing the output vector. In SABL, this process is accomplished by function *state*, e.g.:

output_SV = state (input_SV, System);

Here, *input_SV* is the input vector, *System* is the inference system formed by connected premises, and *output_SV* is the output vector.

Every time when checking whether transition *T1* is enabled, the current markings for input places of transition *T1* (i.e. values of variables I_1~I_4) form the input vector. This input vector is input into function (7-5), and the function returns a value of variable *VRI*. If this value is non-zero, then transition *T1* is enabled; otherwise, it is not.

Now, we use mathematical model to express the enabling rule in combinatorial Petri net. A transition $t_j \in T$ is enabled iff:

$$\forall p \in I(t_j) : |p| < E_1(p, t_j) \qquad \&\& \qquad ABL\left(\sum_{p \in I(t_j)} M(p)\right) \neq 0$$

Where,
$I(t_j)$: collection of all input places for transition t_j. So, here, p presents any one of the input places for transition t_j.
$|p|$ is the number of tokens in place p.
$E_1(p, t_j)$: E_1 arc expression from place p to transition t_j.
$M(p)$: the marking of place p. So, $\sum_{p \in I(t_j)} M(p)$ represents the collection of markings for all input places of transition t_j.
$ABL\left(\sum_{p \in I(t_j)} M(p)\right)$: t_j's *ABL* function which takes the current markings of input places for transition t_j as arguments, and returns an integer to indicate whether this transition can be enabled.

In other words, in combinatorial Petri net, a transition t_j is enabled if and only if there are enough tokens in each input place of t_j, and the value returned by inferring function *ABL()* is non-zero. Following pseudo-code can be used to express this checking process:

if (there are enough tokens in input places of transition t_j)
  if ($ABL(\sum_{p \in I(t_j)} M(p))$)
    transition t_j is enabled
  end
end

If a set of premises are defined for transition t_j, ABL is used to form function $ABL(\sum_{p \in I(t_j)} M(p))$ according to following procedure [Møller, 1995]:

- Define domain for all variables.
- Model each premise, and connect them by appropriate logic operator to form the inference system. This inference system can be viewed as function $ABL(\sum_{p \in I(t_j)} M(p))$ which takes markings of all input places for transition t_j as arguments, and returns a value to indicate whether or not this transition can be enabled.
- Input the current markings of all input places for transition t_j into function $ABL(\sum_{p \in I(t_j)} M(p))$, then it returns an integer to indicate whether t_j is enabled.

If no premise is defined for transition t_j, function $ABL(\sum_{p \in I(t_j)} M(p))$ returns *1*. For such situation, the enabling rule in combinatorial Petri net is same as the one in traditional Petri net.

If there is no input place for a transition, this transition is always enabled.

7.5.2.3 Firing rule

After checking all transitions by enabling rule, we come to the next step: firing the enabled transitions. For each transition, there may be a set of input and output places. Correspondingly, for each place, there is an arc (input or output arc) to connect this place to the transition. For each arc, one or more E_2 arc expressions may be attached to it. Firing a transition means executing all these E_2 arc expressions related to the transition. Executing E_2 arc expressions can be expressed by following formulae:

$$M_2(p_i)=IE_{2(pi\text{-}tj)}(M_1(p_i)) \quad \text{for all } p_i \in I(t_j) \qquad (7\text{-}6)$$
$$\text{And} \quad M_2(p_k)=OE_{2(tj\text{-}pk)}(M_1(p_k)) \quad \text{for all } p_k \in O(t_j)$$

Where,

$M_2(p_i)$: the new marking for place p_i, and $M_1(p_i)$ is its old marking.

$IE_{2(pi\text{-}tj)}(M_1(p_i))$: input E_2 arc expressions from p_i to t_j, $IE_{2(pi\text{-}tj)}(M_1(p_i)) \in E$

$OE_{2(tj\text{-}pk)}(M_1(p_k))$: output E_2 arc expressions from t_j to p_k, $OE_{2(tj\text{-}pk)}(M_1(p_k)) \in E$

$I(t_j)$: collection of all input places for transition t_j

$O(t_j)$: collection of all output places for transition t_j

These formulae show that, E_2 arc expressions take old marking as their arguments, and return new markings to the corresponding places. When a transition is fired, the markings of relevant places are updated by this way.

As mentioned in the definition of a combinatorial Petri net, every transition has a parameter *delay* to indicate how long time it needs to accomplish this action. If a transition's *delay* is zero, we call it as zero-*delay* transition; otherwise, we call it as non-zero-*delay* transition. Next, we illustrate when to execute its input and output E_2 arc expressions when a transition is fired.

For the zero-*delay* transitions, when they are enabled, they are fired immediately, and their input and output E_2 arc expressions are executed simultaneously.

For the non-zero-*delay* transitions, when they are enabled, their input E_2 arc expressions are executed immediately, but their output E_2 arc expressions will be executed later. Similar to timed Petri net, a **global clock** is introduced into a combinatorial Petri net model, and its value represents the **model time** which starts at *0*. When transition t_j is enabled, its input E_2 arc expressions are executed immediately, and its parameter *firingTime* is updated to the current model time plus its *delay*. Then, this transition is put into **event queue**. After putting all enabled non-zero-*delay* transitions into event queue, we take one transition which has smallest *firingTime* from event queue, and fire it, i.e. execute its output E_2 arc expressions. After firing this transition, the model time proceeds to this transition's *firingTime*, and this cycle is finished. Then, the process goes to next cycle, i.e. checking all transitions by enabling rule again, and firing a part of the enabled transitions according to the rules mentioned above.

The flow chart to run a combinatorial Petri net model can be shown as Figure 7-5. When running a combinatorial Petri net model, first, it is initialized, i.e. the *model time* is set to *0*, and all places are given initial markings. Secondly, all transitions are checked by enabling rule. For the enabled transitions, we fire them according to following rules: for zero-*delay* transitions, fire them immediately; for non-zero-*delay* transitions, execute their input E_2 arc expressions, update their *firingTime*, and put them into event queue. Then, we only select one transition which has smallest *firingTim*e from the event queue, and fire it. After firing this transition, the *model time* proceeds to its *firingTime*. According to this flow chart, every time after a set of

transitions are fired, the ending condition (e.g. the maximal running time, etc.) is checked. If it is satisfied, just end the running process; otherwise, a new cycle begins.

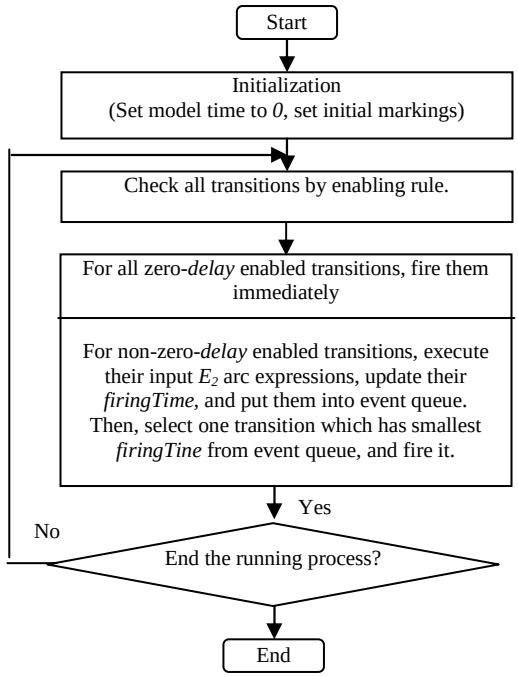


Figure 7-5 Flow chart to run a combinatorial Petri net model

7.5.2.4 Case study

To illustrate the application of combinatorial Petri net, we give this case study based on the example shown in Figure 7-2. In this case study, we mainly concentrate on three things: what an initialized combinatorial Petri net model looks like, how to check a transition by enabling rule, and how to fire an enabled transition. In what follows, we first initialize the combinatorial Petri net model shown in Figure 7-2, and then run it step by step.

(1) Initialization

According to Figure 7-5, the first step to run a combinatorial Petri net model is to initialize it. The initialized combinatorial Petri net model for the example shown in Figure 7-2 is shown in Figure 7-6.

Places *P1~P4* and *P8* represent the inventory status at retailers R_1~R_4 and wholesaler. Following structure is used to describe the inventory status for a warehouse:

```
Structure inventory
                    {
                     float inventoryLevel;
                    int updatedTime;
                    }
```

The first member indicates the present inventory level for the warehouse, and the second member indicates the latest time when this warehouse's inventory is updated. This second member is used to calculate the inventory carrying cost. It is useless in this case study. In Figure 7-6, the marking in place *P1* means that, the present inventory level at retailer R_1 is *65 (units)*, and this inventory was updated at time *0*.

Place *P5* indicates whether a route is being served. In Figure 7-6, a token is placed in *P5*, and the color of this token is *1*. This means route *1* is not being served for the moment.

Place *P6* represents the decision on whether to serve a route. As no decision is made yet, it is empty.

Place *P7* indicates the available vehicles at the wholesaler. There are two tokens in it, and the colors for these two tokens are *1* and *2*, it means there are two vehicles at the wholesaler, and the vehicle No. for them are *1* and *2*.

Place *P9* represents the loaded vehicle at the wholesaler. As no vehicle is loaded now, it is empty.

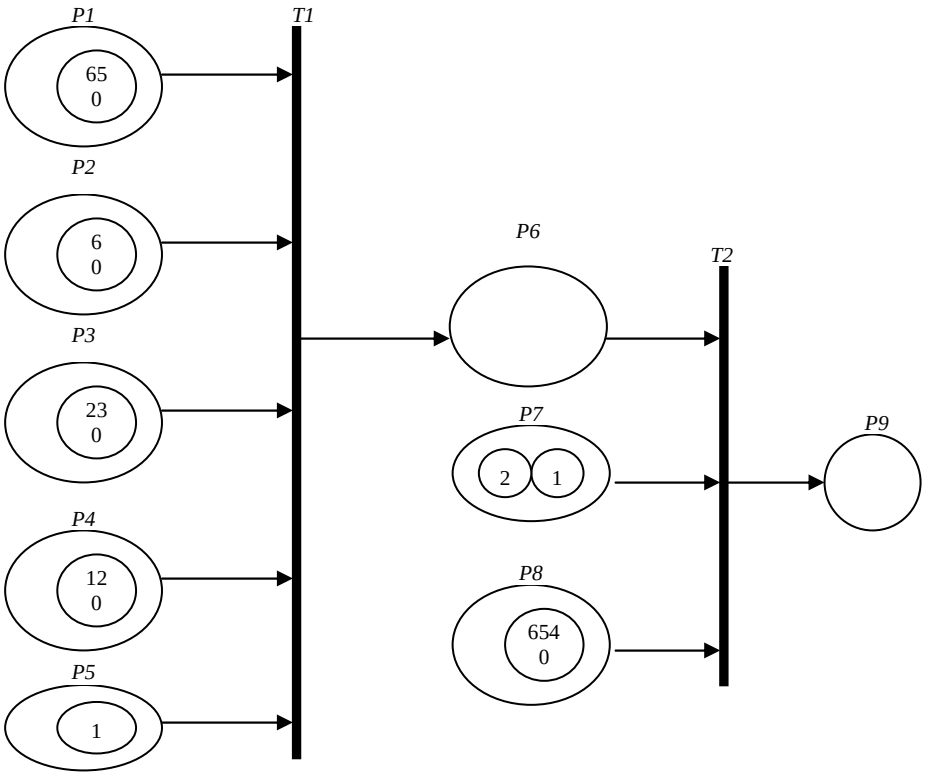


Figure 7-6 The initialized combinatorial Petri net model for the example shown in Figure 7-2

The global clock *modelTime* is set to *0*.

Transition *T1* represents the decision making process on whether route *1* needs to be served. As mentioned in subsection 7.5.2.1, there are three parameters to indicate the

attribute of a transition: *delay*, *firingTime* and *abl*. For transition *T1*, assume that a decision can be made immediately after getting the needed information, so: *T1.delay=0*. As *T1*'s *delay* is zero, so it will not be put into event queue, and its *firingTime* is useless. The premises for *T1* are same as the ones shown in subsection 7.5.2.2. *T1.abl* will be assigned after the inference process is finished. For the moment, its value is arbitrary.

Transition *T2* represents the vehicle loading process at the wholesaler. Assume that it needs *5 (units)* time to finish this loading process, so *T2.delay=5*. *T2.firingTime* will be assigned when it is put into the event queue. As no premise is defined for this transition, *T2.abl=1*.

(2) Checking transitions by enabling rule

For transition *T1*, no E_1 arc expressions are attached to all its input arcs. This means at least one token is needed in each input place to make *T1* token-enabled. As shown in Figure 7-6, there is one token in places *P1~P5*, so *T1* is token-enabled. The premises for this transition are shown in subsection 7.5.2.2. Based on these premises, the inference system is formed by ABL, and it can be represented by function (7-5). Actually, the inference system is realized by a program, and it is conceptually represented by a function. There are two kinds of inputs for this function: parameters L_1~L_4 and variables I_1~I_4. L_i indicates the lead time demand for retailer *i*; I_i represents the current inventory level for retailer *i*, so I_i=*Pi.inventoryLevel* (e.g. I_1=*P1.inventoryLevel=65*, etc.). These inputs are shown in Table 7-2.

Table 7-2 Parameters for retailer *1*~retailer *4*

Retailer No.	I_i	L_i	S_i
1	65	30	300
2	8	20	300
3	23	20	300
4	12	25	300

Inputting L_i and R_i into function (7-5), we can get the output *VRI=2*. Then this value is assigned to *T1.abl*, i.e. *Ti.abl=VRI=2*. As this value is non-zero, *T1* is enabled.

For transition *T2*, no E_1 arc expression is attached to all its input arcs too. This means at least one token is needed in each input place to make *T2* token-enabled. But, place *P6* is empty, so *T2* is not token-enabled, and it is not enabled.

(3) Firing the enabled transition(s)

Here, only transition *T1* is enabled. As its *delay* is zero, so it is fired immediately. No E_2 arc expressions are defined from places *P1~P4* to *T1*. This means the firing of transition *T1* has no influence on these places, so their markings are kept unchanged. This is reasonable. When we make the decision on whether to serve a route, we only

need the information on the inventory status of retailers in this route, but the inventory status itself is not changed. After making the decision, it is no longer necessary to evaluate the inventory status of retailers in this route, so the token in place *P5* is deleted by *P5.route(1)=[];* As the color of this token (i.e. the route No.) will be used in the future programming, so we store this information temporarily in a deleted token by *P5.delRoute(1)=P5.route(1);* This deleted token is just used to store information, it has no effect when evaluating whether a transition can be enabled.

After evaluating the current inventory status of retailers in a route, a decision on whether or not to send a vehicle to serve this route is made. The decision is expressed by a token, and this token is added to place *P6*. We use following structure to express the color of this token:

```
Structure decision
                    {
                     int VRI;
                    int route;
                    float amount;
                    }
```

The first member, *VRI*, is the Vehicle Release Indicator for this route. It indicates the degree of urgency for the route to be served. Its value is returned by function (7-5), and here it is *2*. The second member *route* represents the route No. As indicated by the color of token in place *P5*, here it is *1*. The third one *amount* indicates how much product will be loaded onto the vehicle when releasing it. Following formula is used to determine it:

$$Q = SUM(I_1, I_2, I_3, I_4) = \sum_{i=1}^{4} (S_i - I_i + L_i) \qquad (7-7)$$

Where,

S_i: the order up to level for retailer *i*
I_i: the present inventory level for retailer *i*
L_i: the expectative lead time demand for retailer *i*

Inputting all parameters shown in Table 7-2 into function (7-7), we get $Q=SUM(I_1, I_2, I_3, I_4) = 1187$. All the calculating results for place *P6* are shown in Table 7-3.

Table 7-3 Calculating results for the marking of place *P6*.

Members in structure *decision*	The results come from:	Results
VRI	*T1.abl*	2
route	*P5.delRoute(1)*	1
amount	$SUM(I_1, I_2, I_3, I_4)$	1187

The token is added to place *P6* by following expressions: expression *P6.VRI=[P6.VRI T1.abl]* adds the latest inferring result into the array. Expression *P6.route=[P6.route*

P5.delRoute(1)] adds the route No. into the queue, and expression *P6.amount=[P6.amount SUM(I_1, I_2, I_3, I_4)]* adds the amount of product into the array. After firing transition *T1*, the markings of this combinatorial Petri net model are changed from Figure 7-6 to Figure 7-7. Here, we use dotted circle to represent a deleted token.

After firing *T1*, the first cycle in running this combinatorial Petri net model is finished. As *T1*'s *delay* is zero, so *modelTime* is unchanged, i.e. it is still zero.

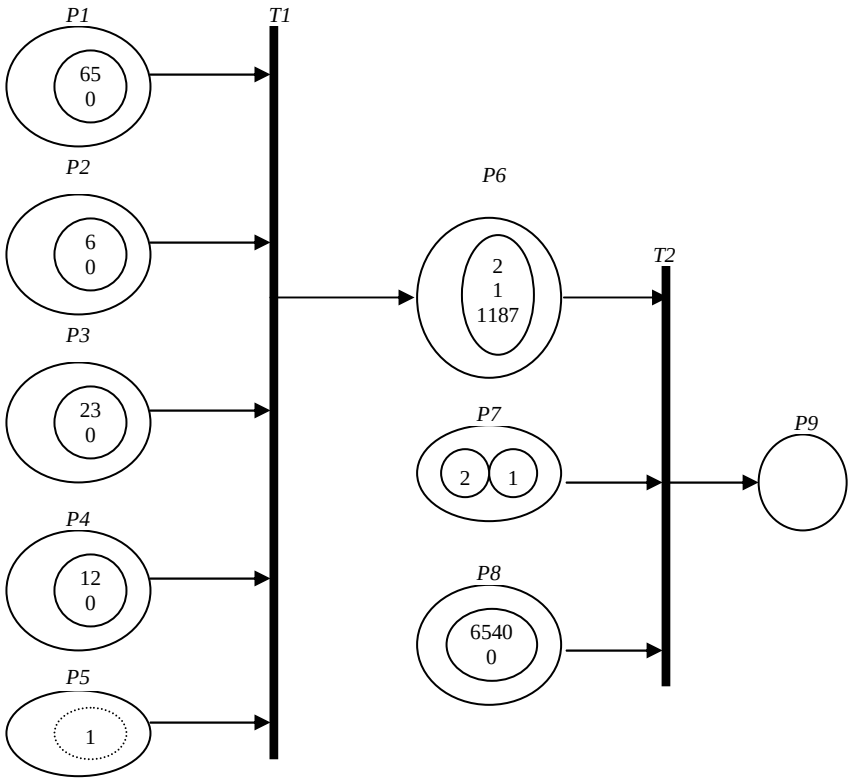


Figure 7-7 The markings of places after the first cycle

(4) Checking transitions by enabling rule

After firing transition *T1*, the first cycle is finished, and we come to the second cycle, i.e. check all transitions by enabling rule again, and fire the enabled one(s).

For transition *T1*, as place *P5* is empty, so it is not token-enabled, and it is not enabled too.

URN:NBN:no-3444

For transition *T2*, no E_1 arc expressions are attached to all its input arcs. This means at least one token is needed in each input place to make *T2* token-enabled. As shown in Figure 7-7, one token is located in place *P6* and *P8*, and two tokens in place *P7*, so *T2* is token-enabled. No premise is defined for transition *T2*, so it is also enabled.

(5) Firing the enabled transition(s)

In this cycle, only *T2* is enabled. As its *delay* is not zero, so we first execute its input E_2 arc expressions, update its *firingTime*, and then put it into event queue.

