Design and Implementation of the Consumer Buying Process for Mobile AGent-based Internet Commerce System (MAGICS)

by Perry Pui Yi Lam

A Thesis submitted in Partial Fulfillment of the Requirements of the Degree of Master of Philosophy in Department of Computing

The Hong Kong Polytechnic University

October 2004

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

____

Lam Pui Yi Perry

ABSTRACT

Abstract of thesis entitled 'Design and Implementation of the Consumer Buying Process for Mobile AGent-based Internet Commerce System (MAGICS)' submitted by Perry Pui Yi Lam for the degree of Master of Philosophy at the Hong Kong Polytechnic University in October 2004.

With the advent of software agent technologies, there has been considerable interest in the development of agent-based e-commerce systems. Agents can make a computer system more effective because they can perform tasks autonomously. To complement current Web-based Internet commerce systems and particularly to support consumer-oriented e-commerce, we have developed a mobile agent-based e-commerce system called Business-to-Consumer (B2C) Mobile AGent-based Internet Commerce System (MAGICS). This project focuses on using B2C MAGICS to facilitate the consumer buying process.

Consumers can use B2C MAGICS through a proxy server via a Web/WAP interface. In accordance with user requirements, agents are sent to obtain responses from sellers. The best seller is chosen based on these responses and an agent is sent to carry out the transaction. To complete these tasks, the agents need to exchange many messages. In this thesis, we develop an Extensible Markup Language (XML) scheme for inter-agent communications in which agents communicate through MAGICS messages which are specified by an XML schema. A MAGICS message consists of three kinds of information: basic, functional, and additional. It is possible to use this framework to define messages that support the agent-based consumer buying process.

We also investigate a product-comparison problem. When an agent visits a list of shops sequentially to perform search and evaluation, the decision whether to buy the item or continue shopping will entail comparisons. We consider two kinds of comparison, price and multi-attribute. In the case of price comparison, we formulate two Markov-decision-based price comparison models. This model uses a backward induction algorithm which determines the optimal decision policy. This allows the agent to buy a product at the minimum expected cost, including the traveling cost. The model is analyzed using a normal price distribution and real price information and simulations are conducted to validate the analytical results. For the case of multi-attribute comparison, we propose a Fuzzy Markov Decision Process (fMDP) for Multi-attribute Product Comparison, which is extended from the price comparison models. The fMDP model handles multiple attributes evaluation using a Fuzzy Logic algorithm. This produces an optimal decision policy that an agent can use to buy the most suitable product. Analytical results are presented to study the behavior of the system with different parameters. Furthermore, experiments are carried out to compare the performance of the fMDP model and the performance of one of the price comparison models.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deeply thankfulness to my supervisor, Dr Henry C. B. Chan of the Department of Computing at the Hong Kong Polytechnic University. Without his help, this work would not be possible. During my Master’s study, Dr Chan has given me patient guidance, encouragement and excellent advice. His insightful comments on my career have also been invaluable to me. Special thanks also go to my co-supervisor Dr Raymond Lee who greatly enriched my knowledge with his extraordinary insights. His discussion was helpful in formulating the analytical model in Chapter 6.

I would like to express my gratitude for my tuition scholarships from Research Postgraduate Studies for financing my study. This project was supported by a Competitive Earmarked Research Grant from the Research Grants Council of the Hong Kong Special Administrative Region, China under grant number PolyU 5078/01E. Preliminary research on MAGICS and its basic framework was supported by a small grant from The Hong Kong Polytechnic University under account number G-T176.

I would like to thank Mr. Benjamin Lam, Mr. Chris Chin and Mr. Gary Li for establishing the basis for my work, particularly on the development of the prototype system.

Last but not least, I am indebted to all my colleagues in the Department of Computing and friends who have supported me throughout the years. I take this opportunity to express my profound gratitude to my beloved parents and brother for their moral support and patience during my study. I hope all of them have a healthy life and brilliant future.

TABLE OF CONTENTS

ABSTRACT .... **i**

ACKNOWLEDGEMENTS .... **iii**

CHAPTER 1 INTRODUCTION **1**

1.1. Motivation 1

1.2. Scope & Objectives 3

1.3. Organization 4

CHAPTER 2 BACKGROUND STUDY **6**

2.1. E-commerce 6

2.1.1. Business-To-Consumer (B2C) E-commerce 7

2.1.2. Business-To-Business (B2B) E-commerce 8

2.1.3. Consumer-To-Consumer (C2C) E-commerce 9

2.1.4. Consumer-To-Business (C2B) E-commerce 10

2.2. Mobile Agent Technology 11

2.2.1. Agent's life cycle 12

2.2.2. Benefits for using mobile agents 13

2.2.3. Mobile agent design patterns 15

2.3. Current Agent-based E-commerce Systems 18

2.3.1. B2C: Kasbah 18

2.3.2. B2B: MAgNET 21

2.3.3. C2C: Nomad 24

2.3.4. Comparison between web-based and agent-based e-commerce 26

2.4. Consumer Buying Process 26

2.4.1. Recognize needs 27

2.4.2. Product brokering 27

2.4.3. Merchant brokering 29

2.4.4. Negotiate and purchase 30

2.4.5. Carry out post-purchase activities 31

CHAPTER 3 MAGICS OVERVIEW 32

3.1. Architecture of MAGICS 32

3.2. B2C MAGICS 33

3.2.1. System Architecture 33

3.2.2. Workflow 34

3.2.3. Implementation 39

CHAPTER 4 COMMUNICATION FRAMEWORK FOR B2C MAGICS 48

4.1. Agent Communication Language 48

4.2. XML-based Communication Scheme 50

4.3. Example of a Simple Buying Process 54

4.4. Summary 62

CHAPTER 5 MARKOV-DECISION-BASED PRICE COMPARISON MODEL 63

5.1. General Markov-decision-based Model Formulation 65

5.2. The Price Comparison Problem 66

5.2.1. Model I 69

5.2.2. Model II 73

5.3. Results and Discussion 75

5.3.1. Analytical Results 75

5.3.2. Experimental Results 86

5.3.3. Discussions 88

5.4. Summary 91

CHAPTER 6 FUZZY MARKOV DECISION PROCESS (fMDP) FOR MULTI- ATTRIBUTE PRODUCT COMPARISON 93

6.1. Concept of Fuzzy Logic 94

6.2. fMDP Formulation 95

6.3. Analytical Results and Discussions.. 102

6.4. Comparison with Model II 108

6.5. Summary 111

CHAPTER 7 CONCLUSION 113

REFERENCES...116

LIST OF FIGURES

Figure 1 The workflow of B2C E-commerce (Amazon).......7

Figure 2 The workflow of B2B E-commerce (GXS).......8

Figure 3 The workflow of C2C E-commerce (e-Bay).......9

Figure 4 The workflow of C2B E-commerce (Priceline).......10

Figure 5 Agent's life cycle.......12

Figure 6 Agents dispatched in parallel.......17

Figure 7 Agents dispatched in a sequential manner.......17

Figure 8 Kasbah workflow and different negotiation strategies of selling and buying agents....19

Figure 9 MAgNET workflow.......22

Figure 10 Nomad architecture.......24

Figure 11 The consumer buying process.......27

Figure 12 The framework of MAGICS.......32

Figure 13 B2C MAGICS Architecture.......34

Figure 14 The buying process in B2C MAGICS.......36

Figure 15 Linear satisfaction function under two different cases.......39

Figure 16 The components of Aglet Object Model.......40

Figure 17 Accessing MAGICS via a WAP interface (Nokia 7210 simulator).......44

Figure 18 Compare the offers returned by the master buying agent.......44

Figure 19 Transferring the digital book to the buying agent.......45

Figure 20 The inputted information for the second demonstration.......46

Figure 21 The recommended shops for the second demonstration.......46

Figure 22 The partly containment of MAGICS element.......50

Figure 23 The structure of the general product and particular products.......53

Figure 24 An XML schema for a MAGICS message.......54

Figure 25 Quotation request message.......55

Figure 26 Quotation response message.......56

Figure 27 Quotation response message with an error.......57

Figure 28 Negotiation request message 58
Figure 29 Negotiation response message 59
Figure 30 Buying request message........ 60
Figure 31 Buying response message 61
Figure 32 The general Markov-decision-based model........ 65
Figure 33 Flow diagram of Model I 67
Figure 34 Flow diagram of Model II........ 68
Figure 35 Preferred choices for different values of C_{travel} ($\sigma = 0.1\mu$) 76
Figure 36 Average cost vs. number of shops (Model I) 77
Figure 37 Preferred choices for different values of m with $\sigma = 0.1\mu$, $C_{travel} = 0.001\mu$ (Model II)78
Figure 38 Average cost vs. number of shops (Model II) 80
Figure 39 The average cost for different numbers of shops of purchasing the book "High-Speed Signal Propagation: Advanced Black Magic" under different price distribution and different traveling costs, C_{travel} (prices based on the information at "http://www.pricescan.com") 81
Figure 40 The % difference between normal and real price distribution (Model I, Analytical).. 82
Figure 41 The % difference between normal and real price distribution (Model II, Analytical) 83
Figure 42 Average cost vs. no. of shops (3 models, Analytical)........ 84
Figure 43 The % difference between normal and real price distribution (Model I, Simulation). 87
Figure 44 The % difference between normal and real price distribution (Model II, Simulation)87
Figure 45 Average cost vs. no. of shops (3 models, Simulation) 88
Figure 46 The price threshold using different μ and σ (Model I)........ 89
Figure 47 The price threshold using different σ and C_{travel} (Model I) 89
Figure 48 The threshold table with different m, σ and C_{travel} (Model II) 90
Figure 49 The general fuzzy system........ 94
Figure 50 Architecture of Fuzzy Markov Decision Process (fMDP) for Multi-attribute Product Comparison........ 97
Figure 51 Fuzzy membership functions for size 98
Figure 52 Preference score distribution 99

Figure 53 The transition probability 102

Figure 54 Preferred choices for each preference categories with C_{travel} = 0.01 103

Figure 55 Overall preferred choices with different C_{travel} 105

Figure 56 Average product score (O_{avg}) vs. number of shops with different C_{travel} 106

Figure 57 Overall preferred choices with different β of the preference distribution 107

Figure 58 Average product score (O_{avg}) vs. number of shops with different β of the preference distribution 107

Figure 59 The comparison between fMDP and Model II with high mean (μ) value 111

Figure 60 The comparison between fMDP and Model II with low mean (μ) value 111

LIST OF TABLES

Table 1 Attributes and their values for the second demonstration....47

Table 2 Average score of each shop for the second demonstration....47

Table 3 Notations used in the price-comparison models....68

Table 4 Values of the parameters used in the example....71

Table 5 Transition probabilities of the example....71

Table 6 Expected costs and preferred choices (Model I)....73

Table 7 Expected costs and preferred choices (Model II)....75

Table 8 The computer books found from priceSCAN.com (http://www.pricescan.com)....81

Table 9 Notations used in the fMDP model....96

CHAPTER 1 INTRODUCTION

In this chapter, we present an overview of the research project and the thesis, including the motivation, the objectives and the organization of the thesis.

1.1. Motivation

Electronic commerce (e-commerce) includes all commercial transactions conducted via an electronic medium. Internet-based shopping or business-to-consumer (B2C) e-commerce has many advantages [20][39] but can be time consuming. In an online store, all the products are categorized and consumers can browse for product information in an electronic catalog. When consumers cannot find an item, they can provide the requirement of the item and look for it by using the search function and this will return a list of results which the consumer can click on and scan to consider and evaluate. Most consumers also like to compare product attributes before making a purchase decision. During the evaluation phase, they may find it difficult to compare or evaluate products. It will certainly take time and effort and finding the best offer may require visits to several stores. Automation of this shopping process would make it easier and more efficient. This automation can be achieved using software agents (for simplicity hereafter called agents), which can perform tasks autonomously and act intelligently [18][26]. Indeed, it is expected that agents will ultimately complement existing Web-based e-commerce systems [18][41][46] and play a variety of mediating roles in next-generation consumer-oriented e-commerce systems [30].

In recent years, research on agent technology for supporting e-commerce has attracted considerable interest, with many agent-based e-commerce systems having been proposed and implemented [9][11][33]. To contribute to the emerging research area of agent-based e-commerce, we have developed an e-commerce system, B2C Mobile AGent-based Internet Commerce System (MAGICS), particularly for the support of consumer-oriented e-commerce. B2C MAGICS complements the current Web-based Internet commerce system with mobile agents, supporting the consumer buying process by automating time-consuming tasks as a way of reducing the user workload. In this thesis, we study the architecture and protocol of B2C MAGICS. A prototype has been developed to show the basic functions. We also propose an XML-based inter-agent communications framework.

Mobile agents can compare or evaluate products at Internet-based stores without any human intervention. However, there are many virtual stores on the Internet, so it is essential to address the issue of how to send agents to visit this large number of virtual stores effectively and efficiently. In this thesis, we propose two Markov-decision-based models for solving a price-comparison problem for B2C MAGICS. We also investigate the evaluation of products with fuzzy attributes. A fuzzy Markov-decision process is formulated for this purpose.

1.2. Scope & Objectives

The aim of this research project is to design and implement the architecture and protocol to support a consumer-oriented mobile agent-based Internet commerce system. We focus on investigating how agents can communicate effectively in order to process a sales transaction for three categories of goods: standard goods (e.g., books), non-standard goods (e.g., dresses), and customizable goods (e.g., computers).

For the purposes of comparing standard and customizable goods, we use two Markov-decision-based models to obtain an optimal decision policy that would allow a product to be bought at the lowest expected cost, including the traveling costs. To permit the comparison of non-standard goods whose attributes cannot be defined precisely, we have also used the Markov Decision theory and Fuzzy Logic algorithm to formulate a product comparison problem. The solution of this problem identifies a product that best fits the requirements of a customer. Moreover, we have proposed a communication architecture based on XML in order to facilitate agent communications.

In summary, the objectives of the research project are to:

- design the architecture and protocol for supporting a consumer-oriented mobile agent-based Internet commerce system called B2C MAGICS.
- define an XML-based communication scheme for B2C MAGICS.
- investigate a price-comparison problem for B2C MAGICS to buy standard goods.

- study a fuzzy Markov decision process for B2C MAGICS to buy non-standard goods.

1.3. Organization

The following is the organization of the remaining chapters of this thesis:

Chapter 2: Background Study

Chapter 2 provides an overview of different types of e-commerce. It also describes the agent technology and the advantages of using agents. Then, various agent-based e-commerce systems are introduced. It finally examines the major phases of a consumer buying process and how mobile agent technology can facilitate the process.

Chapter 3: MAGICS Overview

Chapter 3 describes the architecture and protocol of B2C MAGICS. MAGICS is a mobile agent-based system designed to facilitate business-to-customer (B2C), business-to-business (B2B) and customer-to-customer (C2C) e-commerce with mobile agents. This project focuses on the B2C system. In essence, B2C MAGICS uses mobile agents to assist the consumer buying process.

Chapter 4: Communication Framework for B2C MAGICS

Chapter 4 proposes the B2C MAGICS communication framework. Generally, agents in MAGICS communicate with each other through MAGICS messages, which are defined by XML schema. Examples are presented to illustrate the communication method.

Chapter 5: Markov-decision-based Price Comparison Model

Chapter 5 explains the use of the Markov Decision theory to formulate a price comparison problem. The objective of the problem is to obtain an optimal decision policy in order for an agent to buy a product at the lowest expected cost, including traveling costs. The problem is solved using the backward induction algorithm. The models are analyzed based on a normal price distribution and real price information. Simulation results are presented to compare different models.

Chapter 6: Fuzzy Markov Decision Process (*f*MDP) for Multi-attribute Product Comparison

As an extension of Chapter 5, we formulate a Fuzzy Markov-decision-based model for evaluating multiple decision attributes for non-standard goods. The aim is to select the offer that can best satisfy a consumer’s requirements. As attributes for non-standard goods cannot be specified precisely, a fuzzy logic algorithm is used for the evaluation of each attribute. Analytical results are presented and discussed. Experiments are carried out to compare the performance of the *f*MDP model with the Markov-decision-based model described in the preceding chapter.

Chapter 7: Conclusion

This chapter concludes the thesis.

CHAPTER 2
BACKGROUND STUDY

In the following sections, different types of e-commerce are presented and the mobile agent technology is introduced. By combining e-commerce with mobile agents, various current agent-based e-commerce systems are illustrated briefly. Finally we explain how the several phases of consumer buying process can be facilitated by mobile agents.

2.1.E-commerce

In [20], Kalakota and Whinston define electronic commerce (e-commerce) from various perspectives. Generally speaking, e-commerce involves all the commercial transactions over a communication network, particularly the Internet. These transactions include products or services trading, information exchange, payment mechanisms, back-office management, and so on. As e-commerce is usually associated with sales transactions, which involve buyers and sellers, it is commonly categorized into two major types: consumer-oriented and business-oriented. Based on the relationship between buyers and sellers, it can be further divided into four categories: business-to-consumer (B2C), business-to-business (B2B), consumer-to-consumer (C2C) and consumer-to-business (C2B) [8][39].