Transition *T2* represents the loading vehicle process at the wholesaler. After firing *T2*, the vehicle releasing decision is no longer useful, so we delete the token in input place *P6* by executing corresponding E_2 expressions. As the color of this token will be used in later programming, we store such information temporarily in a deleted token. After loading the vehicle, the available vehicles at the wholesaler is reduced by one, so a token is deleted from place *P7* by *P7.vehicle(1)=[];*, and its color is also stored in a deleted token by *P7.delVehicle(1)=P7.vehicle(1);*. For place *P8*, the *amount* of product is subtracted from the current inventory of the wholesaler by expression *P8.inventory(1)=P8.inventory(1)-P6.delAmount(1);*. Note here, we just reduce the wholesaler's inventory level, but the token is not deleted. After executing all input E_2 arc expressions, *T2*'s *firingTime* is updated to *modelTime+T2.delay=0+5=5*. Then, we put this transition into event queue.

In the event queue, there is only one transition: *T2*, so it is fired and its output E_2 arc expressions are executed. Place *P9* represents the loaded vehicle at the wholesaler. We use following structure to describe the color of tokens in this place:

```
structure loadedVehicle
                    {
                     int vehicleNumber;
                     float amount;
                    }
```

The first member indicates the vehicle number. Here, it is *1*. The second member indicates how much product is loaded on the vehicle. Here, it is *T6.delAmount(1)=1187;* After executing output E_2 arc expressions from transition *T2* to place *P9*, a token is added to place *P9*.

After firing *T2*, *modelTime* proceeds to *T2.firingTime=5*. Then, this cycle is finished, and the markings are changed from Figure 7-7 to Figure 7-8.

(6) Checking transitions by enabling rule

As shown in Figure 7-8, for both transitions, there is one empty input place, so they are not enabled, and the running process stops.

As Figure 7-2 only shows a part of a combinatorial Petri net model, so there is dead lock in running it. In practice, the loaded vehicle (the token in place *P9*) will transport to retailers, and the inventories of these retailers will be replenished. Then, because of customer demand, the retailers' inventory level will be reduced gradually. After all retailers are served, the vehicle comes back to the wholesaler, and a token is added to place *P5*. So *T1* will be enabled again to decide whether it is necessary to release another vehicle to serve the route. Then, this model will be alive again. This iterative running process proceeds until time out.

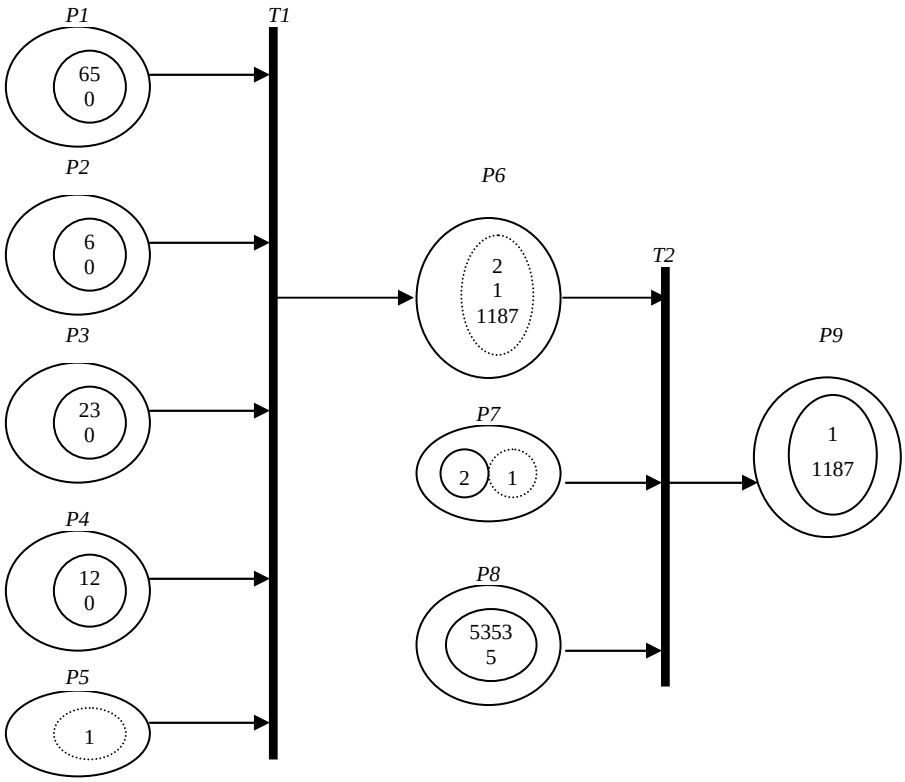


Figure 7-8 The markings of places after the second cycle

7.6 Modeling and Performance Evaluation of a Distribution Chain

In this section, we use the combinatorial Petri net developed above to model a simplified distribution chain, and then evaluate its performance by running this model.

7.6.1 Problem description

A general distribution chain model includes one distribution center, a set of wholesalers and retailers. Its configuration, inventory control parameters at each node and product delivery routes between different nodes have been determined in previous chapter. Given these results, the working process of this distribution chain can be modeled by combinatorial Petri net, and its performance can be estimated by running this model.

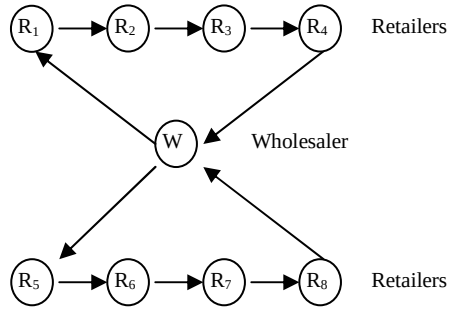


Figure 7-9 A simplified distribution chain with one wholesaler and eight retailers

We view a wholesaler and all retailers connected to it as a group. Obviously, the operation for different groups is similar to each other. To simplify the Petri net model, here, we only take one group as our modeling object. Figure 7-9 shows a simplified distribution chain with one wholesaler and eight retailers. According to routing result, all retailers are divided into two routes: $R_{1\text{-}4}$ and $R_{5\text{-}8}$, and each route is served by one vehicle. Parameters for this simplified distribution chain are shown in Table 7-4.

Table 7-4 Parameters for the distribution chain shown in Figure 7-9

Retailer	Demand (1000units)		Inter-arrival time (day)		Price ($/unit)	Position (km)	Unit inventory cost ($/1000unit/day)		Inventory control parameters (1000units)	
	mean	variance	mean	variance			shortage cost	carrying cost	reorder point s	ordering quantity Q
W						(0,0)	0	5		
R1	5	1	10	2	5	(30,240)	100	10	5	10
R2	6.5	1.4	13	3	6	(80,180)	100	10	5.6	11
R3	4.3	1	8	1	5	(50,130)	100	10	3	9
R4	6.8	1.6	4	1	5	(40,110)	100	10	6.5	11.5
R5	12	2	15	3	4	(10,-60)	100	10	12.5	17.5
R6	9.7	1.8	12	2	4	(40,-100)	100	10	10	13.5
R7	3.8	0.8	9	2	8	(30,-150)	100	10	3.5	8.5
R8	4.5	1	8	2	5	(40,-180)	100	10	4	9

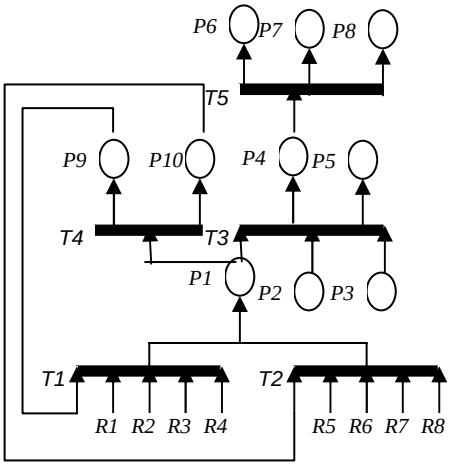
In this distribution chain, two types of costs are considered: inventory cost and product delivery cost. Inventory cost includes inventory carrying cost and inventory shortage cost. The product delivery cost is proportional to the traveling distance for vehicles, and the unit cost is assumed to be a constant. To reflect the reality, we assume that the traveling time between two nodes is a random variable with normal distribution, and the mean of traveling time is proportional to the distance between two nodes.

7.6.2 Combinatorial Petri net model

For the simplified distribution chain shown in Figure 7-9, there are mainly three types of operations: releasing vehicles at wholesaler and delivering product from wholesaler to retailers, delivering product between retailers, and selling product to customers and maintaining inventory at retailers. In this subsection, we will first illustrate the sub-models for these three types of operations, and then integrate them together to form the whole combinatorial Petri net model.

(1) Sub-model 1: Releasing vehicles at wholesaler and delivering product from wholesaler to retailers.

Figure 7-10 shows the combinatorial Petri net model for releasing vehicle at wholesaler and delivering product from wholesaler to a retailer (e.g. R_1), the corresponding notation explanation is shown in Table 7-5. In this model, there are five transitions, and no E_1 arc expression is defined for them. It means that, if none of the input places for a transition is empty, then it will be token enabled. Next, we will explain these transitions one by one.



Places and transitions:
R1-R8: inventory level for retailer*1-8*
P1: pool of routes that need to be served
P2: inventory level at the wholesaler
P3: available vehicles at the wholesaler
P4: loaded vehicle for route *1*
P5: loaded vehicle for route *2*
P6: transportation costs for different vehicles.
P7: run time for different vehicles
P8: arrival vehicle at the first retailer of route *1*
P9: place to indicate whether route *1* is being served
P10: place to indicate whether route *2* is being served
T1: inferring process for the vehicle releasing of route *1*
T2: inferring process for the vehicle releasing of route *2*
T3: loading and vehicle releasing process at the wholesaler
T4: return token to origin for the un-served routes
T5: transportation process from wholesaler to the first retailer of route*1*

E_2 arc expressions:
OA1-1: P1.VRI=[P1.VRI T1.abl], P1.route=[P1.route P9.delRoute(1)], P1.amount=[P1.amount SUM(I_1, I_2, I_3, I_4)];
OA2-1: P2.VRI=[P2.VRI T2.abl], P2.route=[P2.route P10.delRoute(1)], P2.amount=[P2.amount SUM(I_5, I_6, I_7, I_8)];
IA1-4: P1.delVRI=P1.VRI; P1.delRoute=P1.route; P1.delAmount=P1.amount; P1.VRI=[]; P1.route=[]; P1.amount=[];
OA4-9: if P1.delroute(1)==1 or P1.delroute(2)==1 P9.route=[1];
OA4-10: if P1.delroute(1)==2 or P1.delroute(2)==2 P10.route=[2];
IA1-3: I=MAX(P1.VRI), P1.delRoute(1)=P1.route(I), P1.route(I)=[ ], P1.delAmount(1)=P1.amount(I), P1.amount(I)=[ ], P1VRI(I)=[];.
IA2-3: P2.inventory(1)=P2.inventory(1)-P1.delAmount(1);
IA3-3: P3.delVehicle(1)=P3.vehicle(1), P3.vehicle(1)=[ ];
OA3-4: if P1.delRoute==1, P4.vehicle=[P4.vehicle P3.delVehicle(1)], P4.amount=[P4.amount P1.delAmount];
OA3-5: if P1.delRoute==2, P5.vehicle=[P5.vehicle P3.delVehicle(1)], P5.amount=[P5.amount P1.delAmount];
IA4-5: P4.delVehicle(1)=P4.vehicle(1), P4.delAmount(1)=P4.amount (1), P4.vehicle(1)=[ ], P4.amount(1)=[ ];
*OA5-6: P6.tranCost(P4.delVehicle(1))= P6.tranCost(P4.delVehicle(1)) + (CT(W, R_1)*P4.delAmount(1) + CF(W, R_1));*
OA5-7: P7.runTime(P4.delVehicle(1)) = P7.runTime(P4.delVehicle(1))+T5.delay;
OA5-8: P8.vehicle=[P8.vehicle P4.delVehicle(1)], P8.amount=[P8.amount P4.delAmount(1)];

Figure 7-10 Combinatorial Petri net model for releasing vehicle at wholesaler and product delivering from wholesaler to retailer R_1

Table 7-5 Explanation for notations used in arc expressions of Figure 7-10

Notation	Explanation
P1.VRI	An array of VRI for all routes to be served.
P1.route	An array of route numbers to be served
P1.amount	An array to indicate the amount of products to be loaded on vehicles when serving corresponding routes
I	The index of most urgent route to be served, it is found by function *MAX(P1.VRI)*
MAX	Function to find the index of most urgent route to be served.
P2.inventory	Inventory status at wholesaler
P3.vehicle	Queue of vehicles available at the wholesaler
P4.vehicle	The vehicle that has been loaded and is waiting for releasing to route *1*
P4.amount	Amount of product loaded on the vehicle which is waiting for releasing to route *1*
P5.vehicle	The vehicle that has been loaded and is waiting for releasing to route *2*
P5.amount	Amount of product loaded on the vehicle which is waiting for releasing to route *2*
P6.tranCost	Array of transportation costs for different vehicles
CT(W, R1)	Unit transportation cost from wholesaler to retailer R_1
CF(W, R1)	Fixed transportation cost from wholesaler to retailer R_1
P7.runTime	Array of run time for different vehicles
P8.vehicle	Queue vehicles arrived at retailer R_1
P8.amount	Amount of products loaded on the vehicles arrived at retailer R_1
P9.route	Token to indicate whether route *1* is being served
P10.route	Token to indicate whether route *2* is being served

Transition *T1* represents the decision-making process for releasing a vehicle to serve route *1*. *T2* represents the decision-making process for releasing a vehicle to serve route *2*. Both *T1* and *T2* are same as the transition *T1* in Figure 7-2. The working process for such kind of transition has been explained in previous section, so it is unnecessary to say more about them here.

Transition *T3* represents the vehicle loading process at wholesaler. If vehicle(s) is available at the wholesaler (i.e. *P3* is non-empty), and there is route(s) needing to be served (i.e. *P1* is non-empty), *T3* is enabled (as no inference rule is defined for it). When it is fired, the most urgent route is selected by arc expression *I=MAX(P1.index)* and served. If no vehicle is available (*P3* is empty) and there is queue of routes which need to be served (i.e. *P1* is non-empty), *T3* is un-enabled, but *T4* is enabled. When *T4* is fired, it adds a token to *P9* (if route *1* is not served) or *P10* (if route *2* is not served), then *T1* or *T2* is enabled again to re-evaluate the degree of urgency for corresponding route to be served.

Transition *T5* describes the product delivering process from wholesaler to retailer R_1 (the first retailer in route *1*). Here, no inferring rule is defined for it. If necessary, some rules can be defined to indicate the condition that a vehicle can or not begin its journey. When the loaded vehicle is available at place *P4*, *T5* is enabled. When it is fired, the loaded vehicle is added into place *P8* after a period of *T5.delay*, transportation cost and vehicle utilization time are calculated by corresponding arc expressions, and the results are put into place *P6* and *P7* respectively.

(2) Sub-model 2: Unloading product at a retailer and delivering product between retailers

Performance Evaluation for the Designed Distribution Chain

The combinatorial Petri net model for unloading product at retailer R_1 and delivering product from retailer R_1 to retailer R_2 is shown in Figure 7-11, and the corresponding notation explanation is shown in Table 7-6. In this model, there are four transitions, and no E_1 arc expression is defined for them. It means that, if none of the input places for a transition is empty, then the transition will be token enabled. In what follows, we will explain these transitions in detail.

Transition *T7* represents the product delivering process from wholesaler to retailer R_1. This transition has been explained above. Transition *T6* represents the volume rationing process at retailer R_1. When the vehicle is released at the wholesaler, the amount of product loaded on that vehicle was determined according to the inventory level of retailers at that time. Because of random demand at retailers, when the vehicle arrives at a retailer, inventory status of retailers may be unexpected, so we need to re-ration the limited amount of product loaded on this vehicle to retailers. A variance-related rule is used to ration the present amount of product loaded on the vehicle:

$$Q_1 = Ration(Q_v, I_1, I_2, I_3, I_4) = S_1 - I_1 + \frac{\sigma_1}{\sum_{i=1}^{4} \sigma_i}(Q_v - \sum_{i=1}^{4}(S_i - (I_i - L_{1i}))) \qquad (7-8)$$

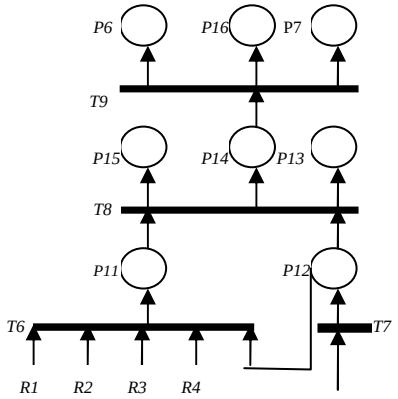
Where,

Q_1: amount of product to be unloaded at retailer R_1.
S_i: order up to level for retailer R_i
I_i: present inventory level at retailer R_i
σ_i: standard deviation of demand at retailer R_i.
Q_v: present volume of product on the vehicle.
L_{1i}: expected lead time demand when the vehicle travels from retailer R_1 to R_i.

When the loaded vehicle arrives at retailer R_1, *P12* is non-empty, so *T6* is enabled. When it is fired, function *Ration*(Q_v, I_1, I_2, I_3, I_4) is called to calculate how many products to be unloaded at retailer R_1, and the calculation result is put into place *P11*.

Transition *T8* represents the unloading process at retailer R_1. As no inference rule is defined for *T8*, once the vehicle arrives at retailer R_1 (*P12* is non-empty) and the volume of product on the vehicle is re-rationed (*P11* is non-empty), *T8* is enabled. When it is fired, unloaded amount of product is added to inventory of retailer *R1* (*P13*); inventory cost is calculated, and result is put into *P15*; and the token representing the unloaded vehicle is put into *P14*, which indicates that the vehicle is ready to continue its journey.

Transition *T9* represents the product delivering process from retailer R_1 to R_2. *Delay* for *T9* represents the transportation time between these two retailers. For other aspects, as they are similar to that of transition *T5* in Figure 7-10, we do not give further explanation here.



Places and transitions:
P6: transportation cost for vehicles.
P7: run time for vehicles.
P11: amount of product to be unloaded at retailer R_1.
P12: arrival vehicle at retailer R_1.
P13: inventory status at retailer R_1
P14: unloaded vehicle departing from retailer R_1
P15: inventory cost for retailers
P16: arrival vehicle (loaded) at retailer R_2
T6: rationing calculation.
T7: product delivering from wholesaler to retailer R_1
T8: unloading process at retailer R_1
T9: product delivering from retailer R_1 to retailer R_2.

E_2 arc expressions:
OA6-11: P11.amount(1)=Ration(P12.amount(1), I_1, I_2, I_3, I_4);
IA11-8: P11.delAmount=P11.amount(1), P11.amount(1)=[ ];
IA12-8: P12.delVehicle(1)=P12.vehicle(1), P12.delAmount(1)=P12.amount(1), P12.vehicle(1)=[ ], P12.amount(1)=[ ];
*OA8-15: If P13.delInventoryLevel(1)≥0 P15.inventoryCost(1)=P15.inventoryCost(1) + CI1*P13.delInventoryLevel(1)*(P13.updatedTime(1)- P13.delUpdatedTime(1)), If P13.delInventoryLevel(1)<0 P15.inventoryCost(1)=P15.inventoryCost(1) + CS1*P13.delInventoryLevel(1)*(P13.updatedTime(1)- P13.delUpdatedTime(1)),*
OA8-14: P14.vehicle=[P14.vehicle P12.delVehicle(1)], P14.amount=[P14.amount (P12.delAmount(1)- P11.delAmount(1))]
OA8-13: P13.delInventoryLevel(1)=P13.inventoryLevel(1), P13.inventoryLevel(1)=P13.inventoryLevel(1) + P11.delAmount, P13.delUpdatedTime=P13.updatedTime, P13.updatedTime=T;
IA14-9: P14.delVehicle(1)=P14.vehicle(1), P14.delAmount(1)=P14.amount(1), P14.vehicle(1)=[ ], P14.amount(1)=[ ];
*OA9-6: P6.tranCost(P14.delVehicle(1))= P6.tranCost(P14.delVehicle(1))+(CT(R1, R2)*P14.delAmount+CF(R1, R2))*
OA9-7: P7.runTime(P14.delVehicle(1))= P7.runTime(P14.delVehicle(1))+T9.delay;
OA9-16: P16.vehicle=[P16.vehicle P14.delVehicle(1)], P16.amount=[P16.amount P14.delAmount(1)]

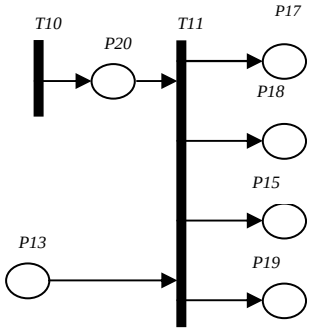
Figure 7-11 Combinatorial Petri net model for unloading product at a retailer and delivering product between retailers

Table 7-6 Explanation for notations used in arc expressions of Figure 7-11

Variable	Explanation
P11.amount	Amount of product to be unloaded at retailer R_1
P12.vehicle	Queue of vehicles arrived at retailer R_1
P12.amount	Amount of product loaded on vehicles when they arrive at retailer R_1
P13.inventoryLevel	Present inventory level at retailer R_1
P13.updatedTime	Time when R_1's inventory is updated
P14.vehicle	Queue of vehicles unloaded from retailer R_1.
P14.amount	Left amount of product loaded on the vehicle when it leaves retailer R_1.
P15.inventoryCost	Array of inventory carrying costs for all retailers
CI1	Unit inventory carrying cost at retailer R_1
CS1	Unit product shortage cost at retailer R_1
CT(R1, R2)	Unit product delivering cost between retailer R_1 and R_2
CF(R1, R2)	Fixed transportation cost between retailer R_1 and R_2
T9.delay	*delay* for transition *T9*, it is a random variable

(3) Sub-model 3: selling product to customers and carrying inventory at retailers.

Figure 7-12 shows the combinatorial Petri net model for selling product to customers and carrying inventory at a retailer (e.g. retailer *R1*). All notations used in this figure are explained in Table 7-7. In this model, there are two transitions, and no E_1 arc expression is defined for them. It means that, if none of the input places for a transition is empty, then the transition is token enabled. In what follows, we will explain these transitions in detail.



Places and transitions:
P13: inventory level at retailer R_1
P15: inventory costs at retailers
P17: global revenue
P18: revenues at retailers
P19: fill rates at retailers
P20: customer demand at retailer R_1
T10: generate customer demands
T11: sell products to customers

E_2 arc expressions:

*OA10-20: D1=uD1+dD1*Rand(n1); n1=n1+1; P20.demand=[P20.demand D1];*
IA20-11: P20.delDemand(1)=P20.demand(1); P20.demand(1)=[ ];
IA13-11: P13.delInventoryLevel(1)=P13.inventoryLevel(1); P13.inventoryLevel(1)=P13.inventoryLevel(1) –P20.delDemand; P13.delUpdatedTime=P13.updatedTime, P13.updatedTime=T;
OA11-17: If P13.delInventoryLevel(1)>P20.delDemand(1)
P17.gloRevenue(1)= P17.gloRevenue(1)+PR1 P20.delDemand(1);*
If 0< P13.delInventoryLevel (1)< P20.delDemand(1)
P17.gloRevenue(1)= P17.gloRevenue(1) + PR1 P13.delInventoryLevel(1);*
OA11-18: If P13.delInventoryLevel(1)>P20.delDemand(1)
P18.revenues(1) = P18.revenues(1) + PR1 P20.delDemand(1);*
If 0< P13.delInventoryLevel (1)< P20.delDemand(1)
P18.revenues(1) = P18.revenues(1) + PR1 P13.delInventoryLevel(1);*
*OA11-15: If P13.delInventoryLevel(1)≥0 P15.inventoryCost(1)=P15.inventoryCost(1) + CI1*P13.delInventoryLevel(1)*(P13.updatedTime(1)- P13.delUpdatedTime(1)),*
*If P13.delInventoryLevel(1)<0 P15.inventoryCost(1)=P15.inventoryCost(1) + CS1*P13.delInventoryLevel(1)*(P13.updatedTime(1)- P13.delUpdatedTime(1)),*
OA11-19: P19.sumOfDemand(1)=P19.sumOfDemand(1)+ P20.delDemand(1);
If 0< P13.delInventoryLevel (1)< P20.delDemand(1)
P19.short(1)=P19.short(1)+P20.delDemand(1)- P13.delInventoryLevel (1);
If P13.delInventoryLevel (1)<0
P19.short(1)=P19.short(1)+ P20.delDemand(1);

Figure 7-12 Combinatorial Petri net model for selling products to customers and carrying inventory at retailer R_1

Transition *T10* represents the action of generating customer demands. No input place is defined for it. This means that the transition is always enabled. When it is fired, one demand (a random value) is generated and put into place *P20*. *Delay* for *T10* is also a random variable. It represents the inter-arrival time of successive demands.

Transition *T11* represents the action of selling products to customers. When there are demand (*P20* is non-empty) and product at retailer R_1 (*P13* is non-empty), *T11* is enabled (as no inference rule is defined for this transition). When it is fired, inventory

level at retailer *R1* (*P13)* is updated, global revenue (*P17*), local revenue (*P18*), inventory cost for R_1 (*P15*), and fill rate at *R1 (P19*) are calculated by corresponding arc expressions.

Table 7-7 Explanation for notations used in arc expressions of Figure 7-12

Notation	Explanation
D1	Demand at retailer 1, it is a random variable
uD1	Mean of variable D1
dD1	Standard deviation of variable D1
Rand	An array of random number with normal distribution (mean is 0, and standard deviation is 1)
P20.demand	Queue of demands generated by transition *T10*
P17.gloRevenue	Sum of revenues for this distribution chain
PR1	Product price at retailer R_1
P18.revenues	Array of revenues for different retailers
P19.sumOfDemand	An array of total demands at different retailers
P19.short	Array of total product shortages (in quantity) for different retailers

The sub-models for main operations in a distribution chain have been illustrated above. Now, we integrate them together to form the whole combinatorial Petri net model (as shown in Figure 7-13) for the simplified distribution chain shown in Figure 7-9. In Figure 7-13, the shaded boxes represent retailers R_1~R_8. For brevity, the actions inside a retailer (e.g. generating demands, etc.) are not shown here. Place *P55* represents the available vehicles at the wholesaler. Once a vehicle finishes its journey in serving a route, it comes back to this place. Places *P51* and *P52* indicates whether the corresponding routes (route *1* and route *2*) are being served. For example, when route *1* is being served by a vehicle, *P51* is empty; when it is not served for the moment, *P51* is not empty, so *T25* is enabled to assess whether route *1* needs to be served. Other places and transitions have been explained previously, for brevity, we do not mention them again here.

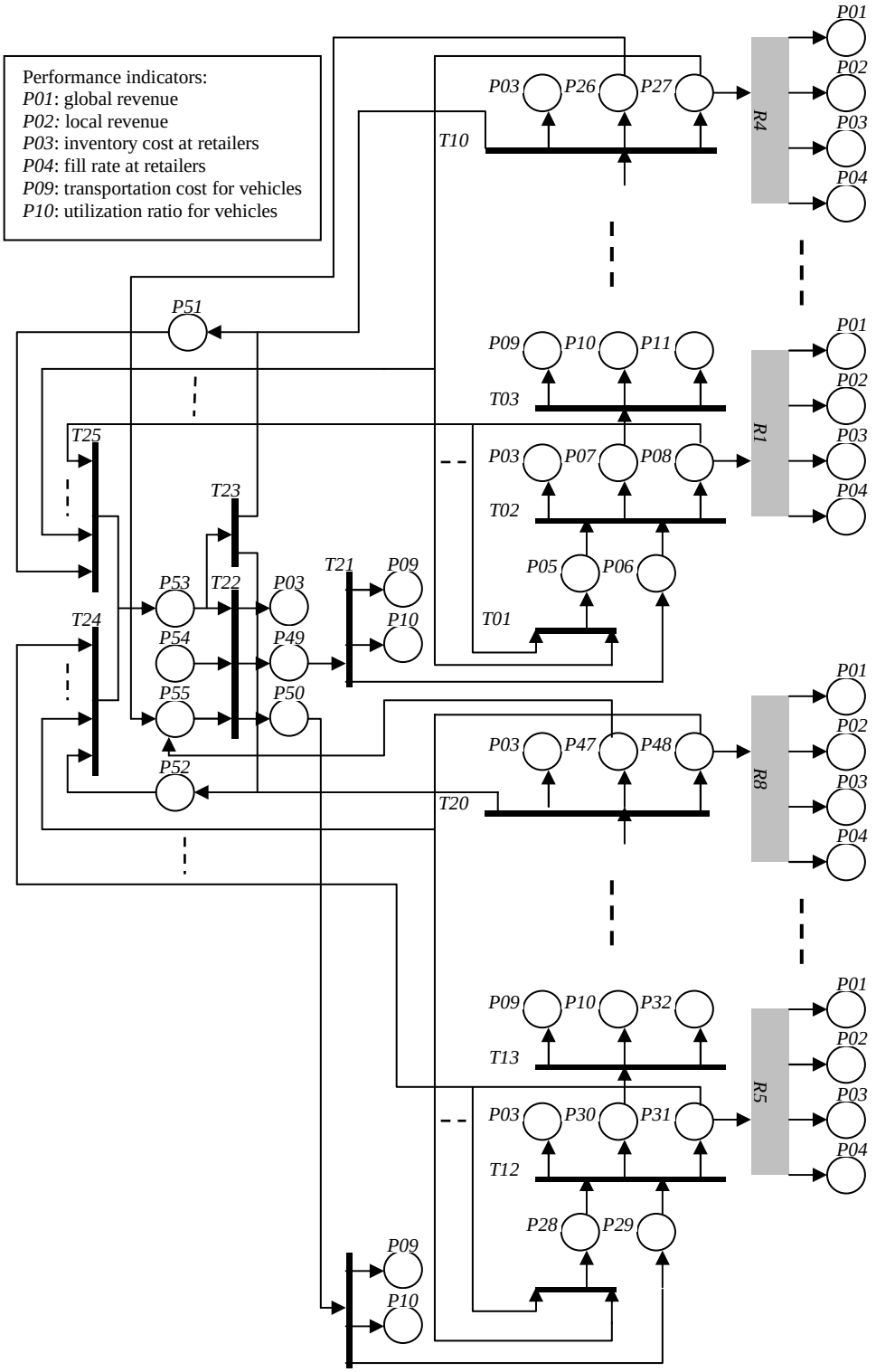


Figure 7-13 Combinatorial Petri net model for the simplified distribution chain shown in Figure 7-9

7.6.3 Realization of the combinatorial Petri net model and performance evaluation of the distribution chain

In the previous sub-section, to evaluate the performance of a distribution chain, we established its combinatorial Petri net model. According to this model, all the performance measures are calculated during the running process. In this sub-section, computer application will be used to run this model, simulate the working process of a distribution chain, and evaluate its performance. For the following two reasons, MATLAB is selected to develop this computer application:

- MATLAB can provide abundant functions in operating matrix. As shown above, most parameters in this Petri net model are expressed by matrix, so it is efficient to develop our simulation tool in MATLAB environment.
- In this dissertation, almost all of the computer tools are developed in MATLAB environment. If this simulation tool is also developed in it, it will be convenient to integrate all these tools together to form a systematic one.

By the tool developed in MATLAB, a combinatorial Petri net model is realized according to following steps:

Step 1. Form the combinatorial Petri net model

(1) Initialize the Petri net model by function *initialization*
(2) Define places by function *node*, e.g.
P(1)=node('place', 1, iniMarking, PN);
Here the third argument is the initial marking for place *P(1)*.
(3) Define transitions. A transition can also be defined by function *node*, e.g.
T(3)=node('transition', 3, initParameters, PN);
Here the third argument is the initial parameters for transition *T(3)*.
(4) Form function *ABL* for transitions based on given inference rules, e.g.
$T(3).abl=ABL(3, M(p_i: p_i \in I(T(3))));$
(5) Store arc expressions in function *arc_expressions*.

Step 2. Check the enabled transitions according to *enabling rule*, and put the enabled ones into *eventQueue*. This is accomplished by function: *enabling*.
Step 3. Fire transitions in *eventQueue* according to *firing rule*. This is accomplished by function *firing*.
Step 4. Check the ending condition, and decide to begin a new circle, or end the program (as shown in Figure 7-5).

The main functions used above are shown in Appendix C.

Once the distribution chain is changed, we only need to change some parameters, or re-define the related places, transitions and arcs, but keep other parts unchanged. Obviously, the re-programming load is reasonable.

After running this combinatorial Petri net model, we get following results: the total profit for this distribution chain is 24212 $/day (global revenue is 25920 $/day, and global cost is 1708 $/day). The distribution of other performance measures is shown in Figure 7-14. By these results, we can verify whether this distribution chain design is satisfactory.

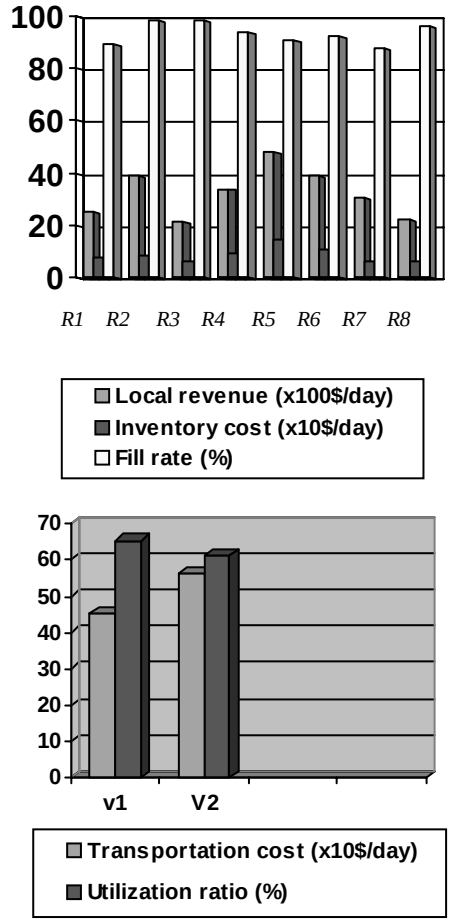


Figure 7-14. Distribution of other performance measures for 8 retailers (*R1-R8*) and two vehicles (*V1-V2*)

7.7 Summary

In this chapter, to verify the design result, a combinatorial Petri net model is developed, and then it is used to evaluate the performance of a designed distribution chain. Normally, traditional Petri net is good at simulating the working process of a discrete event system, but poor at simulating the inferring process. To supplement this weak point, ABL is combined with the traditional Petri net to form the combinatorial Petri net. This newly developed Petri net form can simulate both working and inferring process of a discrete event system. With this new form of Petri net, a simplified distribution chain is modeled, and its performance is evaluated by running this model. By this evaluation result, the decision makers can decide whether this designed distribution chain is applicable in practice. This Petri net form has been realized by computer applications in MATLAB environment.

For this newly developed Petri net form, in the future, we need to do more research work in following directions:

- Here, only some basic definition and rules for combinatorial Petri net are given. In the future, we need to complete the theoretical part of this new Petri net form.
- By combinatorial Petri net, the modeling process is boring. In the future, we will try to simplify this new Petri net form, and develop more basic blocks or sub-models that are re-usable in modeling large scale systems.
- At the same time, we will also complete the computer application to make it more convenient when used in practice.

CHAPTER 8 A NUMERICAL EXAMPLE FOR THE DESIGN OF DISTRIBUTION CHAIN

8.1 Introduction

In chapter 3, the structure of the integrated methodology for distribution chain design is illustrated. To realize this methodology, a set of models, formulae and algorithms are developed in the subsequent chapters. As they are introduced individually in those chapters, the procedure on how to use them in designing a distribution chain is not clear. To solve this problem, we use Figure 8-1 to show the flow chart on how to apply these models in practice. This flow chart accords with the structure of this design methodology shown in Figure 3-2. In this chapter, a numerical example will be given to illustrate the procedure on designing a distribution chain by the methodology developed in this dissertation.

The distribution chain to be designed is described as follows: an enterprise wants to design a divergent multi-echelon distribution chain with one distribution center, one tier of wholesalers and a set of retailers. The enterprise plans to select retailers from potential ones, and build wholesalers and distribution center to form this distribution center. The task for this design is to determine the configuration of the distribution chain, inventory control policy and parameters at each node, and product delivering routes between different nodes. The objective to design this distribution chain is to maximize profit for the host enterprise subject to satisfying customer requirements.

In this chapter, the distribution chain is designed according to the flowchart shown in Figure 8-1. In section 8.2, after selecting a set of retailer and wholesaler candidates, the information about them is given. The distribution center is located closely to the host enterprise. This process can also be viewed as pre-design. In section 8.3, based on the candidates identified above, the configuration of distribution chain, inventory control policy and parameters at each node of this distribution chain, and routes for delivering products between different nodes are determined. In section 8.4, these design results are verified by estimating the performance of this designed distribution chain.