2.1.1. Business-To-Consumer (B2C) E-commerce

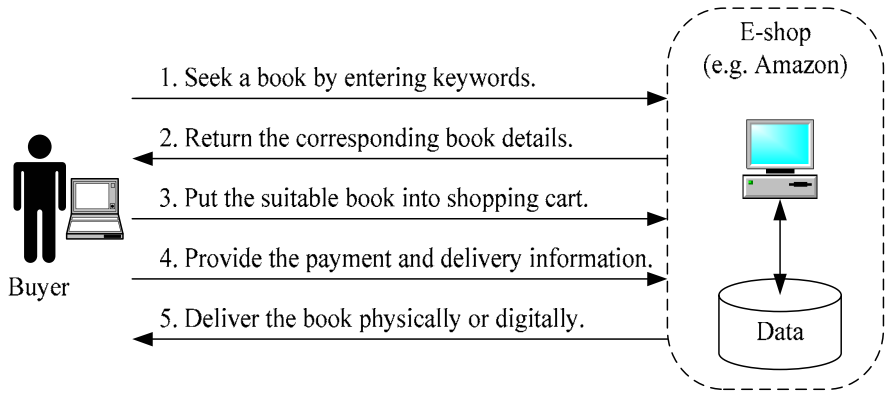

E-shop
(e.g. Amazon)
1. Seek a book by entering keywords.
2. Return the corresponding book details.
3. Put the suitable book into shopping cart.
4. Provide the payment and delivery information.
5. Deliver the book physically or digitally.
Buyer
Data

Figure 1 The workflow of B2C E-commerce (Amazon)

B2C e-commerce is the selling of products or services from businesses to consumers through an electronic medium. In a physical store, a consumer must search for the goods by asking the salesperson. In contrast, consumers can find the required products more easily in an electronic store by browsing an electronic catalog or searching with a search engine. The most popular instance of B2C e-commerce is Amazon.com (http://www.amazon.com), a virtual store in which a buyer can buy standard goods such as books, and music discs. To illustrate the operations in Amazon.com clearly, let us take an example of buying a book (see Figure 1). When a consumer visits Amazon.com, he/she seeks a book by selecting the product type and entering keywords in the search box. Next, he/she puts a suitable book into the shopping cart and then, if required, chooses the next book. The ordering process is completed by providing the payment and delivery information. After completing a transaction, the buyer can rate the books he/she has bought. This rating service provides a useful reference for other users. Amazon.com also provides other special services, particularly for its registered consumers. Its recommendations service makes use of data mining and intelligent computing technologies to suggest books based on the consumer's interests. This is done by classifying products the consumer has previously purchased and the buying behavior of other consumers. Amazon's notification service reminds consumers of special occasions. Another service alerts a consumer when previously unavailable goods become available. Amazon.com also offers many other services and products (e.g., gift certificates).

2.1.2. Business-To-Business (B2B) E-commerce

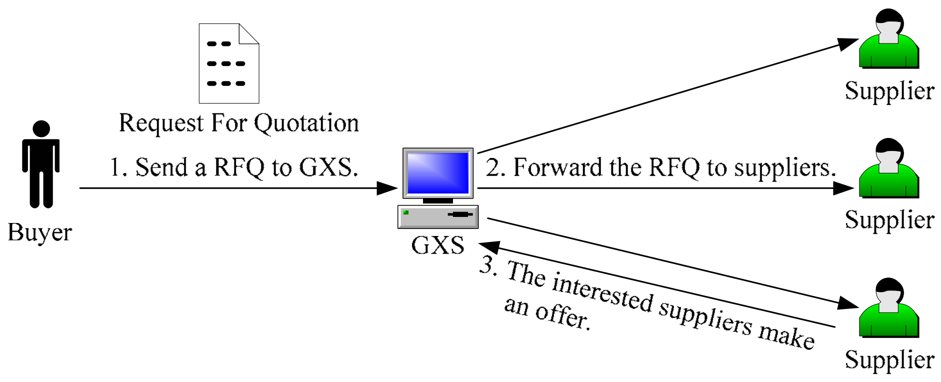

Request For Quotation
1. Send a RFQ to GXS.
Buyer
GXS
2. Forward the RFQ to suppliers.
3. The interested suppliers make an offer.
Supplier
Supplier
Supplier

Figure 2 The workflow of B2B E-commerce (GXS)

B2B e-commerce facilitates the business-to-business transactions. In many situations, this is concerned with procuring raw materials for making products. In general, there are three basic B2B e-commerce models: buy-side e-commerce, sell-side e-commerce, and the electronic marketplace [39]. In the buy-side e-commerce model, a company which buys a large number of different items opens a marketplace for other companies to sell the items. In the sell-side e-commerce model, a company does all the selling to a large number of small and large organizations. In the electronic marketplace model, a marketplace is set up for multiple buyers and sellers to trade with each other. As shown in Figure 2, Global eXchange Service (GXS) (http://www.gxs.com) is an example of B2B e-commerce. GXS provides an online supply chain solution. The basic procurement process is as follows. First, GXS receives from a buyer a Request For Quotation (RFQ) which contains the requirements of the goods. This RFQ is forwarded to potential suppliers around the world. Interested suppliers respond to the RFQ with an offer that is returned to GXS.

2.1.3. Consumer-To-Consumer (C2C) E-commerce

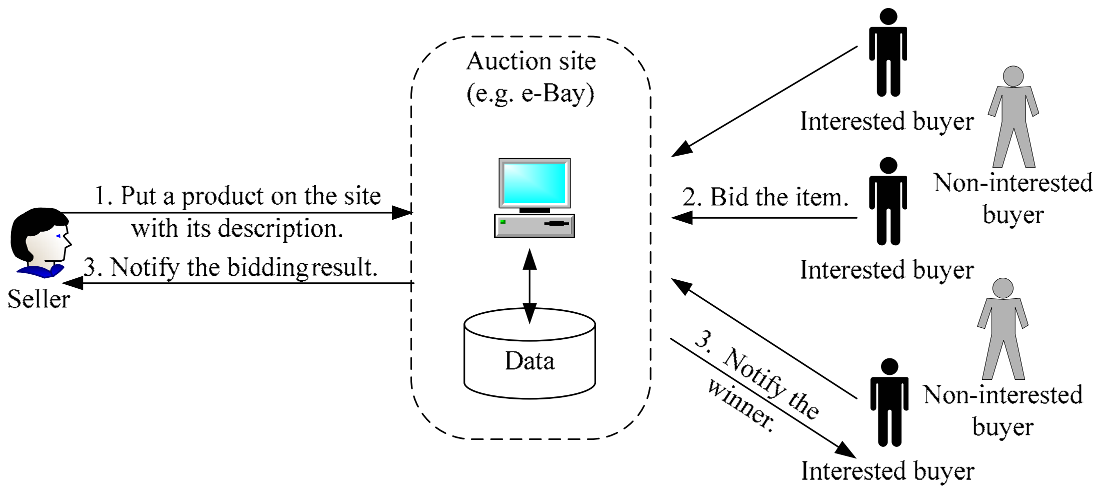


Figure 3 The workflow of C2C E-commerce (e-Bay)

A C2C e-commerce allows consumers to sell goods to the other consumers directly in the Internet environment. Traditionally, individuals who wish to sell items privately have done so through classified advertisements. Nowadays, the Internet auction model is the most popular C2C e-commerce system. Internet auction is a bidding process, which can take a long time. eBay (http://www.ebay.com) is a well-known Internet auction site. It provides a marketplace for a seller to post an item for bidding and for a buyer to bid for the interested item. Before using the system, all the buyers and sellers have to register as members. As shown in the workflow in Figure 3, a seller puts a product on the site by providing its description. Other interested buyers can bid for the item before the deadline. In most cases, eBay uses the English auction, meaning that the highest bidder wins. At the end of the auction, eBay notifies the seller and the winner of the result through e-mail. eBay also offers information for bidders when they are considering whether to submit a bid. This information includes the product description, bidding history, and the seller's rating. A buyer uses the product details to search for a suitable item. Based on the bidding history, a bidder knows the bidding information (e.g., how the bids change with time). The rating is graded by buyers who have completed auction transactions with that seller. The grade is useful for helping a user to assess whether the seller is trustworthy.

2.1.4. Consumer-To-Business (C2B) E-commerce

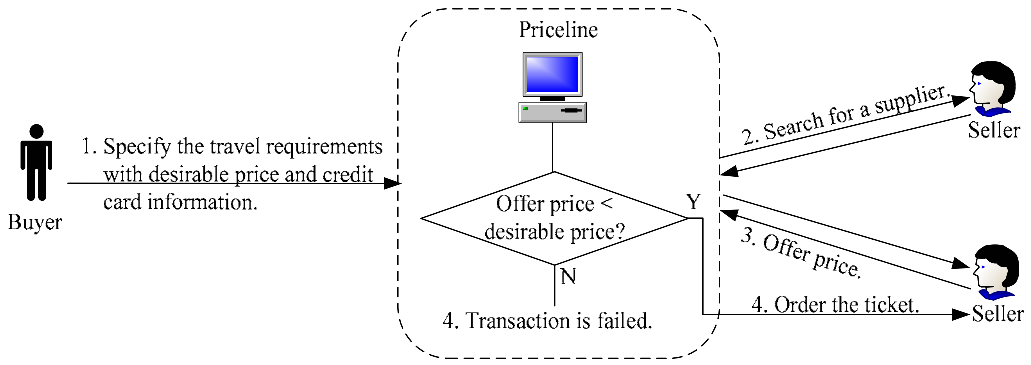

Priceline
Buyer
1. Specify the travel requirements with desirable price and credit card information.
Offer price < desirable price?
Y
N
4. Transaction is failed.
2. Search for a supplier.
Seller
3. Offer price.
4. Order the ticket.
Seller

Figure 4 The workflow of C2B E-commerce (Priceline)

C2B e-commerce is based on a buyer-centric model in which a consumer defines the requirements and a business provides a tailor-made product. Unlike B2C e-commerce, sellers have to find out what buyers want, thus it is highly customized. Priceline.com (http://www.priceline.com) is a representative example of C2B e-commerce. Figure 4 shows the basic operation of Priceline for supporting C2B e-commerce. When a user wants to buy an airline ticket, he/she has to specify the travel requirements such as the departure and arrival date and city, number of tickets, and so on. Furthermore, he/she has to give Priceline the desired price for each ticket and his/her credit card number. Priceline then searches for a supplier who can satisfy the user requirements. If there is a supplier who can give an offer lower than the user expected, Priceline orders the ticket with the user’s credit card and retains the price difference. The important rule at Priceline is that a buyer cannot object or demand a refund once a supplier is found. Apart from selling airline tickets, Priceline also sells other services, including hotel rooms, rental cars, vacation packages, and cruises.

2.2. Mobile Agent Technology

In general, the term *agent* can be defined as a program that can perform a series of operations on behalf of a person. These operations are processed autonomously even though that person is not connected to the network. An agent can also learn, cooperate, control, and make decisions in its life cycle. The life cycle includes agent creation/disposal, agent movement, and agent communications as explained later [27].

A *mobile agent* is a software agent which is not bound to its initial execution system. It can move over the network with its execution states [37]. In contrast, an agent that cannot move is called a *stationary agent*. Both stationary and mobile agents can also communicate with other agents within the same system or in different systems by using remote procedure calling and message passing.

2.2.1. Agent’s life cycle

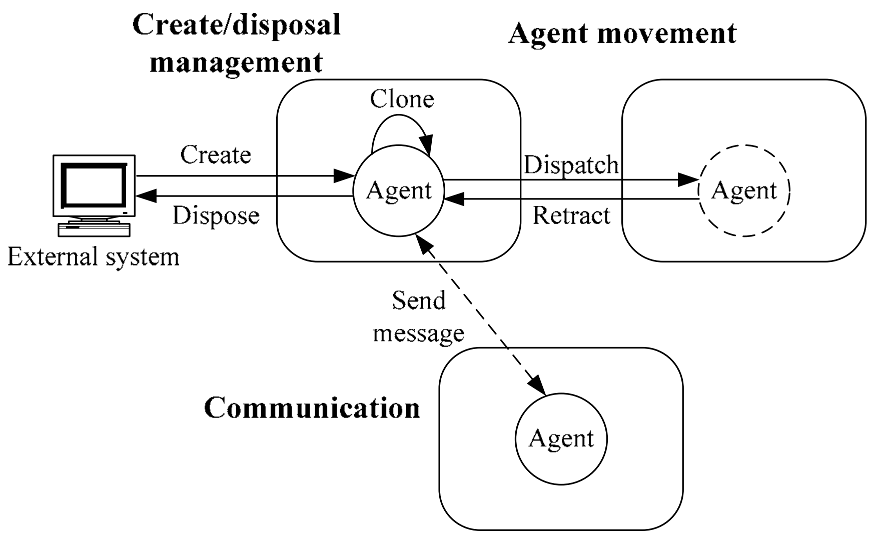


Figure 5 Agent's life cycle

Figure 5 shows the main events in the life cycle of an agent. The descriptions are as follows:

1. Creation/disposal management

Creation and disposal are carried out through, respectively, the constructor and destructor of an object. Both operations can be initialized by other agents in the same place or by an agent-based or a non-agent-based system in other places. When an agent is created, the agent object is instantiated and assigned a unique identifier. Then it is initialized by itself with a set of arguments and waits for further instructions. Apart from those times when it is assigned by others, the disposal process may also be activated when the agent expires, if no one uses it for a long time, if security rules are violated, or when the system shuts down. Before disposal, an agent completes the current tasks, releases the resources being used, and prepares for disposal.

2. Agent movement

Like disposal, agent movement can be invoked explicitly by the agent itself or triggered by another agent-based and a non-agent-based system. The movement includes sending the agent from the current place to the remote site and retracting it back to the origin. When the agent arrives at a new location, it will perform its duties. If a transfer fails, the user will be alerted.

3. Communication

An agent can handle messages passed from agents residing within the same place or from agents residing in other places. An agent can also invoke the authorized methods of other agents by sending a message. Miscommunication can lead to executing a wrong method or processing incorrect data. Consequently, communication plays an important role in an agent-based system.

2.2.2. Benefits for using mobile agents

A mobile agent can move across a system/network. As stated in [17][27], there are seven benefits of using mobile agents. These benefits are summarized as follows:

1. Reduce the network load

Mobile agents can be dispatched to a destination host to carry out the necessary interactions “locally”. Thus, they can reduce data flow across a remote host. For example, consider that a user wants to analyze a large amount of data in a remote site. A lot of bandwidth is consumed if data is copied to his/her site. If a mobile agent is used, it can carry the required method and move to the remote site where it can analyze the date before returning with the result, thus saving the bandwidth.

2. Overcome network latency

Large network latency is unacceptable in certain critical real-time systems. Mobile agents can reduce latency because they can be dispatched from a central controller to a remote host. They can then execute the controller’s instructions directly at the remote site.

3. Encapsulate protocols

In a distributed network, each host normally carries the protocol to properly process outgoing data and incoming data. When the protocol is updated, it would be time-consuming to upgrade the code/protocol on every host. Mobile agents solve this problem since the protocols are encapsulated in them and they can move to any remote host.

4. Execute asynchronously and autonomously

Mobile agents save the network resources because they do not require a persistent network connection. When a mobile agent is dispatched to a remote site, it performs the task automatically. At the same time, it disconnects from the originating device. Later, the device can reconnect to the mobile agent and obtain the results.

5. Adapt dynamically

Mobile agents have the ability to dynamically adapt to changes in their execution environment. For example, a group of mobile agents can distribute themselves among the hosts to achieve maximum efficiency.

6. Naturally heterogeneous

Mobile agents are independent of computer and transport layer and dependent solely on the execution environment. This feature enables them to work in a network with heterogeneous systems.

7. Robust and fault tolerant

Mobile agents have the capability to react dynamically to unfavorable conditions and events. For instance, if a host is being shut down, the mobile agents in that host will be alerted. They will then be dispatched to other hosts and continue their work.

2.2.3. Mobile agent design patterns

Design patterns help developers to capture solutions to common problems in agent-based systems and achieve good applications design through reusability of validated components. The design patterns are classified into three groups, traveling, task, and interaction patterns [27]. Traveling patterns are used to handle the movement of mobile agents. Task patterns are used to design the way that tasks are delegated to agents. Interaction patterns are used to locate agents and facilitate their interactions. Most systems are designed with more than one design pattern. The following section mainly focuses on integrating the master-slave and itinerary patterns.