8.2 Pre-design of the Distribution Chain

According to Figure 8-1, after analysing the enterprise's situation and setting design objectives, we come to the first stage of designing a distribution chain: pre-design. There are mainly two purposes for setting this pre-design stage: selecting the eligible distributors and reducing the scale of the problem. When an enterprise begins to design its distribution chain, it faces a lot of possible distributors. If these distributors are directly input into the design model, the resulted model may be too large to be solved. To avoid this problem, we set this pre-design process to evaluate all possible distributors comprehensively, and select the eligible ones to form the design model. Moreover, the evaluation process is accomplished by an ABL-FL inference module. This inference module can take qualitative even logic variables into account, which are difficult to be considered in the later design stage. Obviously, by setting this pre-design

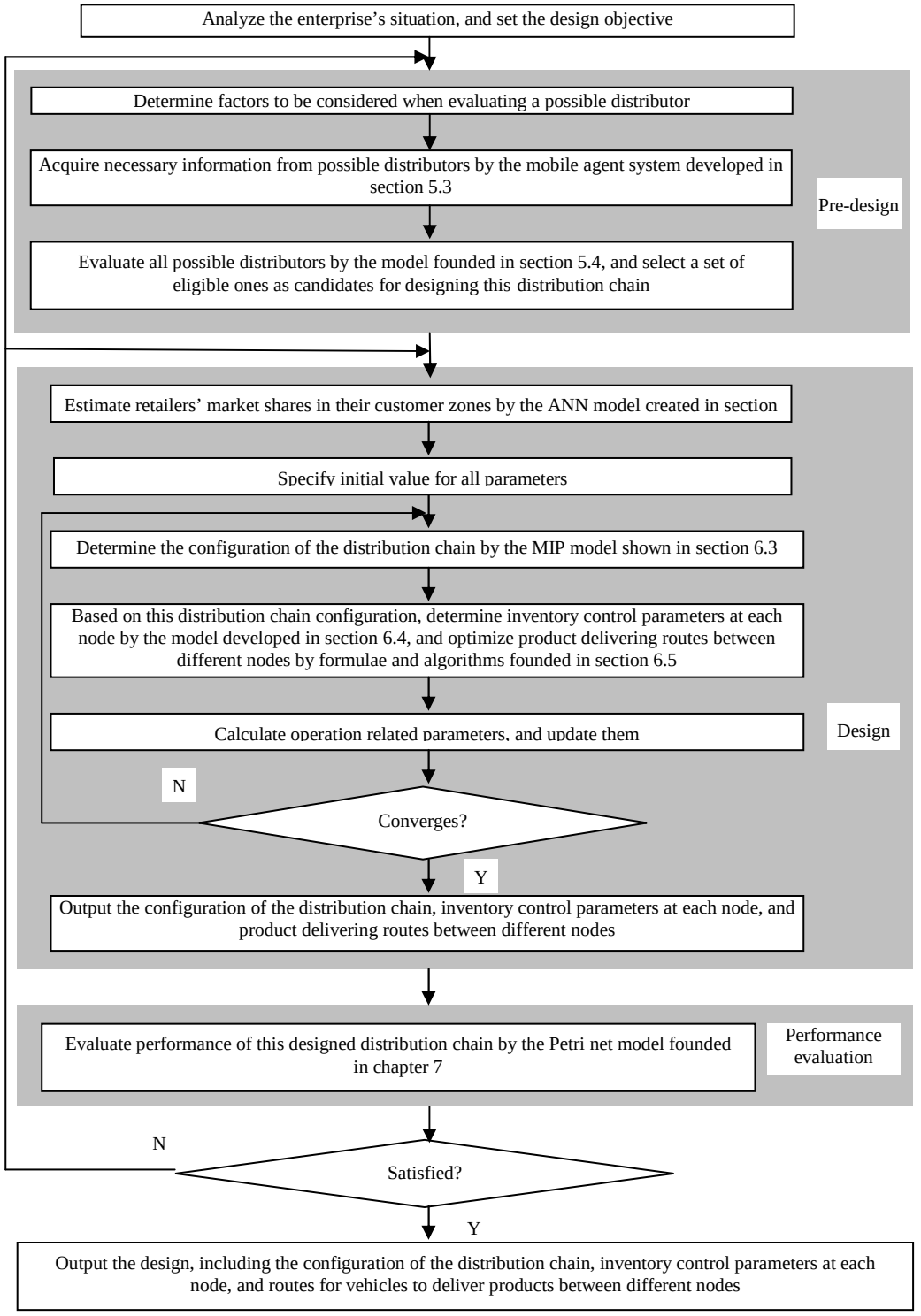


Figure 8-1 Flow chart on how to use the methodology developed in this dissertation in designing a distribution chain

stage, we can evaluate possible distributors more comprehensively, select the really eligible distributors, and improve the design quality. This pre-design is finished according to following procedure:

- Form the variable table based on the evaluating factors shown in Figure 5-3.
- According to this variable table, acquire information from all possible distributors by any kind of approaches, including the mobile agent approach created in section 5.3.
- Evaluate all possible distributors by the ABL-FL inference module founded in section 5.4, and select the eligible ones by subjective judgement.

In subsection 5.4.5, the procedure on how to evaluate a distributor is illustrated. This procedure is based on the model shown in Figure 5-3. Then, in subsection 5.4.6, a case study is given to show how to evaluate a distributor in reality. By this procedure, all possible distributors can be evaluated, and the evaluation results are indicated by numbers ranging from *0* to *100.* Based on these results, the decision makers can select the eligible distributors by subjective judgement.

As the evaluation processes for any other distributors are same as the one in the case study given in subsection 5.4.6, here, for brevity, we do not give the detail evaluation processes for the possible distributors, but just show the evaluation and selection result. Assume that we have finished the evaluation process, and a set of distributors have been selected as retailer candidates. The parameters for these selected retailers are shown as Table 8-1.

Table 8-1 Parameters for selected retailers

Retailers R1-R10	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Total demand at its customer zone (unit/day)	960	1330	1270	1660	1590	2940	2000	2100	3550	2610
Position (coordinate) (km)	[-30, 20]	[0, 30]	[10, 40]	[10, 20]	[20, 30]	[30, 10]	[35, 30]	[45, 10]	[58, 49]	[68, -75]
Retailers R11-R20	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20
Total demand at its customer zone (unit/day)	990	1550	1500	2740	2540	1350	3580	1500	1350	1800
Position (coordinate) (km)	[-30, -10]	[-20, -20]	[-10, -10]	[0, -20]	[15, -10]	[20, -40]	[35, -10]	[45, -10]	[45, -20]	[30, -30]

Based on these retailer candidates, three locations are identified as the possible places to build wholesalers. The parameters for them are shown in Table 8-2. Based on these retailer and wholesaler candidates, we can begin the design process.

Table 8-2 Possible locations to build wholesalers

Locations L1~L3	L1	L2	L3
Position (coordinate) (km)	[30, 30]	[20, -10]	[-30, -30]

8.3 Design of the Distribution Chain

8.3.1 Determine configuration of the distribution chain

In Table 8-1, the total demands at customer zones are given. Normally, there may be several firms in a customer zone that sell the same product as the one to be distributed. To determine the actual demand at a selected retailer, we need to estimate its market share in its customer zone. As mentioned in section 6.2, a firm's market share in a customer zone is related to nine factors, and it can be estimated by the ANN model founded there. This estimation is finished according to following procedure:

- Train the ANN model by historical samples.
- Determine values of factors for a retailer.
- Input these values into the trained ANN model, then the resulted market share in the customer zone is estimated.

As there is no real case here, and so there are no historical samples to train the ANN model. To continue the design process, we assume that the estimation process has been finished, and the results are shown in Table 8-3. As this numerical example is mainly used to illustrate the design process, such assumption is allowable. The product price at each retailer is also shown in Table 8-3.

Table 8-3 Market share, actual demand and price at each customer zone

Retailers R1-R10	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Market share (%)	35	32	30	34	38	26	24	25	20	24
Customer demand at the retailer (unit/day)	340	430	380	570	610	770	480	530	710	630
Price ($/unit)	4	4.5	5	4.5	4	6	5.5	5.5	6	5.5
Retailers R11-R20	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20
Market share (%)	35	30	28	21	26	29	20	28	30	27
Customer demand at the retailer (unit/day)	350	470	420	580	660	390	720	420	410	490
Price ($/unit)	4.8	5.5	4.8	6.3	5.4	4.9	6.1	5.6	4.9	5.4

After giving market shares for retailers and determining customer demands at them, the configuration of the distribution chain can be determined by the MIP model developed in section 6.3. The initial values for parameters at possible retailers, wholesalers and distribution center are shown in Table 8-4, Table 8-5, and Table 8-6 respectively. Inputting these initial values into the program shown in Appendix D, we can obtain values for variables u_i (i=1-20), v_j (j=1-3), and w_{ij}. u_i indicate the selection /rejection of possible retailers (*1* indicates selection, and *0* indicates rejection), v_i indicate the selection /rejection of possible wholesalers, and w_{ij} indicate the assignments of retailers to wholesalers. After solving this model, we get following results: among 20 possible retailers, 14 of them are finally selected. As for the wholesalers, 2 of the 3 possible locations are selected. The detail of these selections is shown in Table 8-7. Besides the selections of possible retailers and wholesalers, the program also outputs values for w_{ij} to indicate the assignments of retailers to wholesalers. So the configuration of this distribution chain is determined, and shown in Figure 8-2.

Table 8-4 Initial values for parameters at possible retailers

Retailers R1-R10	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Unit product delivery cost from W1 ($/unit)	0.6	0.31	0.22	0.22	0.1	0.2	0.05	0.25	0.34	1.12
Unit product delivery cost from W2 ($/unit)	0.78	0.67	0.72	0.54	0.61	0.40	0.60	0.42	0.84	0.59
Unit product delivery cost from W3 ($/unit)	0.5	0.67	0.82	0.64	0.78	[0.72	0.88	0.85	1.18	1.08
Retailers R11-R20	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20
Unit product delivery cost from W1 ($/unit)	0.72	0.71	0.57	0.58	0.43	1.15	0.40	0.43	0.52	0.60
Unit product delivery cost from W2 ($/unit)	0.62	0.51	0.45	0.32	0.25	0.19	0.21	0.25	0.19	0.10
Unit product delivery cost from W3 ($/unit)	0.20	0.14	0.29	0.32	0.49	0.95	0.68	0.79	0.77	0.60

Table 8-5 Initial values for parameters at possible wholesalers

Wholesalers W1~W3	W1	W2	W3
Average inventory level I_j (1000 units)	380	420	460
Highest inventory level I_{hj} (1000 units)	95.2	99.6	114.8
Opening cost $C_{openj}(I_{hj})$ (allocated for one planning period) ($)	50000	50000	60000
Unit inventory maintaining cost C_j ($/unit*day)	0.01	0.01	0.01
Maximal throughput C_{tmax} (1000 units)	1000	1100	1300
Minimal throughput C_{tmin} (1000 units)	500	500	500
Required delivery flexibility ε_{lj} (%)	20	20	20
Position (km)	[30, 30]	[20, -10]	[-30, -30]

Table 8-6 Initial values for parameters at the distribution center and host enterprise

Planning period T (day)	200	Required production volume flexibility ε_p (%)	20
Maximal production capacity C_{pmax} (1000 units)	1920	Average inventory level I_0 (1000 units)	500
Minimal production capacity C_{pmin} (1000 units)	1000	Inventory maintaining cost C_0 ($/unit*day)	0.005

Table 8-7 The selected retailers and wholesalers (The grey ones are not selected)

Retailers	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20
wholesalers	W1, W2, W3

8.3.2 Determine inventory control parameters at each node

The inventory related parameters for the nodes in Figure 8-2 are shown in Table 8-8 and Table 8-9. Based on these given parameters, the model created in section 6.4 can be used to determine the optimal parameter pair (*s*, *Q*) (*s* is the re-ordering

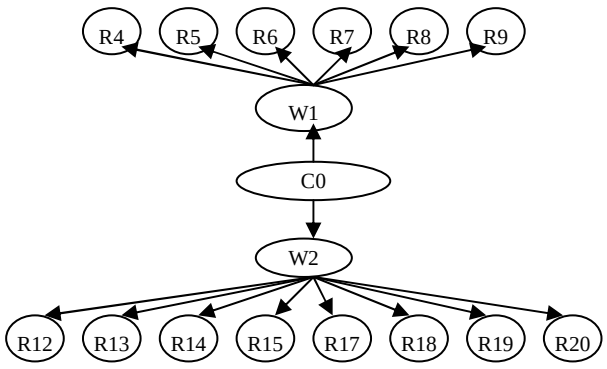


Figure 8-2 Configuration of the distribution chain

point, Q is the ordering quantity) and corresponding inventory maintaining cost for retailers and wholesalers in Figure 8-2. This process is accomplished by the program shown in B-1 of Appendix B, and the corresponding results are shown in Table 8-10.

Table 8-8 inventory related parameters for retailers

Nodes		R4	R5	R6	R7	R8	R9	R12
Demand (unit)	Mean	570	610	770	480	530	710	460
	Standard deviation	0.1	0.1	0.15	0.1	0.1	0.15	0.1
Interarrival (day)	Mean	1	1	1	1	1	1	1
	Standard deviation	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Cost coefficient	Holding cost ($/1000units*day)	10	10	10	10	10	10	10
	Shortage cost ($/1000units*day)	100	100	100	100	100	100	100
	Lead time (day)	10	8	9	7	10	14	10
	Ordering cost ($)	1000	1000	1000	1000	1000	1000	1000
Nodes		R13	R14	R15	R17	R18	R19	R20
Demand (unit)	Mean	420	580	660	720	420	410	490
	Standard deviation	0.1	0.1	0.1	0.15	0.1	0.1	0.1
Interarrival (day)	Mean	1	1	1	1	1	1	1
	Standard deviation	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Cost coefficient	Holding cost ($/1000units*day)	10	10	10	10	10	10	10
	Shortage cost ($/1000units*day)	100	100	100	100	100	100	100
	Lead time (day)	9	8	6	8	9	10	7
	Ordering cost ($)	1000	1000	1000	1000	1000	1000	1000

Table 8-9 Inventory related parameters for wholesalers

		W1	W2
Demand (1000 units)	Mean	10.5	9.5
	Standard deviation	2	1.8
Interarrival (day)	Mean	3	2.2
	Standard deviation	0.5	0.4
Cost coefficient	Holding cost ($/1000units*day)	10	10
	Shortage cost ($/1000units*day)	100	100
	Lead time (day)	15	15
	Ordering cost ($)	10000	10000

Table 8-10 Optimal parameter pairs (s, Q) and corresponding cost for retailers and wholesalers

Nodes	R4	R5	R6	R7	R8	R9	R12	R13
Re-ordering point s (1000 units)	5	5	8	3	5	10	4	3
Ordering quantity Q (1000 units)	11	10	12	10	9	11	9	9
Inventory maintaining cost ($/day)	106	109	125	95.2	102.3	122.7	95.6	90.5
Nodes	R14	R15	R17	R18	R19	R20	W1	W2
Re-ordering point s (1000 units)	4	4	6	4	4	5	60	70
Ordering quantity Q (1000 units)	10	11	12	8	8	9	80	90
Inventory maintaining cost ($/day)	106.8	112	118.9	90.9	89.7	97.95	878.9	966.5

8.3.3 Plan product delivery routes between different nodes

After determining inventory control parameters for retailers, the genetic algorithm model provided in section 6.5 can be used to optimize product delivering routes from wholesalers to retailers. Before applying this model, we need to give some route related parameters. In addition to the inventory related parameters given above, we give the route related parameters which are in Table 8-11 and Table 8-12. Both will be used in the genetic algorithm model.

Table 8-11 Route related parameters for retailers

Retailers		R4	R5	R6	R7	R8	R9	R12
Time related cost	Early arrival cost coefficient α ($/1000units)	10	10	10	10	10	10	10
	Late arrival cost coefficient β ($/1000units)	5	5	5	5	5	5	5
	Stock out cost coefficient γ ($/1000units)	100	100	100	100	100	100	100
	Ideal refill point (1000 units)	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Retailers		R13	R14	R15	R17	R18	R19	R20
Time related cost	Early arrival cost coefficient α ($/1000units)	10	10	10	10	10	10	10
	Late arrival cost coefficient β ($/1000units)	5	5	5	5	5	5	5
	Stock out cost coefficient γ ($/1000units)	100	100	100	100	100	100	100
	Ideal refill point (1000 units)	0.5	0.5	0.5	0.5	0.5	0.5	0.5

Table 8-12 Other route related parameters

Distance related cost coefficient ($/km)	Vehicle renting cost coefficient ($/vehicle)	Vehicle capacity (1000units)
5	2000	350

Inputting these parameters into the program shown in B-2 of Appendix B, the product delivering routes from wholesalers to retailers are optimized. As there are only several retailers in a route, the optimization process converges in a few generations, and the optimized routes for this distribution chain are (1) *W1-R7-R5-R4-W1*; (2) *W1-R6-R8-R9-W1*; (3) *W2-R15-R17-R18-R19-W2*; (4) *W2-R20-R14-R13-R12-W2* (as shown in Figure 8-3). Actually, for such small scale problem, the advantage of routing algorithm provided in this dissertation is not significant. When there are dozens, even hundreds of retailers in a route, this optimization approach will be very helpful in determining routes for vehicles.

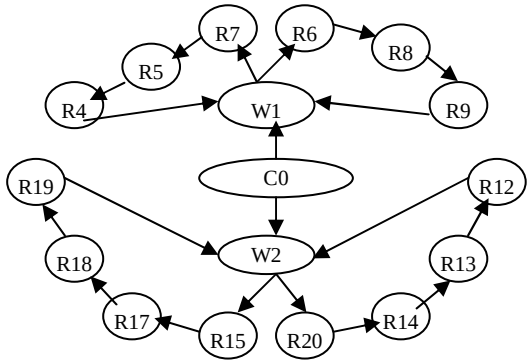


Figure 8-3 Routes in the distribution chain

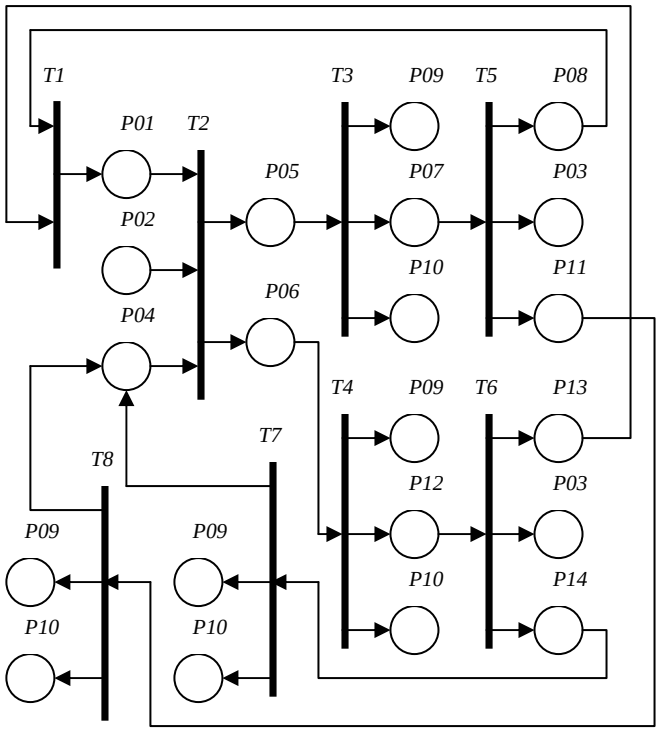
According to the iterative design process shown in Figure 8-1, after determining the inventory control parameters for each retailer, the average inventory levels at them are re-calculated and updated. After planning the product delivery routes from wholesalers to retailers, the delivery cost per unit product for each retailer is also re-calculated and updated. These updated parameters are input into the program shown in Appendix D, and the configuration of the distribution chain is re-determined. The re-determined configuration is same as the one shown in Figure 8-2, this means that the design process converges, and the results shown above can be viewed as the final design.

So far, we have finished the design of this distribution chain. The configuration of the distribution chain is shown in Figure 8-2. Inventory control parameters at each node of this distribution chain are shown in Table 8-10. Product delivery routes between different nodes are shown in Figure 8-3. After designing a distribution chain, we can evaluate its performance by "running" it. In next section, we will use combinatorial Petri net to simulate the running process of this distribution chain, and then evaluating its performance to verify our design.

8.4 Performance evaluation for the designed distribution chain

Based on the distribution chain shown in Figure 8-2 and routes sketched in Figure 8-3, with all necessary parameters (including inventory maintaining parameters) at hand, the combinatorial Petri net model developed in chapter 7 can be used to evaluate performance of this designed distribution chain.

The combinatorial Petri net model from distribution centre to wholesalers *W1* and *W2* is shown in Figure 8-4. This model is similar to the model shown in Figure 7-10. Now, we simply introduce its working process.



Places:
P01: pool of wholesalers that need to be served.
P02: inventory level at the distribution centre
P03: inventory carrying cost at wholesalers
P04: available vehicles at the distribution centre
P05: loaded vehicle for wholesaler *W1*
P06: loaded vehicle for wholesaler *W2*
P07: arrival vehicle at wholesaler *W1*
P08: inventory level at wholesaler *W1*
P09: transportation costs for vehicles
P10: utilization ratios for vehicles
P11: unloaded vehicle at wholesaler *W1*
P12: arrival vehicle at wholesaler *W2*
P13: inventory level at wholesaler *W2*
P14: unloaded vehicle at wholesaler *W2*

Transitions:
T1: decision making process for releasing vehicles to wholesalers
T2: loading and vehicle releasing process at the distribution centre
T3: transportation process from distribution centre to wholesaler *W1*
T4: transportation process from distribution centre to wholesaler *W2*
T5: unloading process at wholesaler *W1*
T6: unloading process at wholesaler *W2*
T7: transportation process from wholesaler *W2* back to distribution centre
T8: transportation process from wholesaler *W1* back to distribution centre

Figure 8-4 The combinatorial model from distribution centre to wholesalers *W1* and *W2*

First, according to the present inventory status of wholesaler *W1* and *W2*, the decision on whether wholesaler(s) need to be served is made by transition *T1*, and the decision is put into place *P01*. If wholesaler *W1* needs to be served, a vehicle at the distribution

centre is loaded (the loading process is represented by transition *T2)*, and the loaded vehicle (a token) is put into place *P05*. This loaded vehicle travels to wholesaler *W1* (the travelling process is presented by transition *T3*), and a token is added to place *P07* to represent the arrival vehicle at wholesaler *W1*. This arrival vehicle is unloaded at wholesaler *W1* (the unloading process is represented by *T5*), and the inventory level at wholesaler W1 (place *P08*) is updated. The working process for wholesaler *W2* is same as the one for *W1*. For brevity, we do not mention it here.

The combinatorial Petri net model from wholesaler *W1* to its retailers is shown in Figure 8-5. It is almost same as the model shown in Figure 7-13. In this model, there are two routes: route *W1-R7-R5-R4-W1* and route *W1-R6-R8-R9-W1*. First the decision on whether the route(s) need to be served is made at transition *T09* and *T10*. This decision is put into place *P45*. Based on this decision, vehicles are released from wholesaler *W1* and begin their journey to serve the corresponding routes. After serving routes, these vehicles come back to wholesaler *W1*, and corresponding tokens are put into place *P47*. As this working process has been explained in Chapter 7, we will not explain it in detail here.

The combinatorial Petri net model from wholesaler *W2* to its retailers is almost same as the one shown in Figure 8-5. For brevity, we do not show the detail model here.

Integrating these three models together, we get the combinatorial Petri net model for the distribution chain shown in Figure 8-2. This model can be realized by the functions shown in Appendix C. By running this Petri net model, the main performance measures for this distribution chain is shown in Table 8-13, and the distribution of local avenues at retailers, and costs for different activities are shown in Figure 8-6, Figure 8-7, and Figure 8-8. By these results, the decision makers can decide whether this designed distribution chain is good enough to be implemented.

Table 8-13 Main performance measures for this distribution chain

Total revenue (1000$)	Sum of inventory maintaining cost (1000$)	Sum of transportation cost (1000$)	Total profit (1000$)
8358.9	604.6	24.1	7730.2

By Table 8-13, we find that, as one vehicle only serves a route, the utilization ratios for vehicles are very low. In the future, the situation that one vehicle serves several routes needs to be considered. Because of limited time, we leave it as future work.

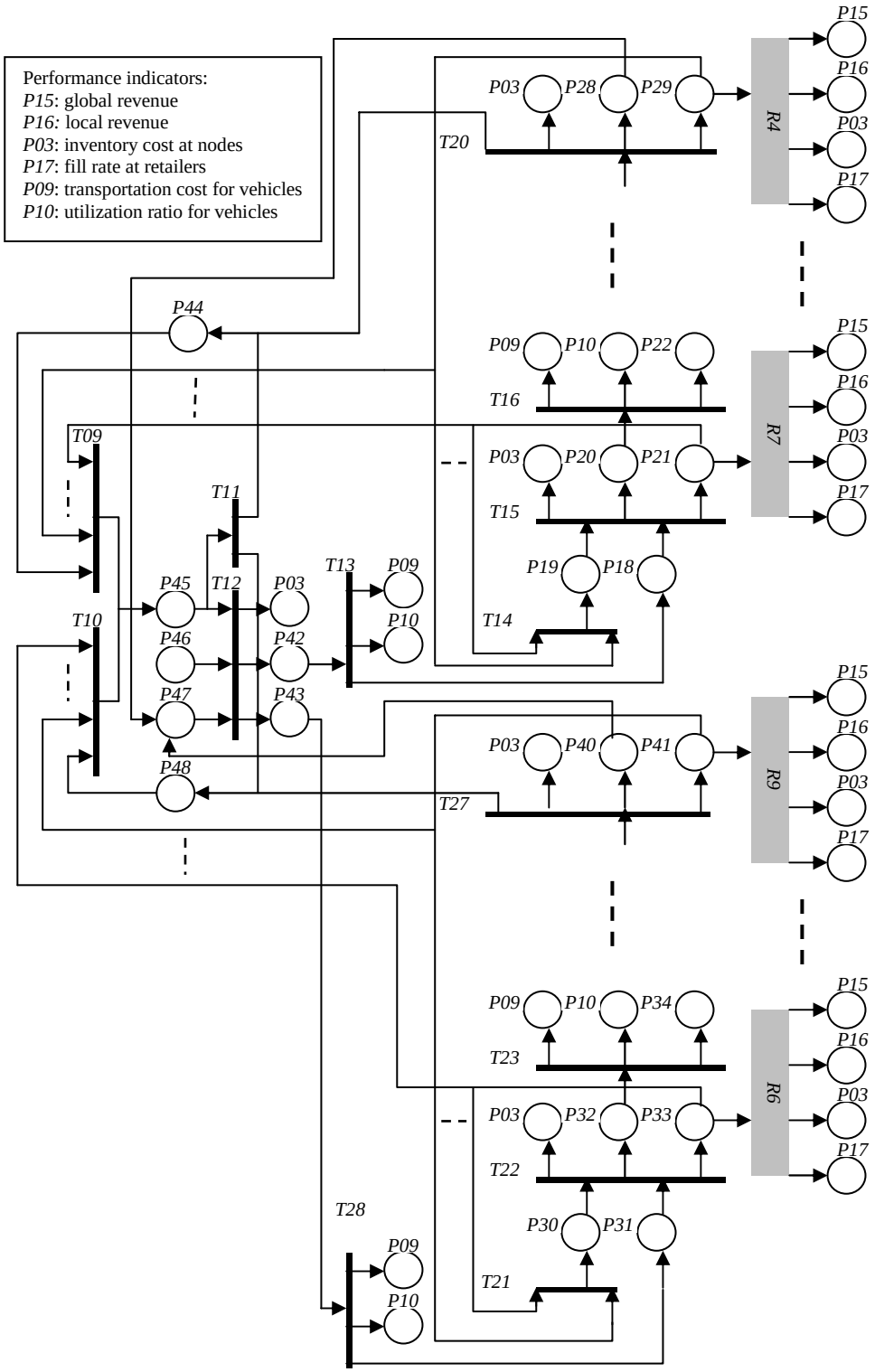


Figure 8-5 Combinatorial Petri net model from wholesaler *W1* to its retailers

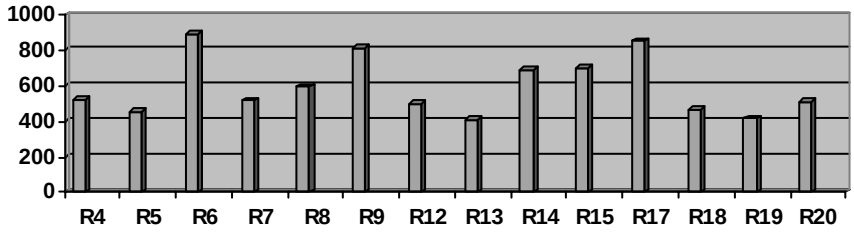


Figure 8-6 Local revenues at retailers (1000$)

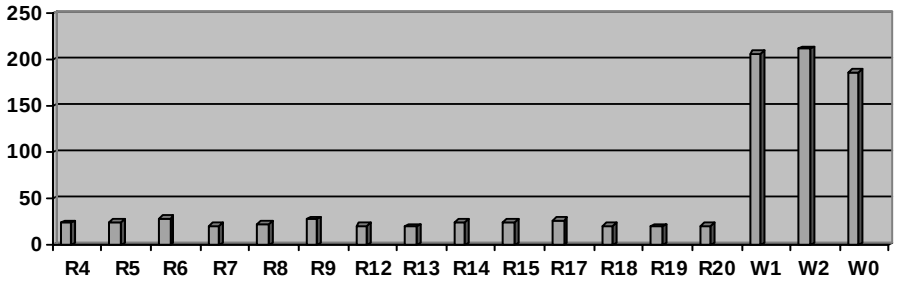
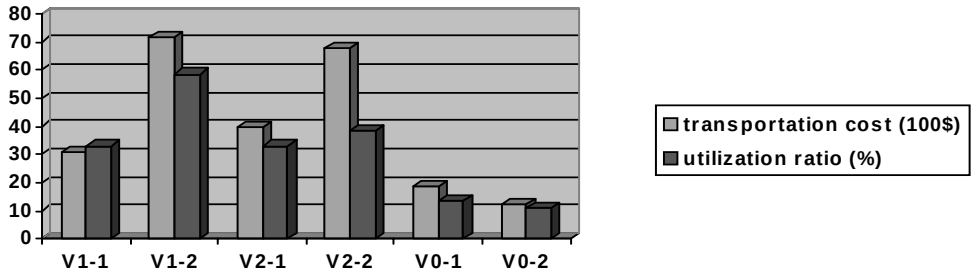


Figure 8-7 Inventory maintaining cost at retailers, wholesalers and distribution center (1000$)



V1-1: the vehicle serving retailers R4, R5, R7. V1-2: the vehicle serving retailers R6, R8, R9
V2-1: the vehicle serving retailers R15, R17, R18, R19 V2-2: the vehicle serving retailers R20, R14, R13, R12.
V0-1: the vehicle serving wholesaler 1 V0-2: the vehicle serving wholesaler 2

Figure 8-8 Transportation cost and utilization ratio for vehicles

CHAPTER 9 CONCLUSION AND FUTURE WORK

9.1 Conclusion

Generally, a supply chain is composed of three parts: supply, production and distribution part. The distribution part plays crucial role for the success of supply chain management. To study it in depth, we separate the distribution part from supply chain, and define it as distribution chain. Like any other systems, a distribution chain needs to be designed before implemented. The primary objective for this dissertation is to provide an integrated methodology for the design of distribution chain.

To get high quality design, first, the structure of this design methodology is developed. According to this structure, before designing a distribution chain, the present status of the host enterprise needs to be analyzed. Based on this analysis, the objective for designing this distribution chain is set. The design objective set in this dissertation is to maximize profit subject to satisfying customer requirements. When designing the distribution chain, first, all possible distributors need to be evaluated, and a set of eligible ones are selected as candidates to design a distribution chain. With these candidates, mathematical models are used to finish the design process. After designing the distribution chain, the performance of this designed distribution chain must be evaluated to verify the design result. This structure acts as guidance in developing the methodology. The methodology developed here can be used to design a distribution chain for medium or large enterprises with mass production.

In the distributor evaluation module, a trilogy is applied. First, to evaluate a distributor more comprehensively, a relatively complete factor set is identified by systematic analysis. All these factors will be considered when evaluating a possible distributor. Secondly, to acquire information on these factors efficiently and economically, a mobile agent system is developed, by which the host enterprise can collect data from possible distributors on Internet. Thirdly, based on these acquired information, an integrated FL-ABL approach is developed to evaluate a distributor quantitatively. Compared with traditional evaluation method, this integrated approach can better reflect the complicated relationship between factors and the evaluation of a firm. At the same time, it successfully integrates the decision makers' opinion into the evaluation process. After evaluating all possible distributors, a set of eligible ones are selected as candidates for designing a distribution chain.

After selecting candidates, we may begin to design the distribution chain. Customer demands at retailers are crucial parameters in designing a distribution chain, but they were not determined precisely in existing design methodologies. To solve this problem, in this dissertation, an ANN based model is developed for market share estimation. First, marketing mix variables that have influence on a firm's market share are identified, then the ANN model is determined to estimate the resulted market share of

this firm in its customer zone. By the estimated market share, the customer demands at retailers can be determined more precisely.

Operation related parameters are another type of parameters which were not determined accurately in existing design methodologies. To solve this problem, in this dissertation, an iterative process is used to design a distribution chain. First, initial values for parameters (including operation related parameters) are specified by experience. Based on these initial values, the configuration of a distribution chain is determined by a MIP model. Given this configuration, the operation models are optimized, i.e. the inventory control parameters at each node are determined by probability theory, and product delivering routes between different nodes are optimized by a genetic algorithm model. After optimizing the operation models, all operation related parameters are re-calculated and updated. These updated parameters are input into the MIP model mentioned above, the configuration of this distribution chain is re-determined, and operation models are re-optimized. This process proceeds until there is no significant difference between two successive designs. As operation related parameters are determined after optimizing the operation models, obviously, they are closer to the exact value than the ones determined by experience in existing methodologies. This design module turns out following results:

- The configuration of the distribution chain, i.e. the numbers and locations of wholesalers and retailers, and the assignments of retailers to wholesalers.
- Inventory control policy and parameters at each node.
- And product delivering routes between different nodes, i.e. from distribution center to wholesalers, and from wholesalers to retailers.

We use performance evaluation to verify the design results. In the performance evaluation module, to simulate the working and inferring process simultaneously in a discrete event system, a new Petri net form: combinatorial Petri net is developed by combining ABL with traditional Petri net. With this newly developed Petri net form, the distribution chain designed above is modelled, and its performance is evaluated by running this model. This performance evaluation module outputs main performance measures for this designed distribution chain. These performance measures can help the decision makers to assess whether this designed distribution chain is satisfactory. If not, the decision-makers may re-consider some subjective judgements or re-assess the objectives set before, and then design the distribution chain again.

Any of these modules can be used individually in practice. For example, the distributor evaluation module can be applied in distributor selection; the performance evaluation module can be used to analyze and evaluate the performance of an existing distribution chain, etc.

All models, algorithms and formulae used in this design methodology have been implemented by computer applications, and most of them are developed in our laboratory. This makes it possible to realize automatic distribution chain deign.

9.2 Future Work

Distribution chain design is a large research area. Because of limited time, it is impossible to solve all problems in one dissertation. In the future, we need to do further research work in following directions:

- *Extend distribution chain design into supply chain design*. In this dissertation, to simplify the problem, we concentrate on the design of distribution chain, and only consider distribution activities. Actually, some parameters in distribution chain (such as the lead time, etc.) are tightly related to production process, and remarkably affected by the supply part. In the future, we need to consider the design of entire supply chain. In principle, it is possible to extend the methodology developed here into the one for design of supply chain. Of course, it will not be a simple extension, as in supply chain, we will face more activities and much more complex situations.
- *Consider other distribution chain forms.* In this dissertation, an integrated methodology is developed to design the basic distribution chain form shown in Figure 4-1(c). In the future, we need to consider other forms shown in this figure, and develop methodology to design them. As the basic form is considered in this dissertation, the methodology developed here will be very helpful in developing other methodologies.
- *Develop commercialized computer application for distribution chain design*. At present, although all models, algorithms and formulae are implemented by computer applications, they are only separate islands. In the future, we will integrate them to form a commercialized computer application for the design of distribution chain. As most of the applications in this dissertation are developed in MATLAB environment, this is not a tough work.

PUBLICATIONS

[1] Hongze Ma, Ziqiong Deng, Wei Deng Solvang, "Optimization of distribution chain structure with considering market share", International Conference of Computational Engineering in Systems Application (cesa2003).

[2] Hongze Ma and Ziqiong Deng, (2002), "Architecture of methodology for distribution chain design", Proceedings of International Conference on e-Business (ICEB2002).

[3] Hongze Ma, Bjorn Solvang, and Ziqiong Deng, (2000), "Realizing efficient and user-friendly simulation for material flow in shop floor", Proceedings of WCC (World Computer Conference) 2000.

---

[4] Hongze Ma, Ziqiong Deng and Wei Deng Solvang, "An on-line approach for distributor benchmarking", accepted for publication in Benchmarking – An International Journal.

---

[5] Hongze Ma and Ziqiong Deng, "An intelligent Petri net approach for performance evaluation of distribution chain", submitted for publication in Petri net Newsletter.

[6] Hongze Ma and Ziqiong Deng, "An integrated approach for the design of distribution chain", submitted for publication in International Journal of Physical Distribution & Logistic Management.

URN:NBN:no-3444

REFERENCES

Alfieri, A., Brandimarte, P. (1997), "Object-oriented modelling and simulation of integrated production/distribution systems", Computer Integrated Manufacturing Systems, Vol. 10, No. 4.

Anthony, D., Ross, A., (2000), "Two-phased approach to the supply network reconfiguration problem", European Journal of Operational Research 122.

Arntzen, B.C., Brown, G.G., (1995), "Global supply chain management at digital equipment corporation", Institute for Operation Research and the Management Sciences.

Aronsson, H., (2000), Three perspectives on supply chain design, Dissertation of Linkoping Institute of Technology, No. 44.

Asbjornsen, O.A., (1992), System engineering—Principles and practices, Marland, System Knowledge Application to Real-time Process Operating Deficiency Diagnosis.

Ballou, R.H., (1992), Business logistics management, 3rd ed., Prentice-Hall, Englewood Cliffs, NJ.

Baunach, B., Mercer, A., Napp, A., (1995), "Sales effects on depot locations", European Journal of Operational Research 81, 474—478.

Beamon, B.M., (1999), "Measuring supply chain performance", International Journal of Operation & Production Management, Volume 19, No. 3

Berman, O., Larson, R.C., (2001), "Deliveries in an inventory/routing problem using stochastic dynamic programming", Transportation Science, Vol. 35, No. 2, May.

Berry, L.M., Murtagh, B.A., Welling, .L.D., (1998), "Generic algorithms in the design of complex distribution networks", International Journal of Physical Distribution & Logistics Management, Volume 28, No.5.

Brown, G.G., Grave, G.W., and Honczarenko, W.D., (1987), "Design and operation of a multicommodity production/distribution system using primal goal decomposition", Management Science, 33/11, 1469—1480.

Carter, C.R., Jennings, M.M., (2002), "Social responsibility and supply chain relationships", Transportation Research Part E-logistics and Transportation Review, 38 (1): 37-52.

Cavusgil, S. T., Yeoh, P. L., (1995), "Selecting foreign distributors, an expert systems approach", Industrial Marketing Management 24.

Chandra, P., (1993), "A dynamic distribution model with warehouse and customer replenishment requirements", Journal of Operation Research Society Vol. 44, No. 7.

Chao, M.I., (2002), "A tabu search method for the truck and trailer routing problem", Computers & Operations Research 29, 33-51.

Chen, R., Gen, M., (1996), "Fuzzy vehicle routing and scheduling problem using genetic algorithms", Genetic Algorithms and Soft Computing, Spriger, Berline, 683-709.

References

Chen, M., Wang, W., (1997), "A linear programming model for integrated steel production and distribution planning", International Journal of Operations & Production Management, Vol. 17, No. 6.

Chen, X., Wan, W., Xu, X., (1998), "Modeling rolling batch planning as vehicle routing problem with time windows", Computers and Operations Research 25 (12), 1127-1136

Chick, S.E., Olsen, T.L., Sethuraman, K., Stecke, K.E., "A descriptive multi-attribute model for reconfigurable machining system selection examing buyer-supplier relationships", International Journal of Agile Management Systems, 2/1, 33-48.

Christofides, N., (1985), "Vehicle routing, the traveling salesman problem", John Wiley & Sons Ltd., NewYork, 431-448.

Cohen, M.A., and Lee, H.L., (1985), Manufacturing strategy: concepts and methods, The Management of Productivity and Technology in Manufacturing, Plenum, New York.

Cohen, M.A., and Lee, H.L., (1989), "Resource deployment analysis of global manufacturing and distribution networks", Journal of Manufacturing Operations Management 2, 81—104.

Cohen, M.A., and Moon, S., (1991), "An integrated plant loading model economies of scale and scope", European Journal of Operational Research 50, 266—279.

Cole. M.H., (1995), Service considerations and the design of strategic distribution systems, Ph.D Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Dasgupta, P., Narasimhan, N., Moser, L.E., Melliar-Smith, P.M., (1999), "MAgNET: Mobile agents for networked electronic trading", IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 4, July/August.

Davidrajuh, R., (2001), Automating supplier selection procedures, Ph.D. Dissertation, Norwegian University of Science and Technology.

Dempster, M., Pedron, N. H., (2000) , "Planning logistics operations in the oil industry", Journal of the Operational Research Society.