A master-slave pattern is a kind of task pattern. It involves two types of agent, a master and a slave. A master agent creates a slave agent and then delegates a task to the slave agent. Next, the slave agent is dispatched to a remote destination to perform the task. Sharing tasks between two agents saves time. An itinerary pattern is a traveling pattern. It defines the trip of an agent to multiple destinations. The agent performs its pre-set task after arrival and then dispatches itself to the next destination in the itinerary. It can be given different types of itineraries without having to modify its own code.

When combining two patterns in a system, slave agents are created by a master agent and they travel to other sites according to the itinerary. The system has the advantage of the master agent being stationary and therefore able to centrally receive tasks from everywhere. Master agent may dispatch slave agents either in parallel or in sequence. One way is to dispatch them in parallel. The process is shown as follows (see Figure 6):

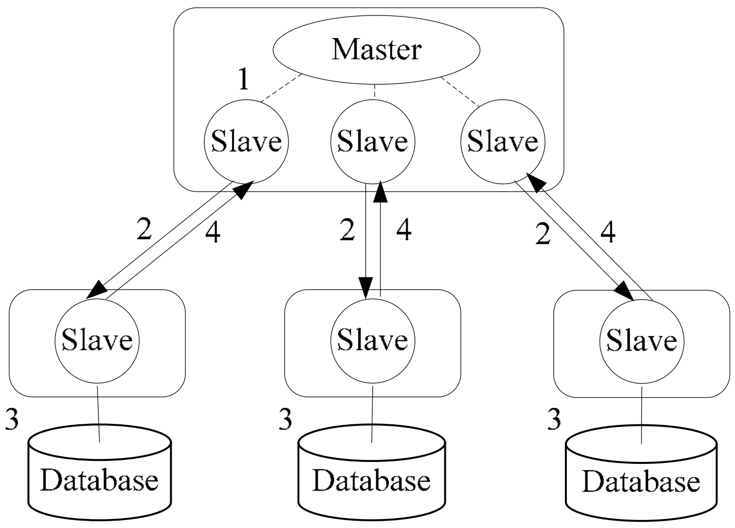


Figure 6 Agents dispatched in parallel

1. A master agent creates several slave agents.
2. All slave agents are dispatched to different remote hosts in parallel.
3. A slave agent executes its task.
4. A slave agent collects the result and sends it back to the master agent.

The process of sequential dispatch is as follows (see Figure 7):

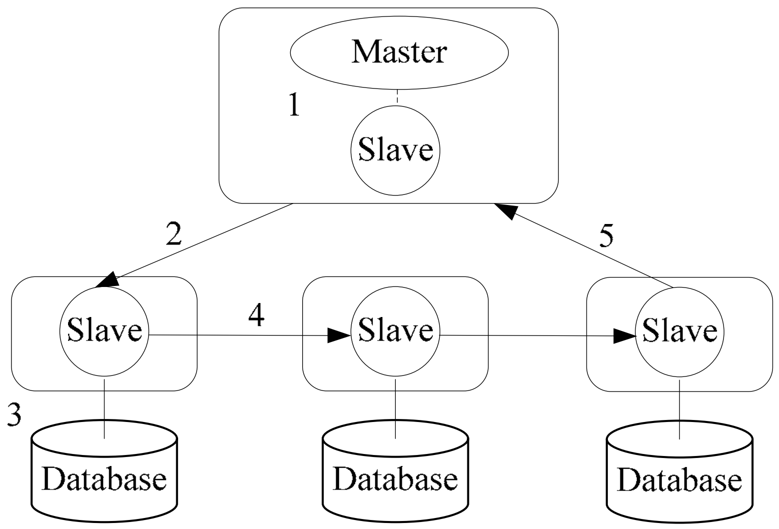


Figure 7 Agents dispatched in a sequential manner

1. A master agent creates one slave agent.

2. The slave agent is dispatched to the first destination in the itinerary.
3. The slave agent executes its task.
4. The slave agent travels to the next destination and performs the task. This process continues until the final destination is reached.
5. After completing the task in the final destination, the slave agent returns to the master agent with all the results.

2.3. Current Agent-based E-commerce Systems

In recent years, there has been significant interest in using agent e-commerce technology to create completely automated services. A number of agent-based e-commerce systems have been developed. The following presents some typical agent-based e-commerce systems for each type of e-commerce.

2.3.1. B2C: Kasbah

Kasbah [9] is a typical agent-based B2C e-commerce system. It is a digital/electronic agent marketplace where agents can carry out trades on behalf of consumers. Besides providing buy and sell operations, negotiation of price is supported in the Kasbah system.

Figure 8 illustrates the buying/selling process in Kasbah. When a user wants to sell items, he/she registers the goods to be sold in the system. Then a selling agent is created. The agent carries the description of the items it sells. Acting proactively, it goes to the marketplace to find interested buying agents. Similarly, a consumer creates a buying agent and provides it with purchase requirements. Then the buying agent is dispatched to the marketplace to interact with selling agents. The buying agents and selling agents negotiate with each other using different price-time functions to find the best deal.

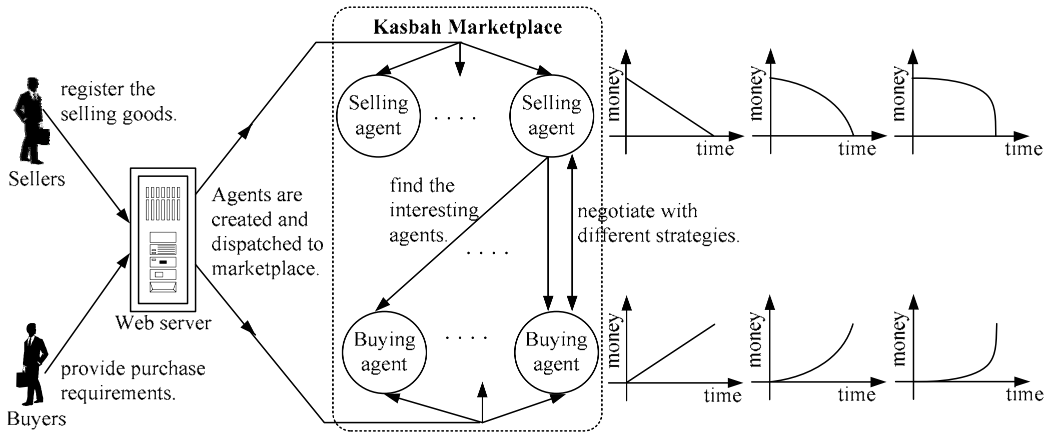

Kasbah Marketplace
register the selling goods.
Sellers
Web server
provide purchase requirements.
Buyers
Agents are created and dispatched to marketplace.
Selling agent
Selling agent
find the interesting agents.
negotiate with different strategies.
Buying agent
Buying agent
money
time

Figure 8 Kasbah workflow and different negotiation strategies of selling and buying agents

As agents can act autonomously without user intervention, users have to set several parameters before creating the agents. These parameters are specified to describe the behavior of agents. Actually, a selling agent and a buying agent have a similar structure. Let us take a selling agent as an example to explain how to set the parameters. The first parameter is the description of the item for sale. This parameter is used to make a match with the product requested by other buying agents. The second parameter is the desired price of the item for sale. It is the selling price at which the seller wants to sell. The third parameter is the lowest acceptable price. This is the minimum selling price that the seller is willing to accept. The fourth parameter is the desired date by which the item should be sold, the deadline for selling the item. In other words, the product cannot be sold after the desired date. The final parameter is the negotiation strategy, the approach used by the selling agent when negotiating with the buying agents. The parameters used by a buying agent are nearly the same as those described above except that it uses a buying operation and sets a maximum acceptable price.

The selling agent firstly sells the item at the desired price. If there is a buying agent willing to buy the item, the transaction is completed. Otherwise, the selling price is decreased over a given time frame according to the negotiation strategy. When the desired date arrives and no buying agent buys the item, the selling price will be decreased to the lowest acceptable price. If the selling agent can make the deal at a certain price, it sends a message to the seller to ask him/her whether the item can be sold at that price. That means the seller can make the final decision before an agreement is reached with the buying agent.

In Kasbah, the negotiation is straightforward. If a buying agent and selling agent are matched, one agent makes an offer and the other one replies with either an “accept” or a “reject” message. The selling agent lowers the selling price using a mathematical function based upon price/time (see Figure 8). Possible functions are linear, quadratic and cubic. The linear strategy decreases the price to the lowest acceptable price very quickly. The cubic strategy lowers the price slowly at the beginning but decreases the price significantly towards the desired date. The rate of decrease of the quadratic strategy falls between that of the linear and cubic strategies.

2.3.2. B2B: MAgNET

MAgNET [11] (Mobile Agents for Networked Electronic Trading) is a B2B agent-based system. It is used for networked electronic trading. Implemented with Java mobile agent technology, it supports product brokering, merchant brokering, negotiation and product reservation. It can also be employed to support a hierarchical supply chain.

The MAgNET system uses the push model of marketing in which buyers send agents to suppliers in order to find suitable products. There are several agents involved in the system including a buyer's stationary agent, a buyer's mobile agent, a buyer's surrogate agents and a supplier's stationary agent. The stationary agent for the buyer is used to capture user requests through a GUI and an Open Buying on the Internet (OBI) interface. It creates the mobile agents, manages communications between mobile agents and surrogate agents, and updates information in the database according to those messages. The buyer's mobile agent is used to ask for quotations from a supplier's stationary agent. It makes a reservation if the quotation is attractive. If the supplier's stationary agent needs extra time to quote, the mobile agent creates a temporary surrogate agent to wait. The surrogate agent also waits for an instruction from the buyer's stationary agent to confirm the purchase or cancel the reservation. The stationary agent for the supplier's is used to interact with the buyer's mobile agent. It records the interactions in a transaction log. It also obtains a quotation from the production planning software when the requested item is not in the current inventory.

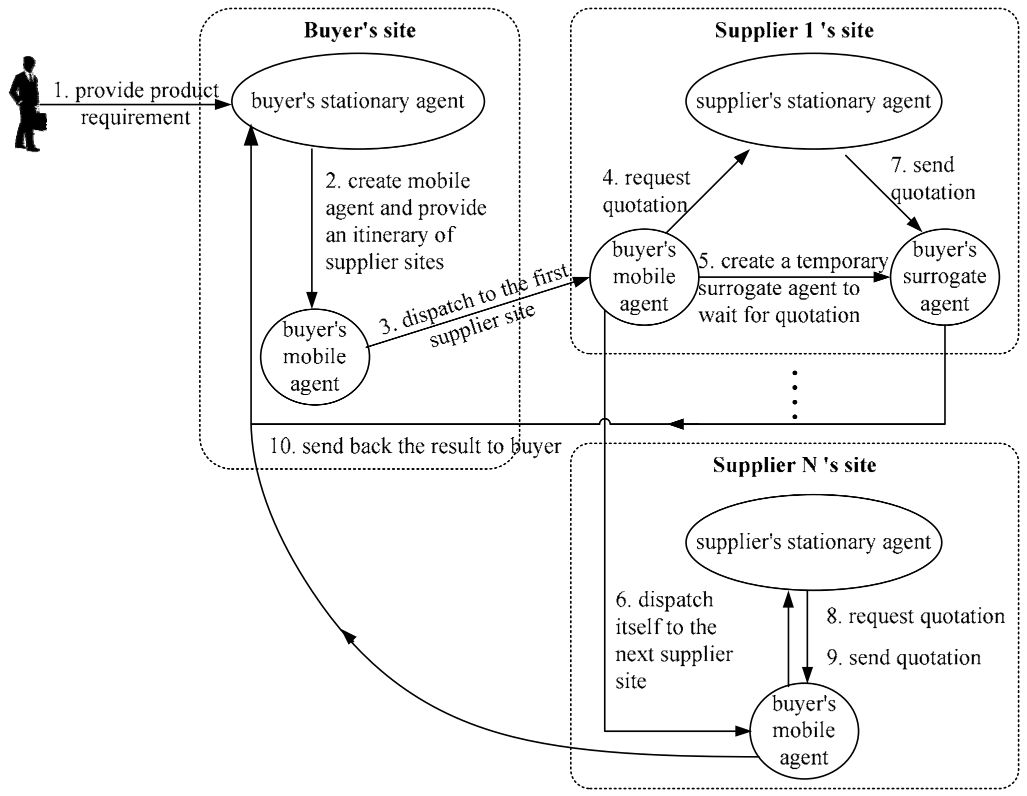

Buyer's site
Supplier 1 's site
1. provide product requirement
buyer's stationary agent
supplier's stationary agent
2. create mobile agent and provide an itinerary of supplier sites
4. request quotation
7. send quotation
buyer's mobile agent
5. create a temporary surrogate agent to wait for quotation
buyer's surrogate agent
3. dispatch to the first supplier site
buyer's mobile agent
10. send back the result to buyer
Supplier N 's site
supplier's stationary agent
6. dispatch itself to the next supplier site
8. request quotation
9. send quotation
buyer's mobile agent

Figure 9 MAgNET workflow

Figure 9 shows how the agents work in the system. When a human buyer wants to buy an item, he/she sends the product requirement in the form of a directed acyclic graph (DAG) to the stationary agent in the buyer's site through the GUI. Then the buyer's stationary agent traverses the DAG depth-first and inserts information into an assignment table which stores the instructions for the buyer's mobile agents. MAgNET system provides an itinerary of supplier sites. The buyer's stationary agent creates a mobile agent and dispatches it to the first supplier site on the itinerary list. When the mobile agent arrives at a supplier site, it sends a message to the supplier's stationary agent in order to request a quotation. If the current supplier provides a better offer than the recorded offer, the mobile agent makes a reservation for that item and updates the best offer record. Then the mobile agent creates a surrogate agent to wait for instructions from the buyer’s stationary agent to confirm or cancel the reservation. If the supplier’s stationary agent requires more time to quote, the mobile agent also leaves a surrogate agent for the pending request before moving to the next supplier site. After visiting all the supplier sites in the itinerary list, the mobile agent returns to the buyer site. The buyer’s stationary agent compares the results from the mobile and surrogate agents. Next it sends messages to the surrogate agents in different supplier sites to confirm the purchase or to cancel the reservation. If the surrogate agent cancels a reservation, it has to pay the reservation fee.

When a supplier does not have enough stock, it may need to contact its suppliers. A complication arises if several suppliers ask the same supplier when ordering the raw materials. MAgNET can handle this deep supply chain by a codeword strategy. A buyer sends a request to a supplier with a codeword which contains two parts. The first part represents a buyer and the second part represents a potential purchase. When the supplier has insufficient stock, he/she asks for more stock from another supplier using a new codeword which is the combination the buyer codeword and his/her own codeword (e.g., the new codeword is buyerA-10, supplierB-20). By comparing the two sets of codewords, the supplier can check whether there is a conflict. A conflict occurs if the first codeword in each set is different or if the first difference occurs in the second part of a codeword. In other words, the first case means that two transactions are raised by different buyers, while the second case means that two transactions are raised by the same buyer for different purposes.

2.3.3. C2C: Nomad

Nomad [33] is a C2C mobile agent-based system for an Internet-based auction house, e-AuctionHouse. Through Nomad, a mobile agent can be created as an auction participant. The agent then travels to an auction service provider and participates in an auction on the user's behalf. Nomad supports two agent types, one programmed by users and the other created from its template agent library.

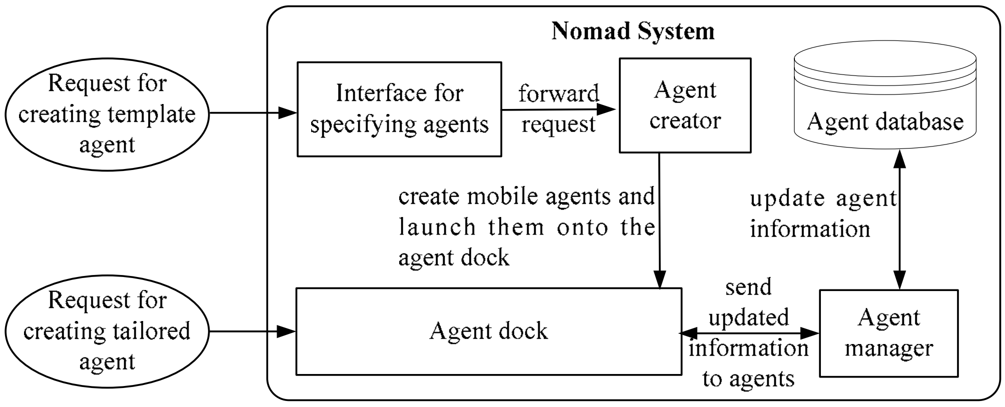

Nomad System
Request for creating template agent
Interface for specifying agents
forward request
Agent creator
Agent database
create mobile agents and launch them onto the agent dock
update agent information
Request for creating tailored agent
Agent dock
send updated information to agents
Agent manager

Figure 10 Nomad architecture

The Nomad architecture is shown in Figure 10. It consists of four main components, an interface for specifying agents, an agent dock, an agent manager, and an agent database. The interface receives a request for generating agents from a user. Upon receiving a request, it forwards the request to the agent generator. According to the user's requirements, the agent generator creates mobile agents and launches them onto the agent dock, which is a place for mobile agents to process auctions. Nomad uses the Concordia agent dock because Concordia supports mobile agents written in Java and provides mobile agents with a safe execution platform. The agent manager sends a message to inform the agents when updated information becomes available about products of interest. The agent database stores the agent details, such as its creator and products of interest.