Demuth, H., Beale, M., (2001), User's guide for MATLAB, Neural Network Toolbox.

Diks, E.B., Kok, A.G., (1998), "Optimal control of a divergent multi-echelon inventory system", European Journal of Operational Research.

Dogan, K., Goetschalckx, M., (1999), "A primal decomposition method for the integrated design of multi-period production-distribution systems", HE Transactions.

Escudero, L. F., Galindo, E., Garcia, G., (1999), "Schumann, a modeling framework for supply chain management under uncertainty", European Journal of Operational Research 119, 14—34.

Fisher, M. L., (1994), "Vehicle routing, Optimal solution of Vehicle Routing Problems using minimum K-Tress", Operations Research 42: 626—642.

Franksen, O. I. (1979), Group representation of finite polyvalent logic—a case study using APL notation , In Niemi, A. (ed.): A link between science and application of automatic control, Pergamon Press, Oxford and New York.

Gabbert, P., Brown, D., Huntley, C., (1991), "A system for learning routes and schedules with genetic algorithms", Proceedings of Fourth International Conference on Genetic Algorithm, 430-436.

Ganeshan, R., (1999), "Managing supply chain inventories: a multiple retailer, one warehouse, multiple supplier model", International Journal of Production Economics 59.

Gavirneni, S., et al, (1999), "Value of information in capacitated supply chains", Management Science, Vol. 45, No. 1.

Gavirneni, S., (2001), "Benefits of co-operation in a production distribution environment", European Journal of Operation Research 130.

Geoffrion A.M., Graves, (1974), "Multicommodity distribution system design by Benders decomposition", Management Science, 20/5.

Geotschalckx, M., Cole, M.H., Dogan, K., and Wei, R., (1995), "A generic model for the strategic design of production-distribution systems", Version 1.0.

Geunes, J., Amy, Z., (2001), "Impacts of inventory shortage policies on transportation requirements in two-stage distribution systems", European Journal of Operational Research 129.

Gjerdrum, J., et al, (2001), "Transfer prices for multi-enterprise supply chain organization", Ind. Eng. Chem. Res.

Gjerdrum, J., Shan N., et al, (2001), "A combined optimization and agent-based approach to supply chain modeling and performance assessment", Production Planning and Control, Volume 12, No. 1.

Haimovich, M., et al., (1985), "Bounds and heuristics for Capacitated Routing Problems", Mathematics of Operations Research 10:527—542.

Heijden, M.C., (1997), "Supply rationing in multi-echelon divergent systems", European Journal of Operational Research.

Heijden, M.C., (1999), "Inventory control in multi-echelon divergent systems with random lead times", Springer Verlag.

Heijden, M.C., (2000), "Near-optimal inventory control policies for divergent networks under fill rate constraints", International Journal of Production Economics 63, 161-179.

Henk, C. T., Stochastic models, John Wiley & Sons, 1994.

Hoder, J.E., and Dincer, M.C., (1986), "A multi-factor model for international plant location and financing under uncertainty", Computers and Operations Research 13/5.

References

Hornik, K., Stinchcobe, M., (1989), "Multilayer feedforward networks are universal approximators", Neural Networks, 4, 251-257.

Houshyar, A., Lyth, D., (1992), "A systematic supplier selection procedure", Computers and Industrial Engineering, Vol. 23, No 1-4, 173-176.

Huff, D., (1964), "Defining and estimating a trading area", Journal of Marketing, 28, 34-38.

Lakhal, S., Martel, A., (2001), "On the optimization of supply chain networking decision", European Journal of Operational Research 129, 259—270.

Lau, K., Kagan, A., Post, G., (1997), "Marketing share modeling within a switching regression framework", Omega Int. J. Mgmt. Sci. Vol. 25, No. 3, pp. 345-353.

Lee, YH, Kim, SH, Moon, C., (2002), "Production-distribution planning in supply chain using a hybrid approach", Production Planning and Control, 13 (1): 35-46.

Lefteri, H.T., Robert, E.U., (1997), Fuzzy and neural application in engineering, John Wiley & Sons, Inc.

Jain, A. K., Mahajan, V., (1979), "Evaluating the competitive environment in retailing using multiplicative competitive interactive models", Research in Marketing, Greenwich, CT: JAI Press.

Jayaraman,V., (1998), "An efficient heuristic procedure for practical-sized capacitated warehouse design and management", Decision Sciences, Vol. 29, No. 3.

Jayaraman, V., Pirkul, H., (2001), "Planning and coordination of production and distribution facilities for multiple commodities", European Journal of Operational Research 133, 394-408.

Jayashankar, M. S., (1998), "Modelling supply chain dynamics: a multi-agent approach", Decision Science, Volume 29, No. 3.

Jensen, K., (1992), Coloured Petri net: basic concepts, analysis methods and practical use, Vol. 1, Springer-Verlag.

Johnson, M.E., Davis, T., Lee, H.L., (1996), "Robustness of order reliability models with applications to order aging", International Journal of Production Research, 34 (12).

Korpela, J., Lehmusvaara, A., (1999), "A customer oriented approach to warehouse network evaluation and design", International Journal of Production Economics 59.

Kreyszig, E., (1999), Advanced engineering mathematics, John Wiley & Sons. Inc.

Mamdani, E.H., (1974), "Application of Fuzzy Algorithms for Control of Simple Dynamic Plants", Proceedings of IEE, Vol. 121, No. 12.

Mcculloch, W.S., Pitts, W., (1943), "A logical calculus of ideas immanent in nervous activity", Bulletin of Mathematical Biophysics, Vol. 5.

Miltenbueg, J., (1996), "Managing and reducing total cycle time: models and analysis", International Journal of Production Economics, 46-47 (1996) 89-108.

Min, H., Melachrinoudis, E., (1999), "The relocation of a hybrid manufacturing/distribution facility from supply chain perspectives: a case study, Omega. Int. J Mamt. Sci. 27.

Min, S., Mentzer, J.T., (2000), "The role of marketing in supply chain management", International Journal of Physical Distribution & Logistics Management 30, no. 9: 765-787.

Mintzberg, H., Raisinghani, D. & Theoret, A., "The structure of unstructured decision processes", Administrative Science Quarterly, Vol. 21, June, 246—275, 1976.

MirHassani, S.A., et al, (2000), "Computational solution of capacity planning models under uncertainty", Parallel Computing 26.

Mobråten, B., (1996), Performance measurement of logistics processes, Norwegian University of Science and Technology, Trondheim, Norway.

Motwani, J., Youssef, M., Kathawala, Y., Futch, E., (1999), "Supplier selection in developing countries: a model development", Integrated Manufacturing Systems, 10/3, 154-161.

Murata, T., (1989), "Petri nets --- properties, analysis, and applications", Proceedings of the IEEE, 77, April, pp541-580.

Murthy, I., et al, (2001), "Bicriterion distribution planning for agriculture power fuels", INFOR, Vol. 39.

Møller, G. L. (1995), On the technology of Array-Based logic, Ph.D. Dissertation, Technical university of Denmark.

Nagle, T., (1987), "The strategy and tactics of pricing", Prentice-Hall, Englewood California.

Nemhauser, G. L., Wolsey, L. A., (1988), Integer and combinatorial optimization, John Wiley & Sons. Inc.

Noon, C. E., Mitththal, J., (1991), "A TSSP+1 decomposition approach for the capacity constrained vehicle routing problem", Working paper, Management Science Group, Univensity of Tennessee, Knoxville, TN.

Paessens, H., (1988), "The Savings Algorithm for the Vehicle Routing Problem", European Journal of Operations Research, 34: 336—344.

Park, Y.B., (2001), "A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines", International Journal of Production Economics 73, 175-188.

Patrick, H.W., (1993), Artificial intelligence, Addison-Wesley Publishing Company.

Patterson, D.W., (1996), Artificial neural network – theory and applications, Prentice Hall.

Popken, D.A., (1994), An algorithm for multi-attribute, multi-commodity flow problem with freight consolidation and inventory costs, Operational Research 42, 274-286.

URN:NBN:no-3444

References

Rao, B., (1999), "The Internet and the revolution in distribution: a cross-industry examination", Institute for Technology and Enterprise, Polytechnic University, New York, USA.

Ray, E.T., (2001), Learning XML, O'Reilly & Associates, Inc.

Reis, J., et al, (2001), "Locally perceiving hard global constraints in multi-agent scheduling", Journal of Intelligent Manufacturing 12.

Reisig, W., (1987), Place/Transition systems: Lecture notes on Computer Science, Vol. 254, Springer Verlag.

Reuven, Y. R., (1998), Modern simulation and modelling, John Wiley & Sons, Inc. 1998

Robinson, E.P., (1998), "Decision distribution systems to support vendor strategies in supply chain management", Decision Sciences, Vol. 29, No. 3.

Saaty TL, (1980), The analytic hierarchy process, New York, NY: McGraw-Hill Book Co.

Sabri, E.H., Beamon, B.M., (2000), "A multi-objective approach to simultaneous strategic and operational planning in supply chain design", International Journal of Management Science, Vol. 28.

Scarpelli, H., et al, (1994), High-level fuzzy Petri net and backward reasoning, Proc. IPMU, Paris, France, 1275-1280.

Silver, E.A., Peterson, R., (1985), Decision systems for inventory management and production planning, New York, Wiley.

Singh, M.G., (1996), "Knowledge support for profitable pricing in a competitive environment", 4th Intl. conference on the Cognitive Foundations of Economics and Management, Paris.

Singh, M.G., Cassaigne, N., (1997), "IT support for the generic tactical decision making process of pricing in competitive consumer markets", Proceedings of the OE/IFIP/IEEE International Conference on Integrated and Sustainable Industrial Production.

Solvang, W.D., (2001), Architecture for supply chain analysis and methodology for quantitative measurement of supply chain flexibility, Ph.D dissertation of Norwegian University of Science and Technology.

Swaminathan, J.S., (1998), "Modeling supply chain dynamics: a multi-agent approach", Decision Science, Volume 29, No. 3.

Tayur, S., Ganeshan, R., Magazine, M., (1999), Quantitative models for supply chain management, Kluwer Academic Publishers, Boston/Dordrecht/London.

Themido, I., Arantes, A., Fernandes, A.P.,(2000), "Logistics costs case study----an ABC approach", Journal of the Operational Research Society 51, 1148—1157.

Tijms, H.C., (1994), Stochastic Models: An Algorithmic Approach, John Wiley & Sons Ltd. P51-71.

Tsoukalas,L.H., Uhrig, R.E., (1997), Fuzzy and Neural Approaches in Engineering, John Wiley & sons, Inc

Venkatesh, K., et al, (1996), A Petri net approach to investigating push and pull paradigms in flexible factory automated systems, International Journal of Production Research, Vol.34, No. 3.

Vidal, C. J., Goetschalckx, M., (1997), "Strategic production-distribution models: a critical review with emphasis on global supply chain models", European Journal of Operational Research 98, 1-18.

Vidal, C. J., et al, (2001), "A global supply chain model with transfer pricing and transportation cost allocation", European Journal of Operation Research 129.

Viswanadham, N., et al, (1997), "Flexibility in manufacturing enterprise", Sadhana, Vol. 22.

Viswanathan, S., Mathur, K., (1997), "Integrating routing and inventory decisions in one-warehouse multi-retailer multi-product distribution systems", Management Science, Vol. 43, No. 3.

Wendy, W.Q., James, H.B., Paul, I., (1999), "An integrated inventory-transportation system with modified periodic policy for multiple products", European Journal of Operational Research 115, 254-269.

Weng, Z. K., (1999), "The power of coordinated decisions for short-life-cycle products in a manufacturing and distribution supply chain", HE transactions.

Wu, B. (1994), Manufacturing systems design and analysis: context and techniques, Second edition, Chapman & Hall.

Yam, R., Lo, W., (2000), "Enhancement of global competitiveness for Hong Kong/China manufacturing industries through I-agile virtual enterprising", Managing Innovative Manufacturing Conference (MIM2000), Aston Business School, U.K.

Zadeh, L. A., (1965), Fuzzy Sets, Information and Control, Vol. 8.

References

Appendix A Program in Distributor Evaluation Module

A-1 Evaluation of "Inventory maintaining facility"

```
%A subsystem for the evaluation of inventory facility

%declaring a new fis system
a=newfis('invfacility')

%declaring the first input variable
a=addvar(a, 'input', 'floorSpace', [1000 10000]);
%defining the membership function
a=addmf(a, 'input', 1, 'small', 'gaussmf', [1500 1000]);
a=addmf(a, 'input', 1, 'medium', 'gaussmf', [1500 5500]);
a=addmf(a, 'input', 1, 'large', 'gaussmf', [1500 10000]);

%declaring the second input variable
a=addvar(a, 'input', 'costInv', [5 15]);
%defining the membership function
a=addmf(a, 'input', 2, 'expensive', 'gaussmf', [3 15]);
a=addmf(a, 'input', 2, 'medium', 'gaussmf', [3 10]);
a=addmf(a, 'input', 2, 'cheap', 'gaussmf', [3 5]);

%declaring the third input variable
a=addvar(a, 'input', 'relia', [0 100]);
%defining the membership function
a=addmf(a, 'input', 3, 'unreliable', 'trapmf', [8 10 100 102]);
a=addmf(a, 'input', 3, 'medium', 'trapmf', [2 5 8 11]);
a=addmf(a, 'input', 3, 'ok', 'trapmf', [-5 -2 2 5]);

%declaring the output variable variable
a=addvar(a, 'output', 'invFacility', [0 100]);
%defining the membership function
a=addmf(a, 'output', 1, 'bad', 'gaussmf', [15 0]);
a=addmf(a, 'output', 1, 'eligible', 'gaussmf', [15 50]);
a=addmf(a, 'output', 1, 'excellent', 'gaussmf', [15 100]);

%defining inference rules
ruleList=[...
    1 1 1 1 1 1
    1 1 2 1 1 1
    1 1 3 2 1 1
    1 2 1 2 1 1
    1 2 2 2 1 1
```

```
    1 2 3 2 1 1
    1 3 1 2 1 1
    1 3 2 2 1 1
    1 3 3 2 1 1
    2 1 1 1 1 1
    2 1 2 2 1 1
    2 1 3 2 1 1
    2 2 1 2 1 1
    2 2 2 2 1 1
    2 2 3 2 1 1
    2 3 1 2 1 1
    2 3 2 3 1 1
    2 3 3 3 1 1
    3 1 1 1 1 1
    3 1 2 2 1 1
    3 1 3 2 1 1
   3 2 1 2 1 1
    3 2 2 3 1 1
    3 2 3 3 1 1
    3 3 1 2 1 1
    3 3 2 3 1 1
    3 3 3 3 1 1
];

%add rules to the system
a=addrule(a, ruleList);

% assign input value and evaluating the inventory facility system
invFacilityVal=evalfis([6000 8 5], a)
```

A-2 Evaluation of “Transportation facility”

```
%A subsystem for the evaluation of transportation facility

%declaring a new fis system
a=newfis('transFacility')

%declaring the first input variable
a=addvar(a, 'input', 'throughput', [5 100]);
%defining the membership function
a=addmf(a, 'input', 1, 'small', 'gaussmf', [10 5]);
a=addmf(a, 'input', 1, 'medium', 'gaussmf', [10 50]);
a=addmf(a, 'input', 1, 'large', 'gaussmf', [10 100]);

%declaring the second input variable
a=addvar(a, 'input', 'tranCost', [5 150]);
```

```
%defining the membership function
a=addmf(a, 'input', 2, 'expensive', 'gaussmf', [20 150]);
a=addmf(a, 'input', 2, 'average', 'gaussmf', [20 70]);
a=addmf(a, 'input', 2, 'cheap', 'gaussmf', [20 5]);

%declaring the third input variable
a=addvar(a, 'input', 'reliability', [0 100]);
%defining the membership function
a=addmf(a, 'input', 3, 'unreliable', 'trapmf', [8 10 100 102]);
a=addmf(a, 'input', 3, 'medium', 'trapmf', [2 5 8 11]);
a=addmf(a, 'input', 3, 'reliable', 'trapmf', [-5 -2 2 5]);

%declaring the output variable variable
a=addvar(a, 'output', 'transFacility', [0 100]);
%defining the membership function
a=addmf(a, 'output', 1, 'bad', 'gaussmf', [15 0]);
a=addmf(a, 'output', 1, 'eligible', 'gaussmf', [15 50]);
a=addmf(a, 'output', 1, 'excellent', 'gaussmf', [15 100]);

%defining inference rules
ruleList=[...
  1 1 1 1 1 1
  1 1 2 1 1 1
  1 1 3 2 1 1
  1 2 1 2 1 1
  1 2 2 2 1 1
  1 2 3 2 1 1
  1 3 1 2 1 1
  1 3 2 2 1 1
  1 3 3 2 1 1
  2 1 1 1 1 1
  2 1 2 2 1 1
  2 1 3 2 1 1
  2 2 1 2 1 1
  2 2 2 2 1 1
  2 2 3 2 1 1
  2 3 1 2 1 1
  2 3 2 3 1 1
  2 3 3 3 1 1
  3 1 1 1 1 1
  3 1 2 2 1 1
  3 1 3 2 1 1
  3 2 1 2 1 1
  3 2 2 3 1 1
  3 2 3 3 1 1
```

```
  3 3 1 2 1 1
  3 3 2 3 1 1
  3 3 3 3 1 1
];

%add rules to the system
a=addrule(a, ruleList);

%load the fis system
%a=readfis('invfacility')

% assign input value and evaluating the inventory facility system
insTransportationVal=evalfis([90 100 2], a)
```

A-3 Evaluation of "Human resource"

```
%A subsystem for the evaluation of human factor
%declaring a new fis system
a=newfis('humanFactor')

%declaring the first input variable
a=addvar(a, 'input', 'noOfEmployee', [50 1000]);
%defining the membership function
a=addmf(a, 'input', 1, 'small', 'gaussmf', [150 50]);
a=addmf(a, 'input', 1, 'large', 'gaussmf', [150 1000]);
a=addmf(a, 'input', 1, 'medium', 'gaussmf', [150 500]);

%declaring the second input variable
a=addvar(a, 'input', 'salary', [1000 5000]);
%defining the membership function
a=addmf(a, 'input', 2, 'high', 'gaussmf', [500 5000]);
a=addmf(a, 'input', 2, 'average', 'gaussmf', [500 3000]);
a=addmf(a, 'input', 2, 'low', 'gaussmf', [500 1000]);

%declaring the third input variable
a=addvar(a, 'input', 'eduState', [0 100]);
%defining the membership function
a=addmf(a, 'input', 3, 'low', 'gaussmf', [20 0]);
a=addmf(a, 'input', 3, 'average', 'gaussmf', [20 50]);
a=addmf(a, 'input', 3, 'high', 'gaussmf', [20 100]);

%declaring the output variable variable
a=addvar(a, 'output', 'transFacility', [0 100]);
%defining the membership function
a=addmf(a, 'output', 1, 'bad', 'gaussmf', [15 0]);
a=addmf(a, 'output', 1, 'eligible', 'gaussmf', [15 50]);
```

```
a=addmf(a, 'output', 1, 'excellent', 'gaussmf', [15 100]);

%defining inference rules
ruleList=[...
   1 1 1 1 1 1
   1 1 2 1 1 1
   1 1 3 2 1 1
   1 2 1 2 1 1
   1 2 2 2 1 1
   1 2 3 2 1 1
   1 3 1 2 1 1
   1 3 2 2 1 1
   1 3 3 2 1 1
   2 1 1 1 1 1
   2 1 2 2 1 1
   2 1 3 2 1 1
   2 2 1 2 1 1
   2 2 2 2 1 1
   2 2 3 2 1 1
   2 3 1 2 1 1
   2 3 2 3 1 1
   2 3 3 3 1 1
   3 1 1 1 1 1
   3 1 2 2 1 1
   3 1 3 2 1 1
   3 2 1 2 1 1
   3 2 2 3 1 1
   3 2 3 3 1 1
   3 3 1 2 1 1
   3 3 2 3 1 1
   3 3 3 3 1 1
];

%add rules to the system
a=addrule(a, ruleList);

%load the fis system
%a=readfis('invfacility')

% assign input value and evaluating the inventory facility system
insTransportationVal=evalfis([500 2000 50], a)
```