Nomad supports user-programmed Java agent. Users can take advantage of this function to design their own agents with tailored bidding strategies. They can also program the agents by transmitting a text message to the auction server. In addition, Nomad provides an interface for non-programmers to create mobile agents. Firstly, users choose the agent type according to what they want their agents to do. Secondly, they customize the agents by setting required parameters. Then the corresponding agents are created by the agent generator and travel to the agent dock. There are five kinds of preprogrammed template agents. The information agent is used to notify a user by email of a specified event. The incrementor agent is used for English auctions. It keeps bidding until reaching the user's reservation price. The N-agent is used for single-item, single-unit, sealed-bid first-price auctions. It bids on behalf of the user according to a specific instruction. The control agent is used to mislead other bidding agents by artificially increasing the number of bidders. The discover agent is used to assist a user to determine the bids.

Nomad is good for novice bidders. Even though the novices may not know much about the bidding strategies, the configurable mobile agents can help them to bid on their own behalves. It also allows agents to work individually even though the user is offline. This reduces network traffic. By using mobile agents, it is possible to realize quicker responses to changes in an auction and faster computational times.

2.3.4. Comparison between web-based and agent-based e-commerce

In this section we compare the web-based and agent-based e-commerce from the perspective of a buying process. A consumer using web-based e-commerce first searches for an item through many virtual stores via a single interface. He/she secondly evaluates the product with the assistance of supplementary services. Finally, the payment is completed by using an electronic channel. In the agent-based e-commerce, all the buying jobs can be delegated to agents. They perform the search in parallel or in a sequential manner, and run more intelligent evaluation algorithms to reach a better decision. They then finish the transaction on behalf of their owners. Thus, agents can save time, work as a human and even do something that a human user does not know how to do.

2.4. Consumer Buying Process

Agent technology can be applied to support different types of e-commerce. Our research focuses on consumer-oriented e-commerce which addresses the fundamental consumer buying process in which people can buy products, use products, and make purchase decisions. The consumer buying process consists of five phases (based on [18][21]), shown in Figure 11.

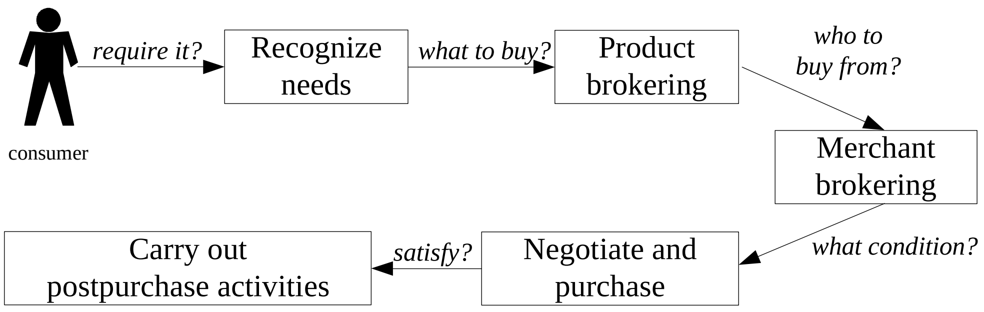


Figure 11 The consumer buying process

2.4.1. Recognize needs

In this initial phase, a consumer identifies a need for products or services and defines his/her requirements. When the consumer has an unfulfilled need, the consumer buying process will be activated. The needs can be stimulated by internal or external stimuli such as, hunger and advertisements respectively. Agent technology can be used in this phase. For example, in [29], an agent called *notification agent* can act as a classified advertisement. A user sends the agent his/her profile and then the agent can constantly seek advertisements of interest. Finally, it informs the user if a suitable product is available in the market. Therefore, the user can get the latest items of interest from the agent.

2.4.2. Product brokering

In the second phase, a consumer determines what product to buy in order to satisfy the defined need. He/She can obtain a set of products by internal and external searches. The internal search is based on buyer's previous experience. The external search includes personal sources, public sources, and marketer-dominated sources. Because of the mobile ability of agents, agents can travel anywhere, making them suitable for use in product brokering. Agents can also make recommendations based on the buyer’s behavior. These agents are commonly called *recommendation agents* [29]. The technologies used by recommendation agents to search for appropriate products are content-based, collaborative-based, or constraint-based filtering methods [18].

Content-based filtering selects products if the features extracted from their contents match features explicitly stated in the users’ requirements. The performance of using this technique fluctuates because there are no standard inputs and presentation methods in different sources. Collaborative-based filtering recommends products based on the ratings and feedback from different users who have similar preferences. For example, an agent in Firefly [2] identifies shoppers with tastes similar to those of the user and then recommends products that the shoppers prefer. Constraint-based filtering chooses products according to the relevance of their features. The selected products should meet the constraints to be considered, and be within the specified scale. Price is an example of such a constraint.

The knowledge of consumers affects the extent of an information search. If the user clearly knows what he/she needs, the content-based technique is better. If the user has only a vague notion of his/her needs, the constraint-based technique is preferred. Otherwise, the collaborative-based technique is the most suitable technique.

2.4.3. Merchant brokering

In the third phase, a consumer determines who to buy the product from and specifies the criteria for evaluation. Along with price, warranty and delivery time can be the evaluation factors for the consumer. He/she compares products based on the evaluation criteria and determines an evaluation set comprising items that the consumer considers acceptable. Because there can be many alternatives to compare, *comparison shopping agents* [29] are good for supporting this process. They can judge the products by calculating scores with different weightings for different attributes. With the use of intelligent agent technology, the evaluation process can be performed more effectively.

The comparison shopping process can be described as taking place in three steps. When comparing two or more items, differentiation agents first differentiate the items based on their attributes. Secondly, evaluation agents evaluate the differences and form the aggregated utility information. Finally, a decision can be made based on user preferences by preference agents [41].

The differentiation agents have three kinds of "living mechanism". The first one is living independently with vendors. This agent acts as a mediator and is neutral when giving differentiation information. The second one has a formal partnership with vendors. Both parties benefit because an agent obtains production information without blocking and the agent puts the vendors in a more favorable competitive position. The third one is embedded in an individual vendor. It only works for implicit differentiation and involves small dimensions of differentiation. For the evaluation agents, quantified evaluation information is provided by differentiating the product in a unidimensional or multidimensional way. The unidimensional way is rarely used except when the demand for information is limited by the channel capacity or customer need. The preference agents are used to collect scenario information which is based on the experience of online shoppers. Moreover, the peer experiences are considered if the shopper’s own experience is not appropriate. Sometimes, the shoppers have trade-off difficulties among products. In this case, the situational analysis provides useful information for them to make decisions.

2.4.4. Negotiate and purchase

In the fourth phase, a consumer determines the conditions under which the order can be completed. After negotiating the conditions with a seller, the consumer pays for the product upon reaching an agreement with the seller. Traditionally, a human user negotiates face-to-face. This, however, is relatively time-consuming and inconvenient. In an agent-based system, negotiation can be conducted by *negotiation agents* [29]. Essentially, the negotiation agents offer bids on behalf of their owners in order to obtain the maximum benefit. Apart from providing automated services, they can ensure consistency and accuracy.

In an automated negotiation, two agents are involved (i.e., buyer and seller agents). A buyer agent is responsible for specifying purchasing needs, criteria and personal preferences. On the other hand, a seller agent concerns with the sale of products and pricing rules [29].

It is impossible to find a general strategy for all negotiation cases. Negotiation agents need to use different strategies for different contexts. According to [18], these negotiation strategies are divided into three types. The first strategy is described as “win/loss negotiation”. In this case, the agents decide what offer they should give based on the opponent’s behavior. The second strategy is described as “win/win negotiation”. In this situation, the agents look for a win-win solution for both parties. This means that both parties win something upon completing the negotiation. The third strategy is “argumentative negotiation”. In this case, the agents exchange arguments to justify their positions. The arguments also include information explaining why an offer is unacceptable.

2.4.5. Carry out post-purchase activities

In the final phase, a consumer conducts post-purchase activities such as maintenance, service, and evaluation of satisfaction with the buying decision. The consumer compares the purchased product with his/her expectation and finds support from others to confirm his/her choice. If the consumer is satisfied with the post-purchase activities provided by the seller, he/she may carry out repeat-purchases from the seller. This phase is important for sellers because “dissatisfied buyers complain to nine people but satisfied buyers only notify three people” [35]. However, to the best of our knowledge, few agent-based systems support this phase.

CHAPTER 3
MAGICS OVERVIEW

The above-mentioned agent-based systems can bring in new services and also complement current Web-based e-commerce applications. To contribute to the emerging research area of agent-based e-commerce, we are developing the Mobile Agent-based Internet Commerce System (MAGICS) [5][6][7] at The Hong Kong Polytechnic University. It facilitates B2C, C2C and B2B e-commerce with mobile agent technology. In the following, the general architecture of MAGICS is introduced. Then, the B2C system is discussed to support the consumer buying process.

3.1. Architecture of MAGICS

MAGICS Applications: for providing various e-commerce applications
MAGICS Protocols: for enabling inter-agent communications
MAGICS Agents: for providing component-based agents
Agent Programming Interface (e.g., Java Aglet): for providing the programming

Figure 12 The framework of MAGICS

MAGICS is based on a distributed framework following the Web System. Figure 12 shows the four-layer model of MAGICS. The first layer is Agent Programming Interface. In MAGICS, IBM Java Aglet [1][27] is used to develop stationary and mobile agents. The second layer provides different kinds of component-based agents with the use of API. The third layer is MAGICS protocols which prepare the means for agents to communicate with each other. Making use of the above layers, various MAGICS applications can be built.

3.2. B2C MAGICS

In essence, B2C MAGICS uses mobile agents to assist the consumer buying process. Users can use the service by providing product requirements via a Web or WAP interface. The system then creates a mobile agent to complete the transaction on behalf of the user.

3.2.1. System Architecture

Figure 13 shows the major components of the proposed system. The interface part allows a consumer to access the system. Users can communicate with MAGICS via a Web browser or a mobile device. User requests are then sent to the proxy server through HTTP. After processing the request, the result is returned from the proxy server to the interface. The proxy server is used to handle the HTTP requests and generate mobile agents for non-MAGICS enabled clients (i.e., for clients that can neither create mobile agents nor support the MAGICS protocol). In other words, this proxy server interfaces between the clients and the merchants. The broker provides the locations of the merchants, which provide the product that users want, for the mobile agents. Working in conjunction with a database, the merchants' servers serve the mobile agents. The supporting systems provide other essential services such as payment services.

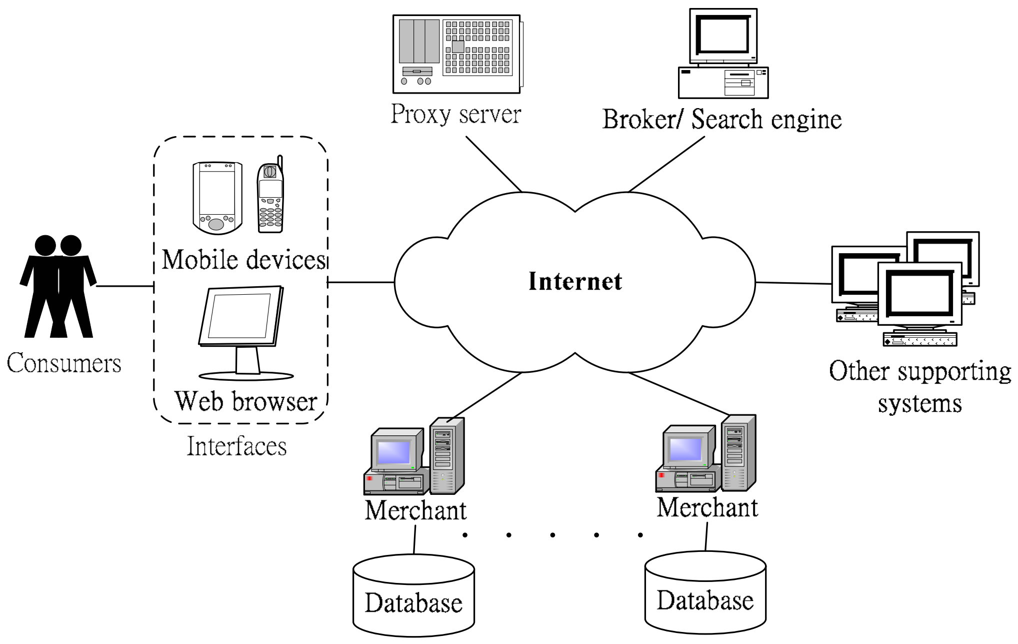


Figure 13 B2C MAGICS Architecture

3.2.2. Workflow

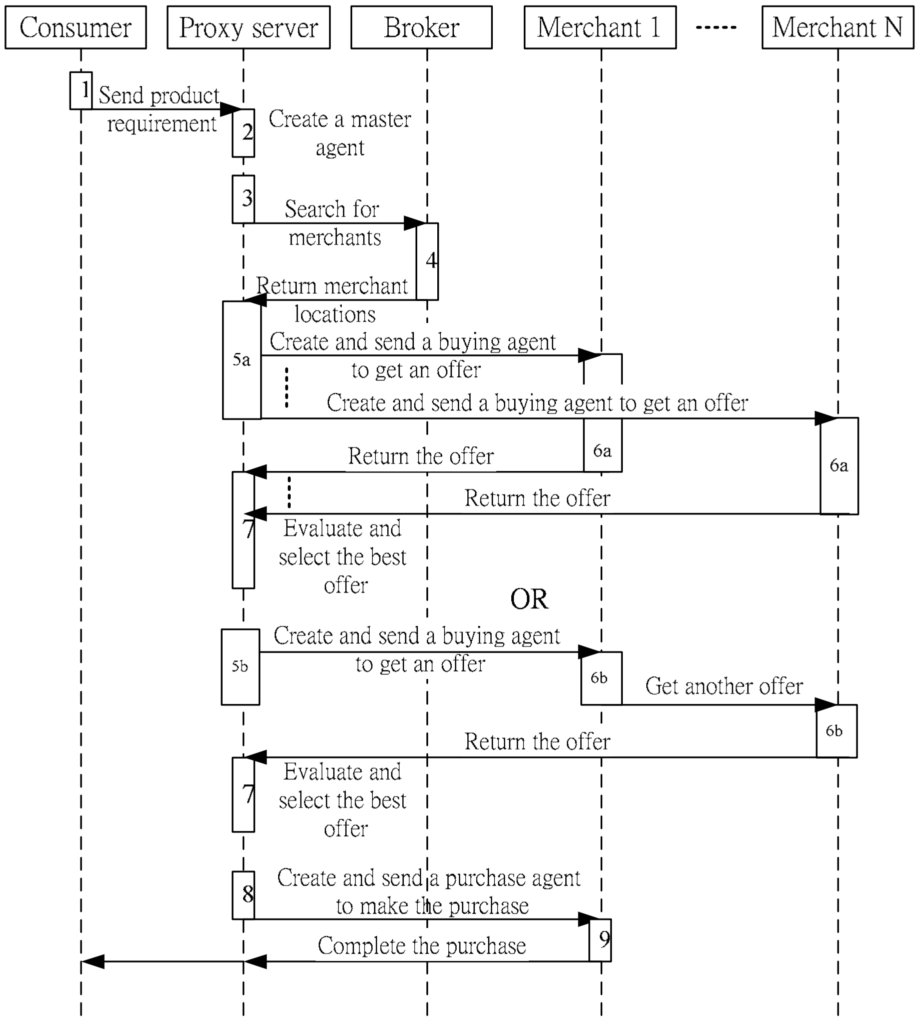
Based on [5][7] and inspired by the above-mentioned agent-based systems [8][11][33], Figure 14 describes the MAGICS buying process. To enhance extensibility, XML-based communication messages are used. The structure will be explained in Chapter 4. The process is summarized as follows:

1. A user conveys a shopping request to the proxy server via a Web browser or a mobile terminal. The requirement can be defined using a variety of terms, such as the product model number or the desired price.
2. The proxy server obtains the customer's information and creates a master agent at the MAGICS agent server for the user. The master agent will control the later buying process.
3. According to the shopping request, the master agent locates the merchant(s)

with the assistance of the broker agent. As in the Web-based e-commerce system, it is expected that brokers [36] or search engines will be available for this purpose.

4. The master agent obtains information on the location of the merchant(s).
5. A comparison is made. This may take place in one of two ways:
    a. Identical buying agents are generated to visit different merchants in parallel. Each buying agent forwards a request for a quotation to the quote agent for each merchant. The quote agent makes an offer to the buying agent by querying the database.
    b. One buying agent is created to visit the merchants sequentially. After receiving an offer from the quote agent, the buying agent moves on to the next shop. This mimics the real-world process of comparison shopping, in which we might visit a series of shops.
6. Depending on the method used (see step 5), the buying agent returns to the proxy server with
    a. a quotation from the current shop, or
    b. all of the quotations.
7. Having obtained the offers, the master agent compares them and chooses the best one. When evaluating a standard product, it may simply find the lowest price. In more complex cases involving many decision variables, the master agent may, however, need to evaluate the offers using mathematical techniques.
8. Upon choosing the best merchant, the master agent creates a purchase agent to complete the purchase with the target merchant. To make payments, the SET (Secure Electronic Transaction) protocol [34] can be used.

9. Lastly, the purchase agent obtains a receipt from the selling agent and moves back to the proxy server. In the case of digital goods, the purchase agent may even carry the product(s) with it. The master agent stores the receipt, and possibly the product(s), based on the user's instruction and informs the users accordingly.


Consumer
Proxy server
Broker
Merchant 1

Merchant N
1
Send product requirement
2
Create a master agent
3
Search for merchants
4
Return merchant locations
5a
Create and send a buying agent to get an offer
Create and send a buying agent to get an offer
6a
6a
Return the offer
Return the offer
7
Evaluate and select the best offer
OR
5b
Create and send a buying agent to get an offer
6b
Get another offer
6b
Return the offer
7
Evaluate and select the best offer
8
Create and send a purchase agent to make the purchase
9
Complete the purchase

Figure 14 The buying process in B2C MAGICS

As there may be a noticeable effect on system performance when the number of agents is large, the system does not create a lot of agents at the same time. When searching in parallel in MAGICS, we use a mathematical model [6] to determine the optimal number of agents to be sent.

This buying process can also be extended to support location-based shopping services in m-commerce. In this case, mobile agents can be employed to help search and evaluate the "digital space". The consumer will then make the final decision and complete the purchase in the "physical place". Essentially, the master agent selects the merchants and then sends their physical locations (e.g., ranked in ascending order of distance to the consumer) to the consumer (e.g., through the Short Messaging Service (SMS)) to complete the remaining buying tasks. This hybrid approach can combine the advantages of traditional commerce and the emerging e/m-commerce.

A consumer may need to evaluate multiple attributes (e.g., price, delivery time) in order to make a buying decision. Inspired by [10][19][40], the following presents an effective mathematical model for achieving this goal when buying standard products. Let us illustrate the model with several agents shopping in parallel. Suppose that there are N merchants for the agents to visit, where the t-th agent visits the t-th merchant. After receiving all of the offers from the merchants, the master buying agent calculates the average weighted score of the t-th merchant so as to determine how well the consumer's requirements can be satisfied. Suppose that there are M attributes for evaluation, such as selling price and traveling time to a shop. The i-th attribute is denoted as A_i and has a weight w_i to reflect its relative importance. To perform the evaluation based on the consumer's preference, we assume that all of the attributes can be "quantified" by using a rating or score. We define $S_{t,i}$ as the rating of A_i of the t-th shop (i.e., the one visited by the *t*-th agent). The average score $\overline{S_t}$ of the *t*-th shop can be found by finding the weighted sum of all $S_{t,i}$ as follows:

$$\overline{S_t} = \sum_{i=1}^{M} w_i \times S_{t,i} \tag{1}$$

Finally, the preferred merchant is selected based on the highest $\overline{S_t}$.

For implementation, the user can enter the searching criteria with the importance ranking of each attribute. The weights can be defined using values from 1 to 10 (from the least to the most important). In essence, a satisfaction function is used. For each attribute (e.g., *i*-th attribute), we assume that the offer by the *t*-th merchant can be represented by $V_{t,i}$. For example, the *t*-th merchant may return a selling price of $300. In this case, A_i is the price and $V_{t,i}$ is 300. Note that we can have boolean attributes (i.e., “Yes” or “No”) or discrete attributes (e.g., red, green or blue, each associated with a score), as well. In this system, we focus on continuous attributes (e.g., price, which has continuous values). Nevertheless, the model can be extended easily to cover other attributes. Based on an attribute value, the corresponding satisfaction score can be found by using a predefined satisfaction function $F(V_{t,i})$. As an example, we consider a linear function as follows (see Figure 15):

$$S_{t,i} = F(V_{t,i}) = \begin{cases} \dfrac{h_i - V_{t,i}}{h_i - l_i} & if \quad F(h_i) < F(l_i) \\ \dfrac{V_{t,i} - l_i}{h_i - l_i} & if \quad F(h_i) > F(l_i) \end{cases} \tag{2}$$

where h_i and l_i are the highest and lowest values for the *i*-th attribute.

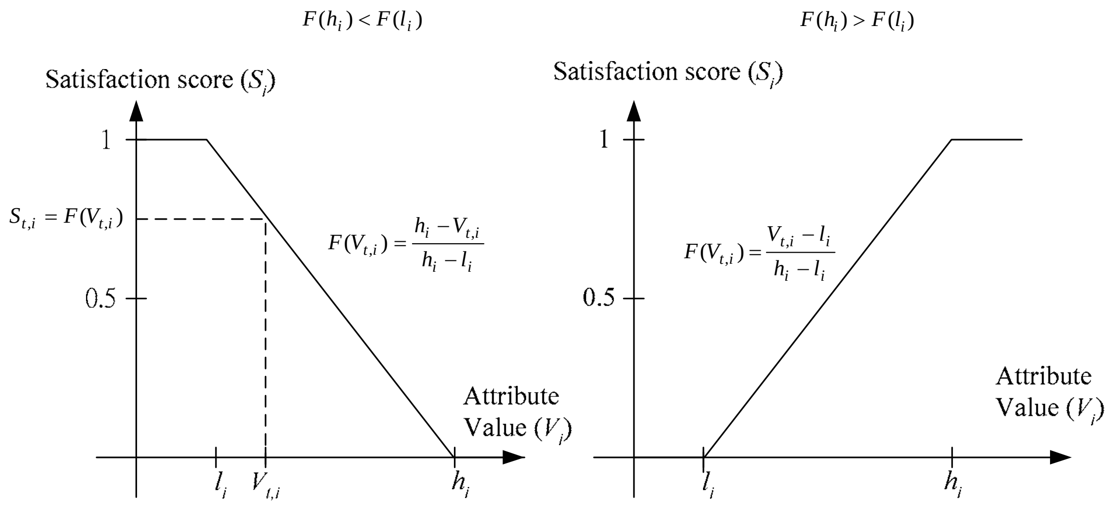


Figure 15 Linear satisfaction function under two different cases

For example, in the case of price, h_i = 300 and l_i = 250 if the price range is between \$250 and \$300. Note that depending on the attribute, a high value may lead to more or less satisfaction. For instance, a consumer prefers a lower price and a higher discount. Hence, the satisfaction function can be decreasing (i.e., $F(h_i) < F(l_i)$) or increasing (i.e., $F(h_i) > F(l_i)$), as shown in Figure 15. In the decreasing case, h_i and l_i result in a satisfaction score of 0 and 1, respectively. In the increasing case, we have the reverse situation. To determine h_i and l_i, we can ask a consumer two questions such as: "At what value (e.g., price) will you be fully satisfied?" and "At what value (e.g., price) will you be totally dissatisfied?" Based on the answers, the linear satisfaction function can be determined for the concerned attribute. Note that the linear function is just an example. Other functions can also be used.

3.2.3. Implementation

The agent programming interface is the base layer of MAGICS. We make use of

IBM Java Aglets [1][27] to implement MAGICS. IBM Java Aglets provides an object-oriented programming interface, a mechanism for moving code and showing data and state information, a platform-independent development and runtime environment, and security mechanisms. The properties of Java make Aglet very suitable for mobile agent programming. For example, multithread programming allows Aglets to behave autonomously, and platform independence lets Aglets be created or executed in any computers on the network.

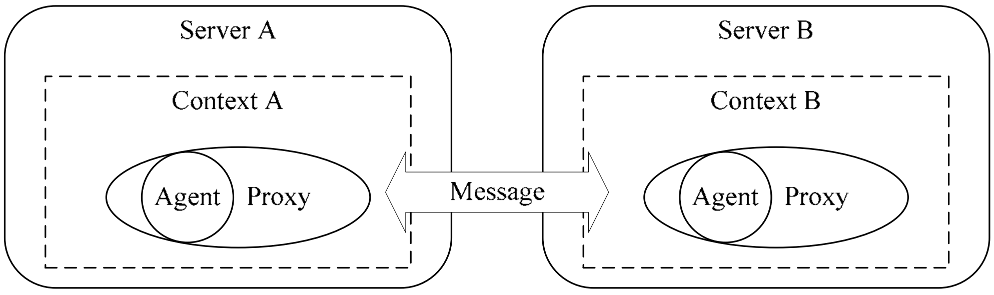


Figure 16 The components of Aglet Object Model

According to the specification of IBM Aglets [1], the fundamental functionalities are defined as the components of Aglet Object Model, which basically includes Aglet, AgletProxy, AgletContext and Message classes. The architecture of the components is shown in Figure 16. The abstract class Aglet defines the basic methods for controlling the mobility and life cycle of an Aglet. The fundamental operation of the Aglet class includes creation, cloning, dispatching, retraction, deactivation, activation and disposal. AgletProxy objects are the intermediary for accessing Aglets and provide protections for Aglets to prevent direct access to their public methods. Any communication between two or more Aglets is done via their respective AgletProxy objects. AgletContext provides an execution environment for Aglets. It does not move, and is uniquely identified by its host address and unique name for access by authorized Aglets. Message class is the object which is used for Aglets to communicate with each other. The message instances are distinguishable based on their “kind”, which is simply a string. A message-receiving Aglet typically supports message passing by implementing the handleMessage() method. The message class also provides a method for the Aglet to reply for an incoming message.

With the use of the operations provided by the IBM Java Aglets, we implement five kinds of agents within B2C MAGICS. They are explained as follows:

1. Master agent

The master agent is a stationary agent that controls the whole shopping process on behalf of the buyer. It captures a user request or user requirement. It also creates different kinds of agents, such as a broker agent, buying agent and purchase agent, so as to perform the later buying process. Moreover, it selects the best offers according to the results from buying agents and the user requirements. It is also responsible for returning the results to the buyer through e-mail or Short Messaging Service (SMS).

2. Broker agent

The broker agent is a mobile agent that retrieves information on the location of the merchants from broker sites. After getting the user requirements from the master agent, a broker agent is dispatched to broker sites to obtain the shop locations of the corresponding products. Finally, it travels back to the proxy server and returns the results to the master agent.

3. Buying agent

The buying agent is a mobile agent that travels to different shops and requests offers from the quote agents. It has two kinds of traveling modes. One is visiting each merchant one by one according to the itinerary. The other one is to dispatch agents in parallel to get quotations.

4. Quote agent

The quote agent is a stationary agent located in each merchant. It is responsible for making an offer to the buying agent. After receiving the selection criteria from the buying agent, the quote agent queries the database and sends the information to the buying agent.

5. Purchase agent

The purchase agent is a mobile agent that handles the purchase with the target merchant selected by the master agent. The purchase agent completes the payment, and then obtains a receipt from the selling agent and returns to the proxy server. In the case of buying an electronic product, the purchase agent carries both the receipt and the product. Note that MAGICS uses a mobile purchase agent to emulate the physical shopping process (i.e., to emulate how a person visits and shops and interacts with sellers).

6. Selling agent

The selling agent is a stationary agent that interacts with the purchase agent and gives the receipt to the purchase agent after getting the payment.

For purposes of demonstration and evaluation, a prototype system has been developed to show phase of the consumer buying process. The current system is extended and enhanced from previous work in [5][7]. Users can forward the shopping requirements to the proxy server via a WAP phone. At each merchant's server, a selling agent is created to communicate with the buying agents, and the product information is kept in a database. Figure 17 shows the WAP interface at the MAGICS proxy server. In the first demonstration, we use MAGICS to purchase a digital book. After the user fills in and confirms the required information (e.g., ISBN = 0123456789 and Name = E-commerce) as shown in Figure 17, the request is submitted to the proxy server. A master buying agent is then generated to control the buying process for the user. For simplicity, we assume that all of the prospective merchants/retailers are available in the proxy server. In this example, there are four target sellers/retailers. When the master buying agent has identified the four retailers, it sends several agents to each of them to obtain their offers. At each seller's server, the buying agent passes a request (in XML format) to the selling agent. The selling agent searches for the product's price in its database, and then provides a reply (offer) to the buying agent. The buying agent carries this offer to the proxy server. Having collected all of the (four) offers and identified the best one (i.e., the lowest price in this example, as shown in Figure 18), the master buying agent sends a purchase agent to the chosen retailer (Shop 2) to purchase the book. At the seller's server, the selling agent provides a copy of the digital book (i.e., an electronic file) for the purchase agent to carry back to the proxy server (see Figure 19). At the proxy server, the buying agent passes the digital book to the master agent, which stores it in the specified location.

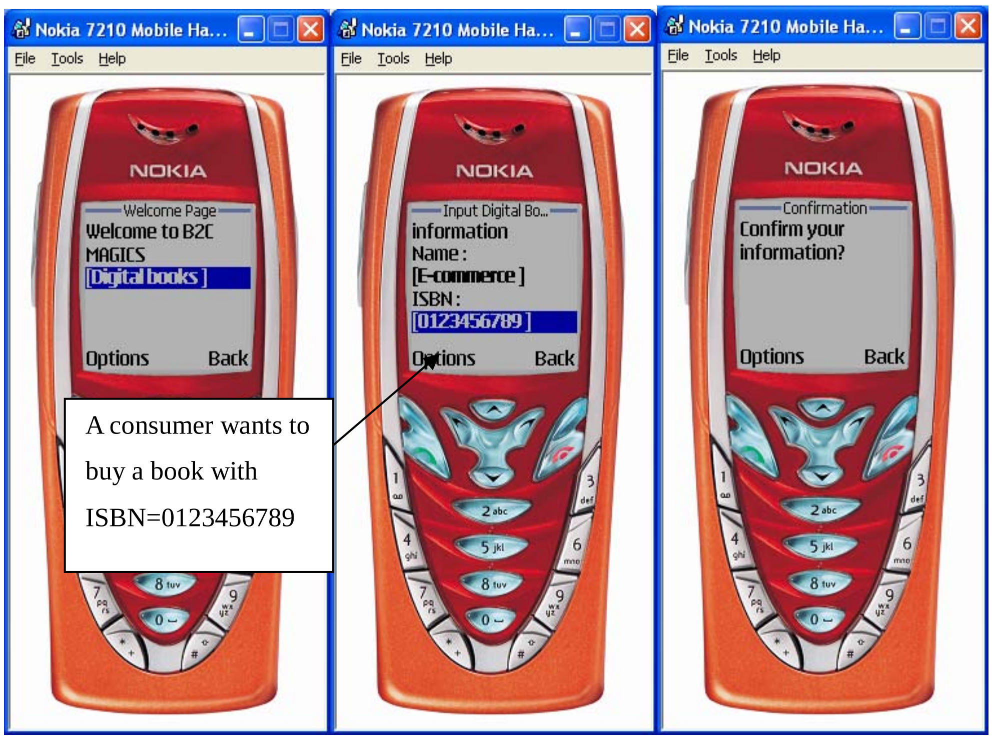
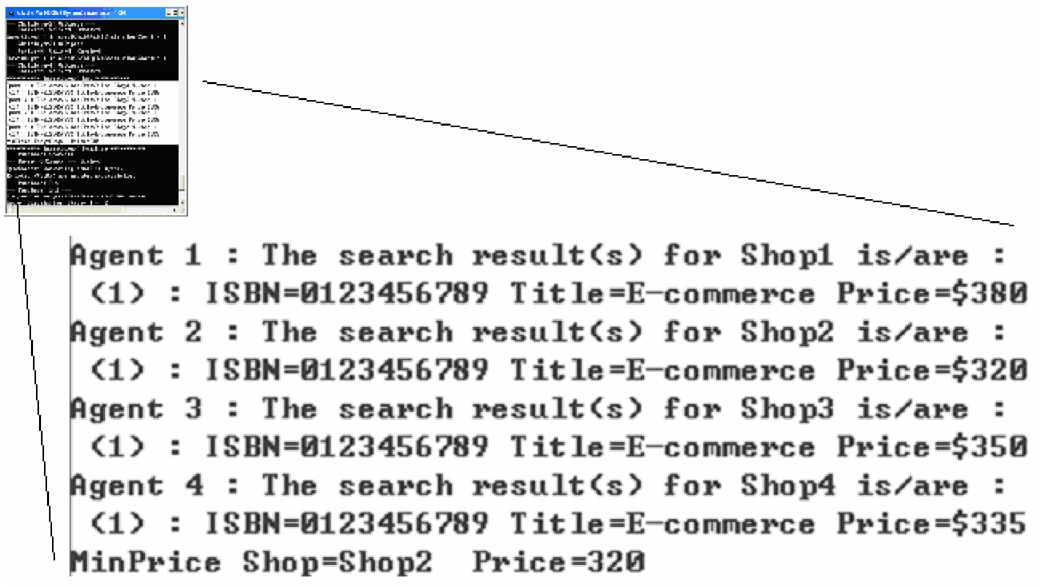
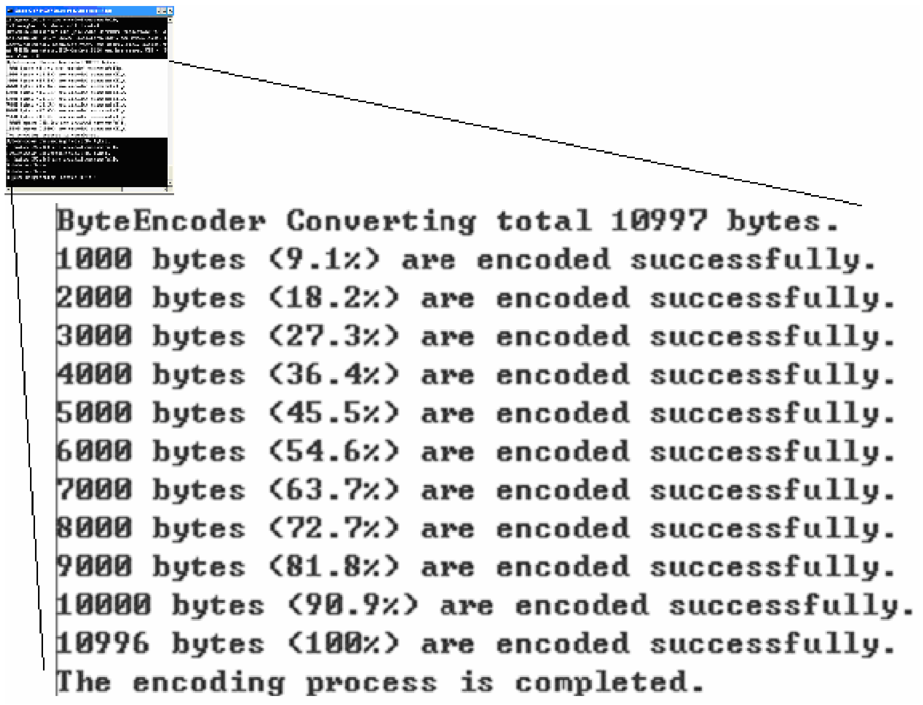