A-4 Evaluation of “Financial factor”

```
%A subsystem for the evaluation of financial factor
%declaring a new fis system
```

Appendix

```
a=newfis('financFactor')

%declaring the first input variable
a=addvar(a, 'input', 'finSale', [1 5]);
%defining the membership function
a=addmf(a, 'input', 1, 'small', 'gaussmf', [0.5 1]);
a=addmf(a, 'input', 1, 'medium', 'gaussmf', [0.5 3]);
a=addmf(a, 'input', 1, 'large', 'gaussmf', [0.5 5]);

%declaring the second input variable
a=addvar(a, 'input', 'finAdd', [1 10]);
%defining the membership function
a=addmf(a, 'input', 2, 'small', 'gaussmf', [1 1]);
a=addmf(a, 'input', 2, 'medium', 'gaussmf', [1 5]);
a=addmf(a, 'input', 2, 'large', 'gaussmf', [1 10]);

%declaring the third input variable
a=addvar(a, 'input', 'finAd', [0 5]);
%defining the membership function
a=addmf(a, 'input', 3, 'small', 'gaussmf', [0.5 0]);
a=addmf(a, 'input', 3, 'medium', 'gaussmf', [0.5 2.5]);
a=addmf(a, 'input', 3, 'large', 'gaussmf', [0.5 5]);

%declaring the output variable variable
a=addvar(a, 'output', 'transFacility', [0 100]);
%defining the membership function
a=addmf(a, 'output', 1, 'bad', 'gaussmf', [15 0]);
a=addmf(a, 'output', 1, 'eligible', 'gaussmf', [15 50]);
a=addmf(a, 'output', 1, 'excellent', 'gaussmf', [15 100]);

%defining inference rules
ruleList=[...
  1 1 1 1 1 1
  1 1 2 1 1 1
  1 1 3 2 1 1
  1 2 1 2 1 1
  1 2 2 2 1 1
  1 2 3 2 1 1
  1 3 1 2 1 1
  1 3 2 2 1 1
  1 3 3 2 1 1
  2 1 1 1 1 1
  2 1 2 2 1 1
  2 1 3 2 1 1
  2 2 1 2 1 1
```

```
  2 2 2 2 1 1
  2 2 3 2 1 1
  2 3 1 2 1 1
  2 3 2 3 1 1
  2 3 3 3 1 1
  3 1 1 1 1 1
  3 1 2 2 1 1
  3 1 3 2 1 1
  3 2 1 2 1 1
  3 2 2 3 1 1
  3 2 3 3 1 1
  3 3 1 2 1 1
  3 3 2 3 1 1
  3 3 3 3 1 1
];

%add rules to the system
a=addrule(a, ruleList);

%load the fis system
%a=readfis('invfacility')

% assign input value and evaluating the inventory facility system
insTransportationVal=evalfis([3 5 1], a)
```

A-5 Evaluation of “Communication system”

```
%evaluating the communication system
%declaring input variables
communicationCost=element('n', {'high', 'fair', 'low'}, {}, 'Communication Cost');
communicationMethod=element('n', {'telephon', 'fax', 'Internet', 'telephon_fax',
'telephon_Internet', 'fax_Internet', 'telephon_fax_Internet',
'telepnon_fax_Internet_Intranet'}, {}, 'communication Method');
%declaring output variables
commSystem=element('n',{'1', '2', '3', '4', '5'}, {}, 'Communication System');

%assing input value
x11 = assign(communicationCost, {'high'});
x12 = assign(communicationCost, {'fair'});
x13 = assign(communicationCost, {'low'});

x21 = assign(communicationMethod, {'telephon', 'fax'});
x22 = assign(communicationMethod, {'Internet', 'telephon_fax', 'telephon_Internet',
'fax_Internet'});
x23 = assign(communicationMethod, {'telephon_fax_Internet',
'telepnon_fax_Internet_Intranet'});
```

```
%assign output value
y1 = assign(commSystem, {'1'});
y2 = assign(commSystem, {'2'});
y3 = assign(commSystem, {'3'});
y4 = assign(commSystem, {'4'});
y5 = assign(commSystem, {'5'});

%inference rules
y=conjunct(x11, x21);
Premise_1=bimp(y, y1);

y21=conjunct(x11, x22);
y22=conjunct(x12, x21);
y=disjunct(y21, y22);
y=fuse(y);
Premise_2=bimp(y, y2);

y31=conjunct(x11, x23);
y32=conjunct(x12, x22);
y=disjunct(y31, y32);
y=fuse(y);
y33=conjunct(x13, x21);
y=disjunct(y, y33);
y=fuse(y);
Premise_3=bimp(y, y3);


y41=conjunct(x12, x23);
y42=conjunct(x13, x22);
y=disjunct(y41, y42);
y=fuse(y);
Premise_4=bimp(y, y4);


y=conjunct(x13, x23);
Premise_5=bimp(y, y5);

%getting the inference system
Premise_12 = conjunct(Premise_1, Premise_2); % combination of the first two
premises
Premise_12 = fuse(Premise_12); % remove the duplicates
Premise_123 = conjunct(Premise_12, Premise_3); % combination of all three
Premise_123 = fuse(Premise_123);
Premise_1234   = conjunct(Premise_123, Premise_4);
```

```
Premise_1234 = fuse(Premise_1234);
Premise_12345 = conjunct(Premise_1234, Premise_5);
System  = fuse(Premise_12345); % remove the duplicates gives the system

%assigning input vector
t2 =assign(communicationMethod, {'telephon_fax_Internet'});
%handle the input variable communication cost
inputCost=150;  %input the communication cost value
minCost=20;
setPoint1=100;
setPoint2=200;
maxCost=250;
t1 =assign(communicationCost, {'high'});
if ((inputCost>=minCost) & (inputCost<=setPoint1))
  t1 =assign(communicationCost, {'low'});
elseif ((inputCost>setPoint1) & (inputCost<=setPoint2))
  t1 =assign(communicationCost, {'fair'});
elseif ((inputCost>setPoint2) & (inputCost<=maxCost))
  t1 =assign(communicationCost, {'high'});
end
input_SV = [t1 t2]          % put together the individual inputs as an input vector

%get output vector
output_SV = state(input_SV, System)     %computing the output state vector
%print(output_SV)

%get output value
for i=1:length(output_SV.array)
  if (output_SV.array(i))
    communicationVal=output_SV.elements.domain{i}
  end
end

%changing the string value into number
communicationVal=str2num(communicationVal)
```

A-6 Evaluation of “Hardware”

```
%A subsystem for the evaluation of hardware
%declaring a new fis system
a=newfis('software')

%declaring the first input variable
a=addvar(a, 'input', 'invFacility', [0 100]);
%defining the membership function
a=addmf(a, 'input', 1, 'bad', 'gaussmf', [15 0]);
```

```
a=addmf(a, 'input', 1, 'eligible', 'gaussmf', [15 50]);
a=addmf(a, 'input', 1, 'excellent', 'gaussmf', [15 100]);

%declaring the second input variable
a=addvar(a, 'input', 'transFacility', [0 100]);
%defining the membership function
a=addmf(a, 'input', 2, 'bad', 'gaussmf', [15 0]);
a=addmf(a, 'input', 2, 'eligible', 'gaussmf', [15 50]);
a=addmf(a, 'input', 2, 'excellent', 'gaussmf', [15 100]);

%declaring the third input variable
a=addvar(a, 'input', 'commSystem', [1 5]);
%defining the membership function
a=addmf(a, 'input', 3, 'bad', 'gaussmf', [0.5 1]);
a=addmf(a, 'input', 3, 'eligible', 'gaussmf', [0.5 3]);
a=addmf(a, 'input', 3, 'excellent', 'gaussmf', [0.5 5]);

%declaring the forth input variable
a=addvar(a, 'input', 'humanFactor', [0 100]);
%defining the membership function
a=addmf(a, 'input', 4, 'bad', 'gaussmf', [15 0]);
a=addmf(a, 'input', 4, 'eligible', 'gaussmf', [15 50]);
a=addmf(a, 'input', 4, 'excellent', 'gaussmf', [15 100]);

%declaring the fifth input variable
a=addvar(a, 'input', 'finFactor', [0 100]);
%defining the membership function
a=addmf(a, 'input', 5, 'bad', 'gaussmf', [15 0]);
a=addmf(a, 'input', 5, 'eligible', 'gaussmf', [15 50]);
a=addmf(a, 'input', 5, 'excellent', 'gaussmf', [15 100]);

%declaring the output variable variable
a=addvar(a, 'output', 'transFacility', [0 100]);
%defining the membership function
a=addmf(a, 'output', 1, 'bad', 'gaussmf', [15 0]);
a=addmf(a, 'output', 1, 'eligible', 'gaussmf', [15 50]);
a=addmf(a, 'output', 1, 'excellent', 'gaussmf', [15 100]);

%defining inference rules
ruleList=[...
    1 1 1 1 1 1 1 1
    1 1 2 1 2 1 1 1
    1 1 3 2 2 2 1 1
    1 2 1 2 1 2 1 1
    1 2 2 2 3 2 1 1
```

```
    1 2 3 2 1 2 1 1
    1 3 1 2 1 2 1 1
    1 3 2 1 2 2 1 1
    1 3 3 2 1 2 1 1
    2 1 1 1 1 1 1 1
    2 1 2 2 1 2 1 1
    2 1 3 1 1 2 1 1
    2 2 1 1 2 2 1 1
    2 2 2 2 2 2 1 1
    2 2 3 1 2 2 1 1
    2 3 1 3 1 2 1 1
    2 3 2 3 2 3 1 1
    2 3 3 2 3 3 1 1
    3 1 1 1 2 1 1 1
    3 1 2 3 1 2 1 1
    3 1 3 1 3 2 1 1
    3 2 1 2 1 2 1 1
    3 2 2 3 2 3 1 1
    3 2 3 3 3 3 1 1
    3 3 1 3 1 2 1 1
    3 3 2 3 3 3 1 1
    3 3 3 3 3 3 1 1
];

%add rules to the system
a=addrule(a, ruleList);

%load the fis system
%a=readfis('invfacility')

% assign input value and evaluating the inventory facility system
insTransportationVal=evalfis([60.1 75.1 4 66.7 50], a)
```

Appendix B Program in Module of Distribution Chain Design

B-1 Determine inventory control model at a retailer

```
SS=[140:10:180];  %%Specify the searching range for order up to level
ss=[30:10:70];        %%Specify the searching range for reorder point
LL=zeros(5,5);
LLS=zeros(5,5);
LLs=zeros(5,5);
for k=1:5
  for l=1:5
     N=40;   %%number of regenerative processes
     M=200;   %%maximum time number of one regenerative cycle
     Dr=0;     %%sum of demand during lead time,initial value
     B=10;      %%lead time
     S=SS(k); %%order up to level
     s=ss(l);   %%reordering point
     Q=S-s;
     S0=SS(3);  %base value for S
     s0=ss(3);  % base value for s
     Q0=S0-s0;   %base value for Q
     Ch=1;    %%constant for holding cost
     Cs=10;    %%constant for shortage cost
     Corder=1000;  %%ordering cost
     uT=1;    %%mean of interarrival time
     sT=0.2;     %%standard deviation of interarrival time
     uD=5.6;   %%mean of demand
     sD=1; %%standard deviation of demand
     T0=randn(N,M);   %%generate standard normal distribution matrix
     T=uT+sT*T0;    %%generaate interarrival time matrix
     D0=randn(N,M);
     D1=uD+sD*D0;
     D2=zeros(N,1);
     D=[D2 D1];       %%generate demand matrix
     L1=0;            %%initial value for the numerator of L
     L2=0;            %%initial value for the denominator of L
     L1i=zeros(1,N);        %%in one regenerative cycle, initial value for numerator
     L2i=zeros(1,N);        %%in one regenerative cycle, initial value for denominator
     W1=ones(N, 1);         %%initial value for W1,likelihood ratio for demand
     for i=1:N       %%regenerative cycles
        Dr=0;     %%initial value for sum of lead time demand
        Tsum=0;    %%sum of time passed in one regenerative cycle
        j=1;          %%j is customer indicator in one regenerative cycle

        while Tsum<100
```

```
            Lt=S-Dr;    % net inventory level after replenishment
            Dr=0;      %%sum of demand during lead time
            Tr=0;        %%initial value for sum of time after placing order
            ti=0;
            while Tr<B   %%before the replenishment order comes
              if Lt>s  %%before placing and an order
                 Lt=Lt-D(i,j);       %%the demand is fulfilled
                 cost=Ch*abs(T(i,j))*Lt;  %%holding cost between two successive orders
                 Tsum=Tsum+abs(T(i,j));
                 L1i(1,i)=L1i(1,i)+cost*W1(i);  %%numerator for one regenerative cycle
                 ti=ti+T(i,j);
                 j=j+1;                    %%next demand
              else                        %% if Lt>s0, after replacing an order
                 Tr=Tr+abs(T(i,j));              %%sum of time after placing an order
                 Dr=Dr+D(i,j);              %%sum of demands after placing an order
                 Lt=Lt-D(i,j);
                 if Lt>0
                    cost=Ch*abs(T(i,j))*Lt;
                 else                     %%a shortage occurs
                    cost=Cs*abs(T(i,j))*abs(Lt);
                 end
                 L1i(1,i)=L1i(1,i)+cost*W1(i);
                 Tsum=Tsum+abs(T(i,j));
                 ti=ti+T(i,j);
                 j=j+1;
              end %if Lt>s0
            end %while Tr<B
          L1i(1,i)=L1i(1,i)+Corder;
          %L2i(1,i)=1+(ti-1)*W1(i);
          L2i(1,i)=L2i(1,i)+ti;
        end %while Tsum<100
      end %for i=1:N
      L1=sum(L1i');
      L2=sum(L2i');
      L=L1/L2;   %%estimator of the performance
      LL(k,l)=L;
      LLS(k,l)=S;
      LLs(k,l)=s;
    end %for l=1:5
end %for k=1:5
surf(SS, ss, LL')
```