Figure 17 Accessing MAGICS via a WAP interface (Nokia 7210 simulator)



```
Agent 1 : The search result(s) for Shop1 is/are :
 (1) : ISBN=0123456789 Title=E-commerce Price=$380
Agent 2 : The search result(s) for Shop2 is/are :
 (1) : ISBN=0123456789 Title=E-commerce Price=$320
Agent 3 : The search result(s) for Shop3 is/are :
 (1) : ISBN=0123456789 Title=E-commerce Price=$350
Agent 4 : The search result(s) for Shop4 is/are :
 (1) : ISBN=0123456789 Title=E-commerce Price=$335
MinPrice Shop=Shop2  Price=320
```

Figure 18 Compare the offers returned by the master buying agent



```
ByteEncoder Converting total 10997 bytes.
1000 bytes (9.1%) are encoded successfully.
2000 bytes (18.2%) are encoded successfully.
3000 bytes (27.3%) are encoded successfully.
4000 bytes (36.4%) are encoded successfully.
5000 bytes (45.5%) are encoded successfully.
6000 bytes (54.6%) are encoded successfully.
7000 bytes (63.7%) are encoded successfully.
8000 bytes (72.7%) are encoded successfully.
9000 bytes (81.8%) are encoded successfully.
10000 bytes (90.9%) are encoded successfully.
10996 bytes (100%) are encoded successfully.
The encoding process is completed.
```

Figure 19 Transferring the digital book to the buying agent

In the second demonstration, we assume that a consumer must buy the same book at the nearest physical bookstore. Now we consider two attributes, namely price and traveling time to the bookstore. For simplicity, they have equal weights. As shown in Figure 20, the consumer enters the required information through his/her WAP phone. It can be seen that the consumer is fully satisfied and dissatisfied if the book is priced at $330 and $380, respectively. The preferred traveling time is 5 minutes and a traveling time of 30 minutes is unacceptable. Based on the information, four mobile shopping agents are sent to look for the book from a nearby bookstore. After obtaining the offers, the master agent evaluates the offer according to the satisfaction scores as discussed above. The evaluation results are shown in Table 2. It can be seen that shop 4 is preferred. Although its offered price is not the lowest, it gives the best satisfaction score after considering both attributes. This is just a simple example to demonstrate the basic function of B2C MAGICS. Many similar applications can be built using this framework to support e-commerce in general and m-commerce in particular.

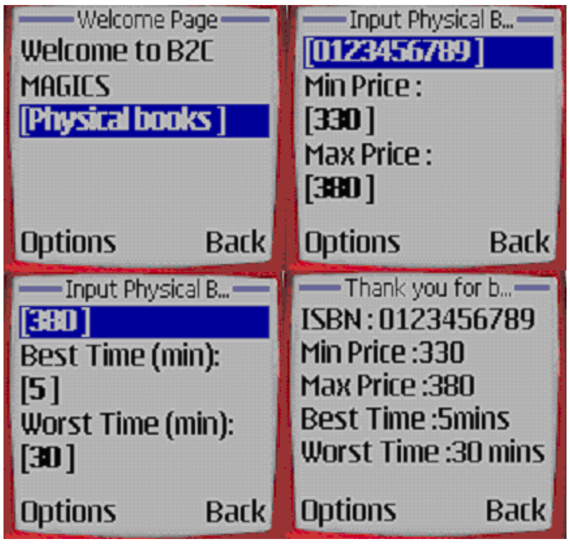
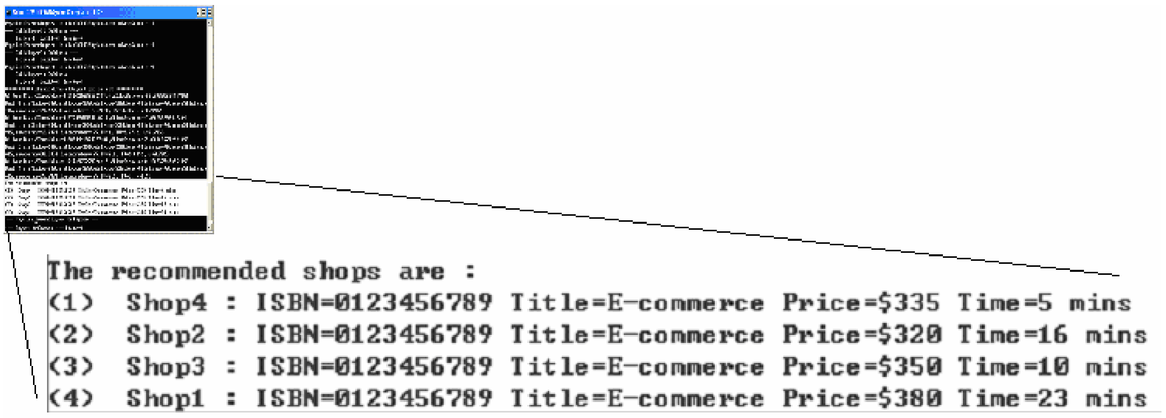


Figure 20 The inputted information for the second demonstration



```
The recommended shops are :
<1>  Shop4 : ISBN=0123456789 Title=E-commerce Price=$335 Time=5 mins
<2>  Shop2 : ISBN=0123456789 Title=E-commerce Price=$320 Time=16 mins
<3>  Shop3 : ISBN=0123456789 Title=E-commerce Price=$350 Time=10 mins
<4>  Shop1 : ISBN=0123456789 Title=E-commerce Price=$380 Time=23 mins
```

Figure 21 The recommended shops for the second demonstration

Table 1 Attributes and their values for the second demonstration

Parameter	Value
Attribute Set (A_i)	{price, traveling time}
Min. Price (l_1) (HK$)	330
Max. Price (h_1) (HK$)	380
Best Time (l_2) (mins)	5
Worst Time (h_2) (mins)	30

Table 2 Average score of each shop for the second demonstration

	Price ($V_{t,1}$) ($)	Traveling Time ($V_{t,2}$) (mins)	Attribute Score 1 (S_1)	Attribute Score 2 (S_2)	Score ($\overline{S_t}$)
Shop 1	380	23	0	0.285	0.143
Shop 2	320	16	1.2 => 1	0.573	0.787
Shop 3	350	10	0.6	0.805	0.703
Shop 4	335	5	0.9	1.0107 => 1	0.95

CHAPTER 4 COMMUNICATION FRAMEWORK FOR B2C MAGICS

One of the objectives of this thesis is to define an XML-based MAGICS communication scheme. In conventional agent-based systems, agents typically communicate using a special language such as Knowledge Query Manipulation Language (KQML) [24][25]. Recently, there has been considerable interest in employing the Extensible Markup Language (XML) to facilitate inter-agent communications [16]. The XML schema has a number of advantages. It can be used to effectively and flexibly define the "grammar" of the communications. It also can maintain the traditional object-oriented features and advantages. Thus, the focus of this chapter is on designing the buying protocol and the inter-agent communication mechanism based on XML.

4.1. Agent Communication Language

Information exchange between buyers and sellers is an important issue. In traditional commerce, they communicate with each other through conversations. In an agent-based environment, the buying and selling agents can communicate using different agent communication languages. KQML is a well-known language that has been used in many agent development projects [42]. In recent years, there has been considerable interest in adopting XML to support agent-to-agent communications [16][23][38], because it can overcome some of the shortcomings of KQML. In particular, the XML schema can be employed to effectively and flexibly specify the structure or format of the communication language.

KQML is a text message according to the speech act theory which concerns the role of language as an action [24]. Systems using KQML have three requirements. Firstly, a set of API is required to handle composition and exchange messages. Secondly, a service infrastructure is required to do naming, registration and basic facilitation services. Thirdly, reserved performatives are required to take semantically prescribed actions [25]. When defining the basic ontology for agent communications, which includes shared vocabularies and the relationships between vocabulary items, KQML offers a narrow and inflexible way to perform the tasks compared with XML [25]. XML is a subset of Standard Generalized Markup Language (SGML). It allows the creation of new tags to describe a set of information and defines the "grammar" of the communications. XML can be structured by Document Type Definition (DTD) or the XML Schema (see http://www.w3.org/XML/Schema for details). By doing so, an XML message can be interpreted by an agent without ambiguity. Thus, it provides a standard communication format for exchanging information. Also, the information can be easily extracted by and integrated with an XML parser as it makes use of the same API for manipulating XML data. Programmatic access to the document's content allows agents to process the data without a doubt [16]. XML is more powerful than other agent communication languages because of its extensibility and ubiquity. Besides adding new tags for each particular subject, it is possible to use an XML schema to extend the data type. That means a complex data type can be defined by inheriting simple data types.

We have chosen XML as the MAGICS agent communication language. MAGICS makes use XML schema to define the XML messages. It is because XML schema provides more functions than DTD [4]. In particular, XML schema can define more relationships between elements such as specifying the exact number of occurrences of each element. It also provides more data types, including string, numeric, date, time and structure. The major advantage of XML schema is in realizing the traditional object-oriented features.

4.2. XML-based Communication Scheme

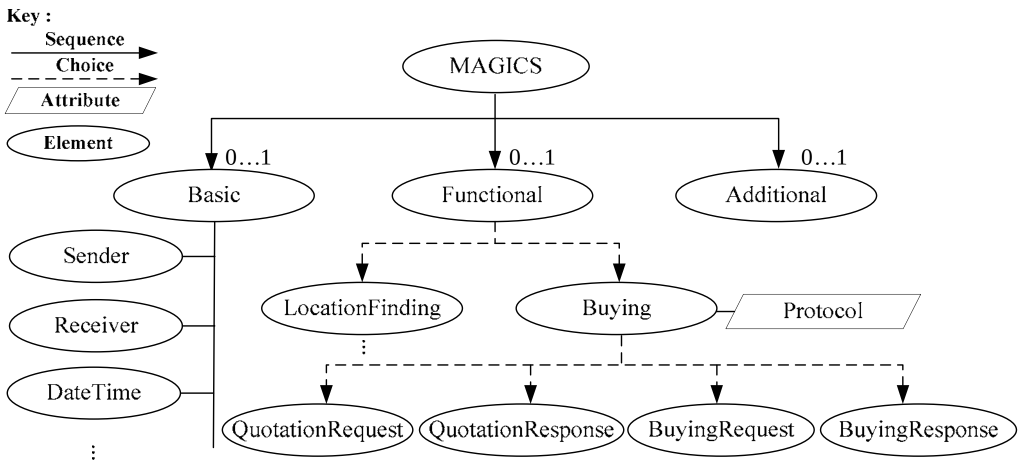


Figure 22 The partly containment of MAGICS element

Agents in MAGICS communicate with each other through MAGICS messages, which are defined by XML schema. A MAGICS message consists of three sections: a basic information section, a functional information section and an additional information section. The general structure is shown in Figure 22.

Basic information is information which is common to all kinds of messages such as locations/addresses of senders and receivers, their reference codes, message generation date/time, and required reply deadline (if any). For sensitive or important messages, the basic information can also include a digital signature to safeguard the integrity of the message. Functional information is function-specific; for instance, a buying function, which will be defined and explained later. Each function usually consists of several procedures. The procedures involved in buying something include getting quotations and completing purchases, and so forth. Each function has a *Protocol* attribute to specify the associated namespace. This allows businesses to flexibly add their own functions and requirements. For instance, having set up a new namespace called *aaa*, the corresponding quotation request element can be specified like: <aaa: QuotationRequest>. In a later example, we discuss the procedures for a quotation request/response, negotiation request/response and buying request/response. In a quotation request, the buying agent gives the basic product information together with the product requirement. In a quotation response, the selling agent replies with the detailed product information about the product and the price of the product. The proposed method can also support negotiations. Specifically, upon evaluating the responses, the buying agent sends the preferred seller a negotiation request which includes the negotiation action, the product information and the negotiation criteria. The selling agent then returns the negotiation response based on the information in the negotiation request. When both agents are satisfied with the result of the negotiation, the buying agent provides the selling agent with the buying request containing the product information and the customer information. Finally, the selling agent returns a buying response to the buying agent with the relevant information. In the following, we employ a simple example to explain the basic operations. Note that by employing the XML schema, we can specify messages flexibly and extend the basic procedures to address other more complicated situations.

As B2C MAGICS is a system for supporting the consumer buying process, it must handle different kinds of product information. Even though the products are grouped in different categories, they still have much common information. For example, when a buyer requests for a quotation of a book, he/she will enter the book name, author name and publication year. When obtaining a quotation for a notebook, he/she will enter the brand name, model number and manufacture year. It can be seen that the product name and creation date are common product information. The non-common product requirement can be defined by extending the general product requirement. As a result, the inheritance mechanism in object-oriented programming language can be applied. An XML Schema can be defined using the object-oriented mechanism. Figure 23 shows the structure of the general product and particular products. As an example, we show the XML schema for the product information in Figure 24. Other schemas (e.g., product requirement and reply) can be specified using a similar approach. The *ProductInformation* provides the basic information on a product, including the title, identifier and creation (year). All products should provide such information. For the identifier, we can give detailed information through the *IdentifierType* attribute. For instance, a book should have a book title, an ISBN (i.e., the identifier) and a publication year (i.e., the creation year). Note that the product information has been defined as generally as possible so that it can be applied to various kinds of products. We may also provide product-specific information for certain products. In the example, the book information may include the author and publisher. Through the XML schema, we can specify additional/detailed information simply by extending the general product information (see Figure 24). This is similar to the approach used in [23]. For instance, as shown in Figure 24, a new element called *BookInformation* (as extended from the *ProductInformation*) is introduced.