B-2 Planning product delivery routes by genetic algorithm

```
% generate 4*10 chromosomes
```

```
for i=1:4
   for j=1:10
      chro(i,j).retailer=randperm(15);
   end
end

%order up level for each node

node(1).S=500;
node(2).S=1100;
node(3).S=1800;
node(4).S=1000;
node(5).S=2000;
node(6).S=1600;
node(7).S=1200;
node(8).S=1000;
node(9).S=2000;
node(10).S=2000;
node(11).S=1300;
node(12).S=1100;
node(13).S=2000;
node(14).S=1400;
node(15).S=2200;

%mean of demand for each node

node(1).d=30;
node(2).d=85;
node(3).d=150;
node(4).d=67;
node(5).d=170;
node(6).d=130;
node(7).d=80;
node(8).d=60;
node(9).d=150;
node(10).d=170;
node(11).d=90;
node(12).d=80;
node(13).d=110;
node(14).d=130;
node(15).d=150;

%standard deviation for each node

node(1).sigma=3;
```

```
node(2).sigma=9;
node(3).sigma=10;
node(4).sigma=8;
node(5).sigma=17;
node(6).sigma=14;
node(7).sigma=8;
node(8).sigma=6;
node(9).sigma=15;
node(10).sigma=17;
node(11).sigma=9;
node(12).sigma=8;
node(13).sigma=11;
node(14).sigma=13;
node(15).sigma=15;

%location for each node

node(1).loc=[30,20];
node(2).loc=[40,20];
node(3).loc=[50,10];
node(4).loc=[70,80];
node(5).loc=[40,50];
node(6).loc=[80,100];
node(7).loc=[20,30];
node(8).loc=[80,10];
node(9).loc=[20,50];
node(10).loc=[30,70];
node(11).loc=[70,20];
node(12).loc=[20,10];
node(13).loc=[50,20];
node(14).loc=[30,80];
node(15).loc=[20,100];

%s: reorder point, Q: order quantity, T: inter-service time,
%alpha: unit cost for earluness, belta: unit cost for lateness,
%gama: penalty for back order, safety factor n=3, lead time L=1.

n=3; L=1;
for i=1:15
   node(i).s=n*node(i).sigma+L*node(i).d;
   node(i).Q=node(i).S-node(i).s;
   node(i).T=node(i).Q/node(i).d;
   node(i).alpha=1;
   node(i).belta=0.5;
   node(i).gama=10;
```

```
    node(i).safe=3;
end

Cveh=4500; %capacity of the vehicles: 4500

for i=1:15
    for j=1:15
        dis(i,j)=(node(i).loc(1)-node(j).loc(1))^2+(node(i).loc(2)-node(j).loc(2))^2;
    end
end

%initialize routes for each chromosome

for i=1:4
    for j=1:10
        chro(i,j).routes=[];
        Chro=chro(i,j).retailer;
        chro(i,j).vehicle=zeros(1,15);
        k=0;
        while (~isempty(Chro))
            k=k+1;
            route(i,j,k).ret=[Chro(1)];  %first node in the new route
            route(i,j,k).nofret=1;       %calculate the number of nodes in one route
            route(i,j,k).vehicle=k;      %vehicle number for the route
            chro(i,j).routes=[chro(i,j).routes k];
            Reta=Chro(1);                %the node to be added into a route
                chro(i,j).vehicle(Reta)=k; %assign vehicle number to the corresponding
                                           node position
            route(i,j,k).T=node(Reta).T; %calculate the inter-service time for one route
            route(i,j,k).Cleft=Cveh-node(Reta).Q;  %left capacity for a route
            Chro(1)=[];                  %delete the node from original chromosome
            if (isempty(Chro))
                break
            end
            while (route(i,j,k).Cleft>=0)  %check if there is more vehicle capacity
                Dis=[];
                for n=1:length(Chro)
                    Dis=[Dis dis(Reta, Chro(n))];
                end
                [Y I]=min(Dis);          %find the shortest distance, and indexed as I
                Reta=Chro(I);            %Ith node is the one to be assigned into the present
                                                route
                if (route(i,j,k).Cleft>=node(Reta).Q)     %if the left capacity is enough for this
                                                            node
                    route(i,j,k).ret=[route(i,j,k).ret Chro(I)]; %adding this node into the route
```

```
            chro(i,j).vehicle(Reta)=k;
            route(i,j,k).nofret=route(i,j,k).nofret+1;
            route(i,j,k).T=route(i,j,k).T+node(Reta).T;
            Chro(l)=[];                    %delete the node from original chromosome
         end
         route(i,j,k).Cleft=route(i,j,k).Cleft-node(Reta).Q; %left capacity for the route
         if (isempty(Chro))   %if there is no more nodes, just end this chromosome
                                routing
            break
         end
      end  %while (route(i,j,k).Cleft>=0)
         if (route(i,j,k).Cleft<0)   %if the left capacity is negative, add the last Q to it
            route(i,j,k).Cleft=route(i,j,k).Cleft+node(Reta).Q;
         end
            route(i,j,k).T=route(i,j,k).T/route(i,j,k).nofret;
   end %while (~isempty(Chro))
   chro(i,j).nofroute=k;
 end  %for j=1:10
end  %for i=1:4

%begin to calculate performances for one route

for i=1:4
   totalCost(i,1)=0;
   for j=1:10
      %chro(i,j).fitness=0;
      chro(i,j).distanceCost=0;
      chro(i,j).timeCost=0;

      for k=1:chro(i,j).nofroute    %for each route in one chromosome
         route(i,j,k).L=1;

          %begin to calculate distance related cost

         route(i,j,k).travelD=0;   %travelD: traveling distance for one route
         route(i,j,k).serviceD=0;  %serviceD: service traveling distance for one route
         route(i,j,k).sequence=[];  %sequence: serving squence of nodes in one route
         Ret=route(i,j,k).ret;     %Ret: indicate the nodes in one route
         Dis=[];
         for n=1:length(Ret)
            Dist=(node(Ret(n)).loc(1))^2+(node(Ret(n)).loc(2))^2;
            Dis=[Dis Dist];
         end
         [Y I]=min(Dis);          %find the closest node to warehouse
         Reta=Ret(I);
```

```
        route(i,j,k).sequence=[route(i,j,k).sequence Reta];  %first node to be served
        Ret(l)=[];              %delete this node from the array of nodes in one route
        route(i,j,k).travelD=route(i,j,k).travelD+Y^(1/2);
        while (~isempty(Ret))
           Dis=[];
           for n=1:length(Ret)
              Dis=[Dis dis(Reta, Ret(n))];
           end
           [Y l]=min(Dis);      %find the closest one
           Reta=Ret(l);         %this node is saved in Reta temporarily
           route(i,j,k).sequence=[route(i,j,k).sequence Reta];
           Ret(l)=[];           %delete this node
           route(i,j,k).travelD=route(i,j,k).travelD+Y^(1/2);
           route(i,j,k).serviceD=route(i,j,k).serviceD+Y^(1/2);
        end
        route(i,j,k).lastR=Reta; %the last node to be served in one route
        Dist=(node(Reta).loc(1))^2+(node(Reta).loc(2))^2;
        route(i,j,k).travelD=route(i,j,k).travelD+Dist^(1/2);
        chro(i,j).distanceCost=chro(i,j).distanceCost+route(i,j,k).travelD;
        %begin to calculate the time related cost
        route(i,j,k).timeCost=0;
        Ret=route(i,j,k).ret;     %Ret: indicate the nodes in one route
        for m=1:length(Ret)
           earCost=earlyCost(node(Ret(m)).alpha, node(Ret(m)).S, route(i,j,k).T,
                    route(i,j,k).L, node(Ret(m)).d, node(Ret(m)).sigma,
                    node(Ret(m)).safe, 10); %calculate early cost for one node
           [lCost penalty laCost]=lateCost(node(Ret(m)).belta, node(Ret(m)).gama,
                                    node(Ret(m)).S, route(i,j,k).T, route(i,j,k).L,
                                    node(Ret(m)).d, node(Ret(m)).sigma,
                                    node(Ret(m)).safe, 10);  %calculate late cost for one
                                    node

           tCost=earCost+laCost;     %calculate total time cost for one node
           route(i,j,k).timeCost=route(i,j,k).timeCost+tCost;
        end
        chro(i,j).timeCost=chro(i,j).timeCost+route(i,j,k).timeCost;
     end

      chro(i,j).fitness=chro(i,j).timeCost+5*chro(i,j).distanceCost+2000*chro(i,j).nofro
      ute;          totalCost(i,1)=totalCost(i,1)+chro(i,j).fitness;
  end
end

%next generation begins here
for generation=2:20
```

```
for i=1:4
   for j=1:10
      %begin to excecute time related mutation
      TCost=[];
      for k=1:chro(i,j).nofroute
         TCost=[TCost route(i,j,chro(i,j).routes(k)).timeCost];
      end
      [max delRoutel]=max(TCost);            %find the largest time related cost route
      Ret=route(i,j,chro(i,j).routes(delRoutel)).ret; %node array of this route
      timeDiff=[];
      for m=1:length(Ret)
         timeD=(node(Ret(m)).T-route(i,j,chro(i,j).routes(delRoutel)).T)^2;
         timeDiff=[timeDiff timeD];
      end
      [timeMax nodel]=max(timeDiff);         %find the node with largest time difference
      oriRoute=route(i,j,chro(i,j).routes(delRoutel));
      TimeDelRoute=deleteNode(oriRoute, nodel);
      for k=1:chro(i,j).nofroute
         timeRoute(i,j,chro(i,j).routes(k))=route(i,j,chro(i,j).routes(k));
      end
      timeRoute(i,j,chro(i,j).routes(delRoutel))=TimeDelRoute; %the mutated route
      Ret=route(i,j,chro(i,j).routes(delRoutel)).ret;
      delNode=node(Ret(nodel));              %get the node to be deleted
      Routes=chro(i,j).routes;
      while (~isempty(Routes))               %try to add this node to another route
         timeDif=[];
         for k=1:length(Routes)
            timeDi=(delNode.T-timeRoute(i,j,Routes(k)).T)^2;
            timeDif=[timeDif timeDi];
         end
         [min addRoutel]=min(timeDif);   %find the route with minimum time difference
         if delNode.Q<=timeRoute(i,j,Routes(addRoutel)).Cleft  %if the capacity is
                                                                  enough
           oriRoute=timeRoute(i,j,Routes(addRoutel));
            timeRoute(i,j,Routes(addRoutel))=addNode(oriRoute, Ret(nodel));
           break
         else
           Routes(addRoutel)=[];
        end
      end

      %begin to execute the distance related mutation

      SerD=[];
      for k=1:chro(i,j).nofroute
```

```
    SerD=[SerD timeRoute(i,j,chro(i,j).routes(k)).serviceD];
end
[maxSD delRouteI]=max(SerD); %find the route with largest service travel
                                        distance
oriRoute=timeRoute(i,j,chro(i,j).routes(delRouteI)); %the route to be mutated
nodeInd=oriRoute.lastR;  %the node to be deleted
for m=1:length(oriRoute.ret)
   if nodeInd==oriRoute.ret(m)
      nodeI=m;        %get the index of the node to be deleted
      break
   end
end
DistDelRoute=deleteNode(oriRoute, nodeI);  %delete this node
for k=1:chro(i,j).nofroute
   distRoute(i,j,chro(i,j).routes(k))=timeRoute(i,j,chro(i,j).routes(k));
end
distRoute(i,j,chro(i,j).routes(delRouteI))=DistDelRoute;
Routes=chro(i,j).routes;
while (~isempty(Routes))       %search where to add the deleted node
   averDist=[];
   for m=1:length(Routes)     %find the route closest to this node
      nRDist=0;
      Ret=distRoute(i,j,Routes(m)).ret;
      for n=1:length(Ret)
         nRDist=nRDist+dis(nodeInd, Ret(n));
      end
      nRDist=nRDist/length(Ret);
      averDist=[averDist nRDist];
   end
   [minDist addRouteI]=min(averDist);  %find the closest route
   if node(nodeInd).Q<=distRoute(i,j,Routes(addRouteI)).Cleft
     oriRoute=distRoute(i,j,Routes(addRouteI));
     distRoute(i,j,Routes(addRouteI))=addNode(oriRoute, nodeInd);
     break
   else
     Routes(addRouteI)=[];  %otherwise, delete this route, and continue to search
   end
end

%begin the merge mutation

capaLeft=[];
for k=1:chro(i,j).nofroute
   capaLeft=[capaLeft distRoute(i,j,chro(i,j).routes(k)).Cleft]; %array for left
                                                                    capacity
```

```
        end
        [max delRoutel]=max(capaLeft); %find the route with largest left capacity
            Routes=chro(i,j).routes;
            Routes(delRoutel)=[];
            capaLeft(delRoutel)=[];
            Ret=distRoute(i,j,chro(i,j).routes(delRoutel)).ret;
            oriRoutes=[];
            nodeInds=[];
            while ~isempty(Ret)
                find=0;
                for k=1:length(capaLeft)
                    if node(Ret(1)).Q<=capaLeft(k)  %find the route with enough capacity
                        capaLeft(k)=capaLeft(k)-node(Ret(1)).Q;
                        oriRoutes=[oriRoutes Routes(k)];  %remember the route
                        nodeInds=[nodeInds Ret(1)];       %remember the node
                        find=1;
                        break
                    end
                end
                if find==0
                    break
                end
                Ret(1)=[];    %delete this node, continue to test next node
            end
            if length(oriRoutes)==length(distRoute(i,j,chro(i,j).routes(delRoutel)).ret)
                for k=1:length(oriRoutes)
                    tranRoute=addNode(distRoute(i,j,oriRoutes(k)), nodeInds(k));
                    distRoute(i,j,oriRoutes(k))=tranRoute;     %add this node to the route
                end
                mutChro(i,j).routes=chro(i,j).routes;
                mutChro(i,j).routes(delRoutel)=[];
                mutChro(i,j).nofroute=chro(i,j).nofroute-1;
            else
                mutChro(i,j).routes=chro(i,j).routes;
                mutChro(i,j).nofroute=chro(i,j).nofroute;
            end
    end
end

for i=1:4
    totalCost(i,generation)=0
    for j=1:10
        mutChro(i,j).distanceCost=0;
        mutChro(i,j).timeCost=0;
        mutChro(i,j).fitness=0;
```

```
        for k=1:mutChro(i,j).nofroute

            mutChro(i,j).distanceCost=mutChro(i,j).distanceCost+distRoute(i,j,mutChro(i,j)
            .routes(k)).travelD;

            mutChro(i,j).timeCost=mutChro(i,j).timeCost+distRoute(i,j,mutChro(i,j).routes(
            k)).timeCost;
        end

         mutChro(i,j).fitness=mutChro(i,j).timeCost+5*mutChro(i,j).distanceCost+2000*mu
         tChro(i,j).nofroute;
        totalCost(i,generation)=totalCost(i,generation)+mutChro(i,j).fitness;
    end
end

for i=1:4
    for j=1:10
        newChro(i,j).routes=chro(i,j).routes;
        newChro(i,j).nofroute=chro(i,j).nofroute;
        newChro(i,j).distanceCost=chro(i,j).distanceCost;
        newChro(i,j).timeCost=chro(i,j).timeCost;
        newChro(i,j).fitness=chro(i,j).fitness;
    end
end

for i=1:4
    chroPoolFit=[];
    chroPool=[];
    for j=1:10
        chroPoolFit=[chroPoolFit newChro(i,j).fitness];
        chroPool=[chroPool newChro(i,j)];
        for k=1:newChro(i,j).nofroute
            poolRoutes((2*j-1),k)=route(i,j,newChro(i,j).routes(k));
        end
        chroPoolFit=[chroPoolFit mutChro(i,j).fitness];
        chroPool=[chroPool mutChro(i,j)];
        for k=1:mutChro(i,j).nofroute
            poolRoutes((2*j),k)=distRoute(i,j,mutChro(i,j).routes(k));
        end
    end
    for m=1:10
        [maxCost l]=max(chroPoolFit);
        chroPoolFit(l)=[];
        chroPool(l)=[];
        poolRoutes(l,:)=[];
```

```
    end
    for j=1:10
        chro(i,j).routes=chroPool(j).routes;
        chro(i,j).nofroute=chroPool(j).nofroute;
        chro(i,j).distanceCost=chroPool(j).distanceCost;
        chro(i,j).timeCost=chroPool(j).timeCost;
        chro(i,j).fitness=chroPool(j).fitness;
        for k=1:chro(i,j).nofroute
            route(i,j,k)=poolRoutes(j,k);
        end
    end

end

end %for generation

x=1:15;
y=totalCost(4, x);
plot(x,y)
```

Appendix C Main Functions in the Application of Combinatorial Petri Net Model

```
function petriNet=initialization(placeSize, transitionSize)
%initialization for a Petri net

petriNet.clock=0;
petriNet.incidenceMatrix=zeros(transitionSize, 2*placeSize);
petriNet.maxTime=100;
petriNet.placeSize=placeSize;
petriNet.transitionSize=transitionSize;
petriNet.eventQueue=[];
petriNet.randArray=randn(1, 1000);

function [ele, petriNet]=node(type, n, init, PN)
%Define an element for a Petri net model

petriNet=PN;
switch type
    case 'place'
        ele=init;
    case 'transition'
        ele=init;
        for i=1:length(ele.inputPlaces)
            petriNet.incidenceMatrix(n, ele.inputPlaces(i))=1;
        end
        for i=1:length(ele.outputPlaces)
            petriNet.incidenceMatrix(n, (petriNet.placeSize+ele.outputPlaces(i)))=1;
        end

     otherwise
        error('invalide element')
end

function [places, petriNet]=enabling(P, T, PN)
%checking enabling transitions, and put them into event queue.
petriNet=PN;
places=P;
TRUE=1; FALSE=0;
for i=1:petriNet.transitionSize
   T(i).enabled=FALSE;
   T(i).tokenEnabled=TRUE;
   if length(T(i).inputPlaces)==0
      if T(i).abl(1)==1
         T(i).enabled=TRUE;
```

```
        end
    else %if length(T(i).inputPlaces)==0
        for j=1:length(T(i).inputPlaces)
            if isempty(P(T(i).inputPlaces(j)).index)
                T(i).tokenEnabled=FALSE;
                break
            end
        end %for k=1:length(T(i).inputPlaces)
        if T(i).tokenEnabled==TRUE
            if T(i).abl(1)==1
                T(i).enabled=TRUE;
            else %if T(i).abl(1)==1
                m=i;
                T(i).abl(2)=ABL(P, T, m)
                if T(i).abl(2)~=0
                    T(i).enabled=TRUE;
                end
            end %if T(i).abl(1)==1
        end %if T(i).tokenEnabled==TRUE
    end %if length(T(i).inputPlaces)==0
    if T(i).enabled==TRUE
        if length(T(i).delay)==3
            Delay=T(i).delay(1)+T(i).delay(2)*petriNet.randArray(T(i).delay(3));
            T(i).delay(3)=T(i).delay(3)+1;
        else
            Delay=T(i).delay(1);
        end
        T(i).fireTime=petriNet.clock+Delay;
        PN=petriNet;
        petriNet.eventQueue=add_event(T(i), PN.eventQueue);
        for j=1:length(T(i).inputPlaces)
            t=i;
            p=T(i).inputPlaces(j);
            PN=petriNet;
            places=arc_expressions(t, p, P, PN);
            P=places;
        end
    end %if T(i).enabled==TRUE
end %for i=1:petriNet.transitionSize

function eventQueue=add_event(event, EQ)
%add an event into the event queue

if isempty(EQ)
    eventQueue=[event];
```

```
else
   skip=0;
   if event.fireTime<=EQ(1).fireTime
       eventQueue=[event EQ];
   else
      for j=2:length(EQ)
         if event.fireTime<=EQ(j).fireTime
            eventQueue=[EQ(1:(j-1)) event EQ(j:length(EQ))]
            skip=1;
            break
         end
      end
      if skip==0
      eventQueue=[EQ event];
      end
   end
end

function [places, petriNet]=firing(P, T, PN)
%fire all events with zero delay and one event with non-zero delay
petriNet=PN;
places=P;
delete=0;
if ~isempty(petriNet.eventQueue)
   for i=1:length(petriNet.eventQueue)
      %transition=petriNet.eventQueue(i);
      t=petriNet.eventQueue(i-delete).number
      for j=1:length(petriNet.eventQueue(i-delete).outputPlaces)
         p=petriNet.eventQueue(i-delete).outputPlaces(j);
         places=arc_expressions(t, p, P, PN);
         P=places;
      end
      petriNet.clock=petriNet.eventQueue(i-delete).fireTime;
      Delay=petriNet.eventQueue(i-delete).delay;
      petriNet.eventQueue(i-delete)=[];
      delete=delete+1;
      if Delay>0
         break
      end
   end
end
```

Appendix D Program in Determining the Configuration of a Distribution Chain

max
272u1+383u2+380u3+509u4+484u5+918u6+528u7+578u8+852u9+688u10+331u11+
512u12+403u13+725u14+713u15+725u16+872u17+470u18+397u19+524u20-41w11-
26w12-17w13-25w14-12w15-31w16-5w17-26w18-48w19-140w110-50w111-66w112-
48w113-67w114-57w115-90w116-57w117-36w118-42w119-58w120-53w21-57w22-
55w23-61w24-74w25-61w26-58w27-45w28-119w29-74w210-43w211-47w212-
38w213-37w214-33w215-15w216-30w217-21w218-15w219-34w31-57w32-62w33-
72w34-94w35-110w36-84w37-89w38-168w39-135w310-14w311-13w312-24w313-
37w314-65w315-74w316-97w317-66w318-62w319-58w320-810v1-890v2-980v3-
336v1-370v2-437v3-500

ST
-0.035u1-0.044u2-0.039u3-0.059u4-0.063u5-0.079u6-0.05u7-0.055u8-0.074u9-
0.065u10-0.036u11-0.048u12-0.044u13-0.06u14-0.069u15-0.077u16-0.074u17-
0.044u18-0.042u19-0.051u20>-0.8

68u1+85u2+76u3+113u4+121u5+153u6+96u7+105u8+142u9+125u10+69u11+93u12
+84u13+115u14+132u15+148u16+143u17+84u18+81u19+97u20>1000

-0.068w11-0.085w12-0.076w13-0.113w14-0.121w15-0.153w16-0.096w17-0.105w18-
0.142w19-0.125w110-0.069w111-0.093w112-0.084w113-0.115w114-0.132w115-
0.148w116-0.143w117-0.084w118-0.081w119-0.097w120>-0.8

-0.062w21-0.077w22-0.069w23-0.103w24-0.11w25-0.139w26-0.087w27-0.095w28-
0.129w29-0.114w210-0.063w211-0.085w212-0.076w213-0.105w214-0.12w215-
0.135w216-0.13w217-0.076w218-0.074w219-0.088w220>-0.8

-0.052w31-0.065w32-0.058w33-0.087w34-0.093w35-0.118w36-0.074w37-0.081w38-
0.109w39-0.096w310-0.053w311-0.072w312-0.065w313-0.088w314-0.102w315-
0.114w316-0.11w317-0.065w318-0.062w319-0.075w320>-0.8

68w11+85w12+76w13+113w14+121w15+153w16+96w17+105w18+142w19+
125w110+69w111+93w112+84w113+115w114+132w115+148w116+143w117+84w11
8+81w119+97w120>500

68w21+85w22+76w23+113w24+121w25+153w26+96w27+105w28+142w29+125w21
0+69w211+93w212+84w213+115w214+132w215+148w216+143w217+84w218+81w2
19+97w220>500

68w31+85w32+76w33+113w34+121w35+153w36+96w37+105w38+142w39+125w31
0+69w311+93w312+84w313+115w314+132w315+148w316+143w317+84w318+81w3
19+97w320>500

```
w11+w21+w31=1
w12+w22+w32=1
w13+w23+w33=1
w14+w24+w34=1
w15+w25+w35=1
w16+w26+w36=1
w17+w27+w37=1
w18+w28+w38=1
w19+w29+w39=1
w110+w210+w310=1
w111+w211+w311=1
w112+w212+w312=1
w113+w213+w313=1
w114+w214+w314=1
w115+w215+w315=1
w116+w216+w316=1
w117+w217+w317=1
w118+w218+w318=1
w119+w219+w319=1
w120+w220+w320=1
end

int u1
int u2
int u3
int u4
int u5
int u6
int u7
int u8
int u9
int u10
int u11
int u12
int u13
int u14
int u15
int u16
int u17
int u18
int u19
int u20
int v1
int v2
```

```
int v3
int w11
int w12
int w13
int w14
int w15
int w16
int w17
int w18
int w19
int w110
int w111
int w112
int w113
int w114
int w115
int w116
int w117
int w118
int w119
int w120
int w21
int w22
int w23
int w24
int w25
int w26
int w27
int w28
int w29
int w210
int w211
int w212
int w213
int w214
int w215
int w216
int w217
int w218
int w219
int w220
int w31
int w32
int w33
int w34
```

```
int w35
int w36
int w37
int w38
int w39
int w310
int w311
int w312
int w313
int w314
int w315
int w316
int w317
int w318
int w319
int w320
```