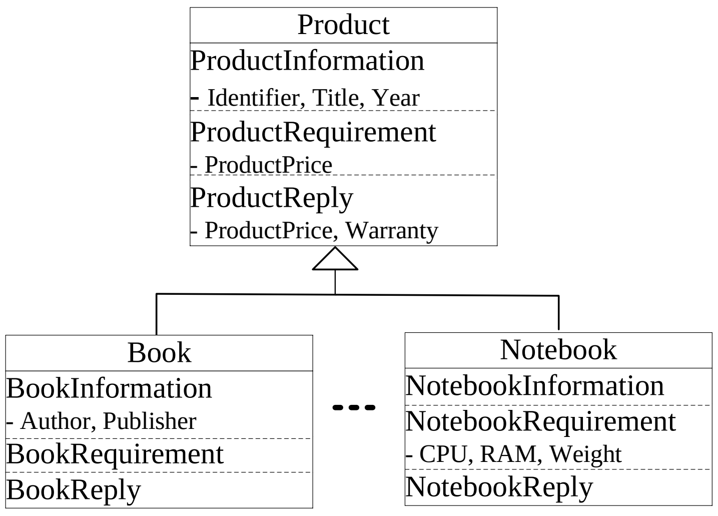


Figure 23 The structure of the general product and particular products

```
<complexType name="ProductInformation">
 <sequence>
  <element name="Identifier" minOccurs="0">
   <complexType>
    <simpleContent>
     <extension base="string">
      <attribute name="IdentifierType" type="string"/>
     </extension>
    </simpleContent>
   </complexType>
  </element>
  <element name="Title" type="string" minOccurs="0"/>
  <element name="Year" type="gYear" minOccurs="0"/>
 </sequence>
</complexType>
<element name="BookInformation">
 <complexType>
  <complexContent>
   <extension base="MAGICS:ProductInformation">
    <sequence minOccurs="0">
     <element name="Author" type="string" minOccurs="0"/>
     <element name="Publisher" type="string" minOccurs="0"/>
    </sequence>
    <attribute name="BookType" type="string" use="optional"/>
   </extension>
  </complexContent>
 </complexType>
</element>
```

Figure 24 An XML schema for a MAGICS message

4.3. Example of a Simple Buying Process

In this section, we use a simple example to explain the basic operations of the communication scheme to support the consumer buying process. For illustration, we have either filled in dummy data or "…" for the elements.

1. A buying agent sends a quotation request.

```
<?xml version="1.0" encoding="UTF-8"?>
<MAGICS xmlns="http://127.0.0.1/Schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://127.0.0.1/Schema MAGICS.xsd">
 <Basic>
  <Sender>magics://aaa.com/buyagent</Sender>
  <Receiver>magics://zzz.com/sellagent</Receiver>
  <DateTime>2004-07-01T9:00:00</DateTime>
  <ExpiryDateTime>2004-07-02T9:00:00</ExpiryDateTime>
  <SenderReferenceCode>S1234</SenderReferenceCode>
 </Basic>
 <Functional>
  <Buying Protocol="http://127.0.0.1/Schema">
   <QuotationRequest ProductType="Book">
    <BookInformation>
     <Name>E-Commerce</Name>
    </BookInformation>
    <BookRequirement>
     <ProductPrice Requirement="Highest">250</ProductPrice>
    </BookRequirement>
   </QuotationRequest>
  </Buying>
 </Functional>
</MAGICS>
```

Figure 25 Quotation request message

Figure 25 shows the message for the quotation request. The basic information provides the message-specific information. The functional information provides the buying information. The associated protocol or schema is stated in the *Protocol* attribute (i.e., the given URL). For illustration, we assume that it is defined locally. Of course, in a real implementation, the associated protocol should be specified/stored at a public location. The *QuotationRequest* element specifies the quotation request information. The *ProductType* attribute identifies the type of product involved. In the example, the type is “Book” and the corresponding elements of *BookInformation* and *BookRequirement* are included in the quotation request. The *BookInformation* element provides the general information on the book e.g., the title of the book (i.e., the information is identical for all buyers). The *BookRequirement* element gives the buyer specific requirements (e.g., the required product price). To facilitate searching/data retrieval, the *ProductPrice* element includes the *Requirement* attribute, which provides further information on the price requirement. In the example, the highest price is given i.e., the buying agent wants to buy a book entitled E-commerce at a price of at most $250.

2. A selling agent replies with a quotation response.

```
<?xml version="1.0" encoding="UTF-8"?>
<MAGICS xmlns="http://127.0.0.1/Schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://127.0.0.1/Schema MAGICS.xsd">
 <Basic>
  <Sender>magics://zzz.com/sellagent</Sender>
  <Receiver>magics://aaa.com/buyagent</Receiver>
  <DateTime>2004-07-01T9:05:00</DateTime>
  <ExpiryDateTime>2004-07-02T9:00:00</ExpiryDateTime>
  <SenderReferenceCode>R1234</SenderReferenceCode>
  <ReceiverReferenceCode>S1234</ReceiverReferenceCode>
 </Basic>
 <Functional>
  <Buying Protocol="http://127.0.0.1/Schema">
   <QuotationResponse ProductType="Book">
    <BookInformation>
     <Identifier IdentifierType="ISBN">0123456789</Identifier>
     <Title>E-Commerce</Title>
     <Year>2004</Year>
     <Author>…</Author>
     <Publisher>… </Publisher>
    </BookInformation>
    <BookReply>
     <ProductPrice>230</ProductPrice>
    </BookReply>
   </QuotationResponse>
  </Buying>
 </Functional>
</MAGICS>
```

Figure 26 Quotation response message

```
<?xml version="1.0" encoding="UTF-8"?>
<MAGICS xmlns="http://127.0.0.1/Schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://127.0.0.1/Schema MAGICS.xsd">
 <Basic>
  <Sender>magics://zzz.com/sellagent</Sender>
  <Receiver>magics://aaa.com/buyagent</Receiver>
  <DateTime>2004-07-01T9:05:00</DateTime>
  <ExpiryDateTime>2004-07-02T9:00:00</ExpiryDateTime>
  <SenderReferenceCode>R1234</SenderReferenceCode>
  <ReceiverReferenceCode>S1234</ReceiverReferenceCode>
 </Basic>
 <Functional>
  <Error>
   <ErrorNumber>2</ErrorNumber>
   <ErrorType>Syntax error</ErrorType>
  </Error>
 </Functional>
</MAGICS>
```

Figure 27 Quotation response message with an error

Having received the quotation request message, the selling agent processes it. It looks for the required product from a database in accordance with the product information and the product requirement. It then sends the quotation response back to the buying agent. Figure 26 shows an example of the quotation response message. For purposes of identification, the sender reference code is enclosed in the basic information. To allow the buying agent to identify the transaction, the receiver reference code should be identical to the reference code sent by the buying agent in the quotation request message. For the functional information, the *QuotationResponse* element is included. The associated sub-elements are: *BookInformation* and *BookReply*. *BookInformation* gives specific information on the book. *BookReply* provides the information about the price and possibly other related information. If the selling agent is not able to process the received message (e.g., because of syntax errors), the corresponding response message as shown in Figure 27 will be returned. An error message is enclosed by means of an *Error* element. Various error messages can be specified in a similar manner.

3. The buying agent sends a negotiation request message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MAGICS xmlns="http://127.0.0.1/Schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://127.0.0.1/Schema MAGICS.xsd">
 <Basic>
  <Sender>magics://aaa.com/buyagent</Sender>
  <Receiver>magics://zzz.com/sellagent</Receiver>
  <DateTime>2004-07-01T9:10:00</DateTime>
  <ExpiryDateTime>2004-07-02T9:00:00</ExpiryDateTime>
  <SenderReferenceCode>S1234</SenderReferenceCode>
  <ReceiverReferenceCode>R1234</ReceiverReferenceCode>
 </Basic>
 <Functional>
  <Buying Protocol="http://127.0.0.1/Schema">
   <NegotiationRequest>
    <NegotiationAction>propose</NegotiationAction>
    <BookInformation>
     <Identifier IdentifierType="ISBN">0123456789</Identifier>
    </BookInformation>
    <BookReply>
     <ProductPrice>200</ProductPrice>
    </BookReply>
   </NegotiationRequest>
  </Buying>
 </Functional>
</MAGICS>
```

Figure 28 Negotiation request message

After evaluating the responses, the buying agent forwards a negotiation request to the preferred seller (see Figure 28). Again, for purposes of identification, the receiver reference code is based on the sender reference code provided in the quotation response message. For the functional information, the specified function is "buying" and the procedure is "negotiation request". Firstly, the request provides the negotiation action. There are four kinds of action types. The first action type is “propose” - the agent starts the negotiation. The second action type is “accept” - the agent accepts the offer raised by the opponent. The third action type is “reject” - the agent rejects the offer and states the new negotiation terms. The last action type is “failed” - the agent does not bargain anymore and terminates the negotiation. Secondly, for simplicity, just the *BookIdentifier* element is provided as the product can be identified with just this information. Finally, the product reply element is used to specifying the negotiation criteria.

4. The selling agent responds a negotiation response message.

```
<?xml version="1.0" encoding="UTF-8"?>
<MAGICS xmlns="http://127.0.0.1/Schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://127.0.0.1/Schema MAGICS.xsd">
 <Basic>
  <Sender>magics://zzz.com/sellagent</Sender>
  <Receiver>magics://aaa.com/buyagent</Receiver>
  <DateTime>2004-07-01T9:15:00</DateTime>
  <ExpiryDateTime>2004-07-02T9:00:00</ExpiryDateTime>
  <SenderReferenceCode>R1234</SenderReferenceCode>
  <ReceiverReferenceCode>S1234</ReceiverReferenceCode>
 </Basic>
 <Functional>
  <Buying Protocol="http://127.0.0.1/Schema">
   <NegotiationResponse>
    <NegotiationAction>accept</NegotiationAction>
    <BookInformation>
     <Identifier IdentifierType="ISBN">0123456789</Identifier>
    </BookInformation>
    <BookReply>
     <ProductPrice>200</ProductPrice>
    </BookReply>
   </NegotiationResponse>
  </Buying>
 </Functional>
</MAGICS>
```

Figure 29 Negotiation response message

The selling agent extracts the negotiation criteria and checks whether the criteria can be fulfilled. Figure 29 shows that the selling agent accepts the proposal. The *negotiationAction* element is "accept" and the same negotiation criteria as in the request are attached. The negotiation ends if both parties receive an "accept" or a "failed" result.

5. The buying agent sends a buying request message.

```
<?xml version="1.0" encoding="UTF-8"?>
<MAGICS xmlns="http://127.0.0.1/Schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://127.0.0.1/Schema MAGICS.xsd">
 <Basic>
  <Sender>magics://aaa.com/buyagent</Sender>
  <Receiver>magics://zzz.com/sellagent</Receiver>
  <DateTime>2004-07-01T9:20:00</DateTime>
  <ExpiryDateTime>2004-07-02T9:00:00</ExpiryDateTime>
  <SenderReferenceCode>S1234</SenderReferenceCode>
  <ReceiverReferenceCode>R1234</ReceiverReferenceCode>
  <DigitalSignature>…</DigitalSignature>
 </Basic>
 <Functional>
  <Buying Protocol="http://127.0.0.1/Schema">
   <BuyingRequest>
    <BookInformation>
     <Identifier IdentifierType="ISBN">0123456789</Identifier>
    </BookInformation>
    <CustomerInformation>
     <CustomerName> … </CustomerName>
     <Address> … </Address>
     <CreditCardNo> … </CreditCardNo>
     <ExpiryYearMonth>2007-07</ExpiryYearMonth>
    </CustomerInformation>
   </BuyingRequest>
  </Buying>
 </Functional>
 <Additional>
  <DigitalCertificate> … </DigitalCertificate>
 </Additional>
</MAGICS>
```

Figure 30 Buying request message

If both agents accept for the negotiation terms, the buying agent forwards a buying request with the product identifier simply. For delivering the book, the customer information is provided in the message as well. To protect the integrity of the message, the digital signature of the message is enclosed in the basic information. A digital certificate can be provided (i.e., attached as the additional information) for the selling agent to verify the digital signature with the buying agent’s public key.

6. The selling agent returns a buying response message.

```
<?xml version="1.0" encoding="UTF-8"?>
<MAGICS xmlns="http://127.0.0.1/Schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://127.0.0.1/Schema MAGICS.xsd">
 <Basic>
  <Sender>magics://zzz.com/sellagent</Sender>
  <Receiver>magics://aaa.com/buyagent</Receiver>
  <DateTime>2004-07-01T9:25:00</DateTime>
  <ExpiryDateTime>2004-07-02T9:00:00</ExpiryDateTime>
  <SenderReferenceCode>R1234</SenderReferenceCode>
  <ReceiverReferenceCode>S1234</ReceiverReferenceCode>
  <DigitalSignature>…</DigitalSignature>
 </Basic>
 <Functional>
  <Buying Protocol="http://127.0.0.1/Schema">
   <BuyingResponse>
    <BookInformation>
     <Identifier IdentifierType="ISBN">0123456789</Identifier>
    </BookInformation>
    <ReceiptInformation>
     <ReceiptNumber>O1234</ReceiptNumber>
     <ProductPrice>230</ProductPrice>
    </ReceiptInformation>
   </BuyingResponse>
  </Buying>
 </Functional>
 <Additional>
  <DigitalCertificate> … </DigitalCertificate>
 </Additional>
</MAGICS>
```

Figure 31 Buying response message

After receiving the buying request message, the selling agent processes the request. It also retrieves the customer’s credit card information for payment purposes. If everything is fine, it sends back the buying response to the buying agent (see Figure 31). The response message functions like a receipt for the order and the message is signed digitally. Similar to step 5, the selling agent can also enclose its digital certificate.

4.4. Summary

In conclusion, we have proposed an Extensible Markup Language (XML) scheme for inter-agent communications. Agents in MAGICS use MAGICS messages to talk with each other. The general structure of MAGICS messages includes basic, functional and additional information. Product information is the general information used in the consumer buying process. We have presented the XML schema for defining general product information. When a new product is added into the system, the structure of the new product information can be defined effectively based on the general structure. Moreover, we have used an example to illustrate the basic operation of the communication scheme so as to support the consumer buying process.

CHAPTER 5
MARKOV-DECISION-BASED PRICE COMPARISON MODEL

In a typical consumer-oriented buying process, before making the purchase decision, a consumer needs to evaluate the product, in particular, by comparing prices. The consumer has to visit shop-by-shop in physical shopping, or site-by-site in electronic shopping. Upon gathering the information from each shop, the consumer differentiates them and evaluates the differences based on certain criteria. This process incurs high searching cost and processing cost. Several agent-based systems have been developed to facilitate such comparisons both in the physical world and the cyber world.

Shopper's Eye [13] is an agent system which makes use of personal digital assistants (PDAs) to search for and evaluate products in a shopping center. The consumer uses a PDA to prepare a shopping list to specify what to buy and which shops to visit. Upon entering a shopping mall, the system uses the Global Positioning System (GPS) to locate shops on the list and then notifies the consumer. A consumer can also use the browse mode to look for adjacent shops and can use Shopper's Eye to compare products sold in nearby shops. When the consumer selects an item, the system can compare the prices of the same item sold by adjacent shops and then notify the consumer of the best local price.

Anderson Consulting's BargainFinder [22], a pioneer in agent-based product comparison, provides availability and price information about compact discs (CDs). In response to user input, the system provides hyperlinks to Web sites that sell the CD that the user requested. The user can buy the CD by clicking on the corresponding link. A Pocket BargainFinder [3] was also developed to work in an integrated electronic and physical commerce environment. In this case, a user can get the identity of a product (e.g., by scanning the bar code) from a physical shop. The user then requests that the system search for the same product from cyber-stores. If a cyber-store offers a lower price, the user can buy the product through his/her mobile terminal. ShopBot [12] obtains information from seller sites in order to provide a product-specific comparison service. To use the service and facilitate the learning process, users are required to provide product domain information (see [12] for details). This agent-based system is in two phases, an offline learning followed by online comparison shopping. In the offline learning phase, it analyzes the uniform resource locators (URL) of the merchants and the product domain information so as to obtain the vendor descriptions/information. Online comparison shopping is activated when the system receives a shopping request from a consumer. The system then compares and analyzes vendor descriptions/information using Artificial Intelligence techniques, including heuristic search, inductive learning and pattern matching and follows this with a recommendation or the provision of other relevant information. Price Watcher [14] is capable of getting price and other information from competitors through the WebSpy system [15]. This agent-based system obtains the required information by retrieving Web pages from competitor sites. It keeps this information in a local database which can be used for price comparisons. Most of these agent-based systems focus on obtaining and comparing price information.

We propose in this chapter to complement this work with a Markov decision-based price-comparison model. Our proposed contribution differs from other current systems in that it investigates a mobile agent-based system, takes into account of the use of network resources and provides a more complete service by using mobile agents to emulate the whole buying process not just to compare prices. To support MAGICS, we consider a Markov-decision-based price comparison problem in this chapter.

5.1. General Markov-decision-based Model Formulation

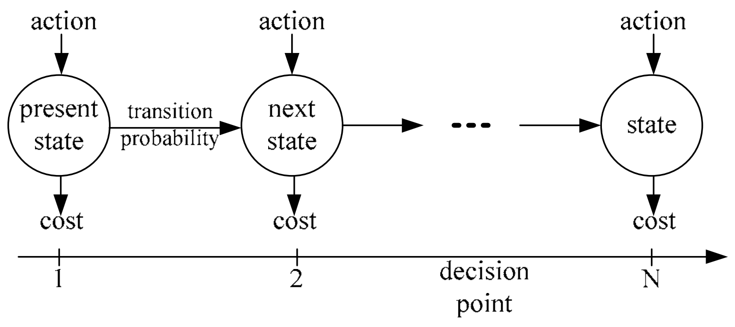


Figure 32 The general Markov-decision-based model

The Markov Decision Process (MDP) [31][32] has been widely used as a general framework for treating control problems with available decisions which can be taken when the system is in certain states. The general structure of the discrete-time model is described in Figure 32. The system is divided into *N* decision points representing the exact time at which decisions can be made. A control problem is classified as either finite horizon or infinite horizon based on the number of decision points. If *N* is a real number, it is a finite horizon MDP; otherwise it is an infinite horizon MDP. The system also has a set of possible states. Based on certain states at each decision point, an action is chosen. This produces two results: one is an immediate cost, and the other is a new state at a subsequent point in the timeline. Both results are calculated by using a state transition probability which represents the probability of changing from the present state to the next state.

In summary, the calculation of MDP requires five sets of items .

- A set of decision points
- A set of system states used to describe the system
- A set of actions available for the system to decide
- An immediate cost associated with a particular state and a certain action
- A state transition probability bringing the system from current state to next state when action is selected

5.2. The Price Comparison Problem

In this section, we formulate the price-comparison problem with Markov Decision theory. In a mobile agent-based system, a user can perform the search and evaluation functions by sending agents to multiple merchants either in parallel or by sending one agent to visit the merchants sequentially. In this thesis, no special strategy is used to set the priorities for site visits. The parallel agents visit the sites at the same time and the sequential agent visits the sites in a predefined order. In future work, priorities can be set in different ways e.g., according to the preferences of previous buyers. In this section, we consider the sequential approach and assume that there is a dispatching cost (or traveling cost) when sending an agent from one merchant to another. This cost may represent the Internet usage charge and other processing costs. Our aim in examining this problem is to determine the optimal decision policy for minimizing the expected cost (including the traveling cost and the cost of buying the item).

We have formulated two Markov-decision-based models that can support price comparison using a mobile agent. The general scenario of the model is as follows. We assume that there are N shops and that a mobile agent visits them in sequence. At each shop, the mobile agent must decide whether to buy the item or travel to the next shop. If the mobile agent buys the item, it will pay the required cost (see later models) and visit no other shops. Otherwise, it will move to the next shop and pay a constant traveling cost, C_{travel}. At the next shop, it must make the same decision again. In the first model, the agent cannot return to the previous shops (see Figure 33). To a certain extent, this emulates the physical shopping process because it is generally inconvenient to make a return trip. In the second model, the agent can return to the previous shop that offers the lowest price and buy the item. The cost of returning to the previous shop, the traveling cost C_{travel}, should also be paid (see Figure 34). The goal of these two models is to minimize the expected cost, which includes the cost of the product and the traveling cost.

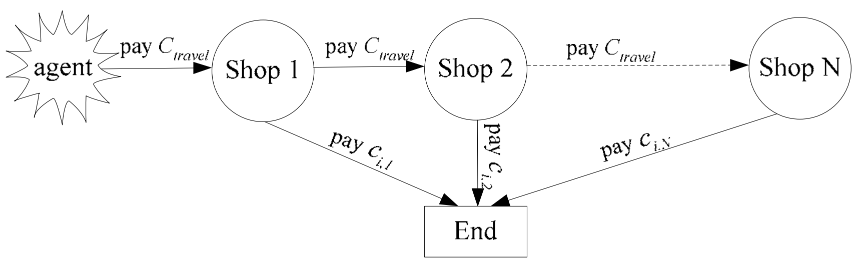


Figure 33 Flow diagram of Model I

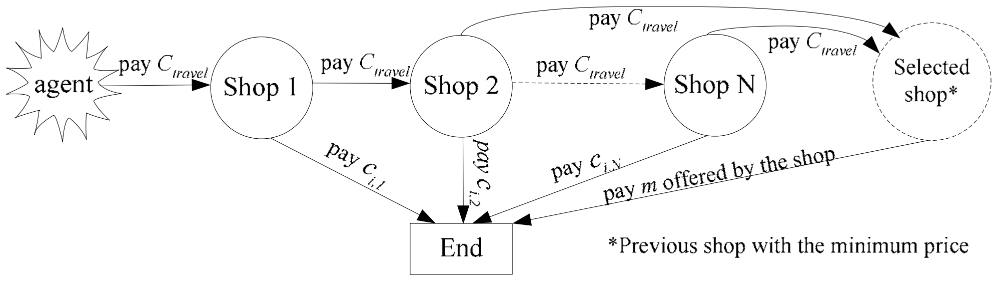


Figure 34 Flow diagram of Model II

Following the approaches/notations in [31][32], we formulated two Markov-decision-based models as follows. Table 3 shows the notations used in the two models. Note that for *s*, *m* and *z*, a subscript *t* can be added to specify the respective values at *t*, e.g., s_t denotes the system state at *t*.

Table 3 Notations used in the price-comparison models

Symbol	Description
t	The decision point t
S	The set of states
s	The system state
c_i	The price level i of the product
$c_{i,t}$	The offered price of the product is c_i at t
m	The lowest price recorded
Θ	The end state
Z	The set of choices
z	The selected choice
ρ	The state transition probability
P	The price distribution
B_t	The cost incurred at t
C_{travel}	The traveling cost
Ω_t	The average accumulated cost at t
ω_t	The minimum average accumulated cost at t

5.2.1. Model I

According to the price comparison problem, the mobile agent must decide whether to buy an item or visit the next shop. The decision points are 1, 2, … t, … N, where N is the number of shops. The current decision point is denoted as t and the next one as t+1. The system states are defined by two elements. One is the price of the product c_i, where $c_i > 0$, $i = 1, 2, \ldots n$, and $c_i < c_{i+1}$ for all i. The other is the end state, Θ. The set of possible states is time-independent, which means that they are the same at each decision point. The set of states at each shop is denoted as $S = \{c_i\} \cup \{\Theta\}$. Furthermore, we use $c_{i,t}$ to represent the state at which the price being offered for the product is c_i at decision point t. Figure 33 illustrates the change in state at each decision point. When the agent travels to the next shop, the state changes to the price offered by the next shop. If the agent decides to buy at the current shop, it moves to the “end” state.

At each decision point, the mobile agent can make one of the two choices: “end” or “move”. The set of choices is time-independent. Let $Z(s)$ denote the set of choices for state $s \in S$:

$$Z(s) = \begin{cases} \{1\,(\text{move}), 0\,(\text{end})\}, \text{if } s \in \{c_i\} \\ \{0\}, \qquad\qquad\qquad \text{if } s = \Theta \end{cases} \tag{3}$$

Note that once the agent reaches the “end” state (Θ), it will remain there.

Let $\rho(s_{t+1} | s_t, z_t)$ denote the state transition probability from the current state, $s_t \in S$ at decision point t, to the next state, $s_{t+1} \in S$ at decision point t+1, when the selected choice at decision point t is $z_t \in Z(s_t)$. It is given by the following expression for t = 1, 2, … N-1:

$$\rho(s_{t+1} \mid s_t, z_t) = \begin{cases} P(s_{t+1}), \ s_t, s_{t+1} \in \{c_i\}, z_t = 1 \\ 1, \qquad (s_t \in \{c_i\}, s_{t+1} = \Theta, z_t = 0) or (s_t = s_{t+1} = \Theta, z_t = 0) \\ 0, \qquad \text{otherwise,} \end{cases} \quad (4)$$

We assume that the offered price is governed by a probability distribution $P(c_i)$. For instance, it may be obtained based on past statistics for similar products. If the decision is "move", the mobile agent moves to state $s_{t+1} \in \{c_i\}$ with probability $P(c_i)$. Otherwise, the agent moves to the "end" state.

Denote $B_t(s, z)$ as the cost incurred if the system state is $s \in S$ at t and the selected choice is $z \in Z(s)$ at t. It is defined by the following equation:

$$B_1(s, z) = \begin{cases} c_i + C_{travel}, \text{ if } s \in \{c_i\}, z = 0 \\ 2 \times C_{travel}, \text{ if } s \in \{c_i\}, z = 1 \\ 0, \qquad \text{if } s = \Theta \end{cases}$$

$$B_t(s, z) = \begin{cases} c_i, \qquad \text{if } s \in \{c_i\}, z = 0 \\ C_{travel}, \quad \text{if } s \in \{c_i\}, z = 1 \\ 0, \qquad \text{if } s = \Theta \end{cases} \quad, \ t = 2, \ldots N\text{-}1, \qquad (5)$$

$$B_N(s) = c_i$$

The cost is determined using these three equations corresponding to the first decision point, the last decision point and the decision points in between, as explained as follows. If the decision is "end", the offered price is paid. If the decision is "move", the traveling cost is paid. At the first shop, one more C_{travel} is paid because the cost of traveling to the first shop should be included. If the mobile agent reaches the last shop, it ends and pays at the price being offered at the last shop.

At a decision point t, the average accumulated cost for s_t and z_t can be found as follows (see [31][32] for details):

$$\Omega_t(s_t, z_t) = B_t(s_t, z_t) + \sum_{s_{t+1} \in S} \rho(s_{t+1} \mid s_t, z_t)\omega_{t+1}(s_{t+1}) \tag{6}$$

where $\omega_{t+1}(s_{t+1})$ is the lowest average accumulated cost for s_{t+1} at t+1. According to [31][32], we solve Eqn. 6 recursively by first setting $\omega_{N+1} = 0$. Let $Z_t^*(s_t)$ be the preferred choice for state s_t at t. Based on Eqn. 6, we can find $Z_t^*(s_t)$ by computing the average accumulated cost for all of the possible choices. The preferred choice is the one that gives the lowest average accumulated cost, and the corresponding value of the cost is used to determine $\omega_t(s_t)$.

Table 4 Values of the parameters used in the example

Parameter	Value(s)
Number of shops (N)	3
Decision points (t)	{1, 2, 3}
Prices (c_i)	{1, 2, 3}
Price distribution ($P(c_i)$)	Uniform (1/3)
System states (s)	{1, 2, 3, Θ}
Traveling cost (C_{travel})	0.1

Table 5 Transition probabilities of the example

s_t, z_t \ s_{t+1}	c_1=1	c_2=2	c_3=3	Θ
c_1=1, 1	1/3	1/3	1/3	0
c_1=1, 0	0	0	0	1
c_2=2,1	1/3	1/3	1/3	0
c_2=2, 0	0	0	0	1
c_3=3, 1	1/3	1/3	1/3	0
c_3=3, 0	0	0	0	1
Θ, 1	0	0	0	1
Θ, 0	0	0	0	1

The recursive process is best explained with a simple example. Table 4 shows the values of the parameters used in the following example. Table 5 shows the transition probabilities.

By using the backward induction algorithm (see [31][32], for details), the procedures are shown as follows:

Step 1: With t = 4, we have $\omega_4(s_4) = 0$, where $s_4 \in S$.

Step 2: With t = 3, the agent has reached the last shop so it must “end”. Therefore $\omega_3(s_3) = B_3(s_3,0)$.

Step 3: With t = 2, compute $\Omega_2(s_2, z_2)$ accordingly. Then choose the option with the minimum average accumulated cost, e.g.,

$$\begin{aligned}\Omega_2(3,0) &= B_2(3,0) + \rho(1\,|\,3,0)\omega_3(1) + \rho(2\,|\,3,0)\omega_3(2) + \rho(3\,|\,3,0)\omega_3(3)\\ &= 3 + 0 + 0 + 0 = 3\end{aligned}$$

$$\begin{aligned}\Omega_2(3,1) &= B_2(3,1) + \rho(1\,|\,3,1)\omega_3(1) + \rho(2\,|\,3,1)\omega_3(2) + \rho(3\,|\,3,1)\omega_3(3)\\ &= {}^{1}\!/_{10} + {}^{1}\!/_{3} + {}^{2}\!/_{3} + {}^{3}\!/_{3} = 2\,{}^{1}\!/_{10}\end{aligned}$$

As $\Omega_2(3,1)$ gives the minimum value so $Z_2{}^*(3) = 1$. Hence, we have $\omega_2(s_2) = \Omega_2(3,1)$.

Step 4: With t = 1, compute $\omega_1(s_1)$, similar to step 3.

All of the expected costs and preferred choices are summarized in Table 6.

Table 6 Expected costs and preferred choices (Model I)

s	**1**	**2**	**3**	**Θ**
$\Omega_3(s,0)$	1	2	3	0
Ω_2 (s,0)/Ω_2 (s,1)	1 / 2.1	2 / 2.1	3 / 2.1	0
Ω_1 (s,0)/Ω_1 (s,1)	1.1 / 1.9	2.1 / 1.9	3.1 / 1.9	0
ω_3 (s)	1	2	3	0
ω_2 (s)	1	2	2.1	0
ω_1 (s)	1.1	1.9	1.9	0
Z_3*(s)	0	0	0	I
Z_2*(s)	0	0	1	I
Z_1*(s)	0	1	1	I

Note: I=Infeasible

5.2.2. Model II

Similar to Model I, the price of a product is c_i with a known probability distribution $P(c_i)$, where $c_i > 0$, $i = 1, 2, \ldots n$, and $c_i < c_{i+1}$ for all i. The decision points are the same as those in Model I. In Model II, the agent can return to a previous shop to buy the product at the lowest price. The system states are represented by the price of the product c_i and the current minimum price of the product m. The set of states at each shop is therefore denoted as $S = \{c_i,m\} \cup \{\Theta\}$.

There are also two choices in each state: "move" and "end". When the agent decides to continue shopping, the minimum cost should be updated as $\min(c_{i,t},m_t) = m_{t+1}$ before moving to the next state with probability $P(c_i)$. When the agent locates at the shop 1, the minimum cost is not recorded. The minimum price at the shop 2 should be based on the price offered by the first shop. The state transition probability is given by the following expression for $t = 2, \ldots N$-1.

$$\rho(s_{t+1} \mid s_t, z_t) = \begin{cases} P(c_{i,t+1}), & \min(c_{i,t}, m_t) = m_{t+1}, z_t = 1 \\ 1, & (s_t \in \{c_i, m\}, s_{t+1} = \Theta, z_t = 0) or (s_t = s_{t+1} = \Theta, z_t = 1) \\ 0, & \text{otherwise,} \end{cases}$$
$$\rho(s_2 \mid s_1, z_1) = \begin{cases} P(c_{i,2}), & c_{i,1} = m_2, z_1 = 1 \\ 1, & (s_1 \in \{c_i, m\}, s_2 = \Theta, z_1 = 0) or (s_1 = s_2 = \Theta, z_1 = 1) \\ 0, & \text{otherwise,} \end{cases} \quad (7)$$

At *t*, the cost for each state, $B_t(s, z)$, is defined as follows:

$$B_1(s, z) = \begin{cases} c_i + C_{travel}, & \text{if } s \in \{c_i, m\}, z = 0 \\ 2 \times C_{travel}, & \text{if } s \in \{c_i, m\}, z = 1 \\ 0, & \text{if } s = \Theta \end{cases}$$
$$B_t(s, z) = \begin{cases} c_i, & \text{if } s \in \{c_i, m\}, c_i \le m, z = 0 \\ m + C_{travel}, & \text{if } s \in \{c_i, m\}, c_i > m, z = 0 \\ C_{travel}, & \text{if } s \in \{c_i, m\}, z = 1 \\ 0, & \text{if } s = \Theta \end{cases} \quad (8)$$
$$B_N(s) = \begin{cases} c_i, & \text{if } s \in \{c_i, m\}, c_i \le m \\ m + C_{travel}, & \text{if } s \in \{c_i, m\}, c_i > m \end{cases}$$

where $t = 2, 3, \dots N$-1.

The major difference in the cost equation between Model I and Model II is that in Model II the minimum cost is paid. This means that if the agent decides to end, it will buy the product at the previously visited shop that offers the lowest price. Note that there is a traveling cost of C_{travel} to return to the shop. Similarly, the minimum average accumulated cost $\omega_t(s_t)$ can be obtained by Eqn. 6.

We again explain this model with a simple example. The number of shops, the price distribution, and the traveling cost are the same as those in the previous example. The set of states are S = {(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3), Θ}. Based on the above calculation, the expected costs and preferred choices are summarized in Table 7.

Table 7 Expected costs and preferred choices (Model II)

$s = (c_i, m)$	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(3,3)	Θ
$\Omega_3(s,0)$	1	1	1	1.1	2	2	1.1	2.1	3	0
$\Omega_2(s,0)$ /	1 /	1 /	1 /	1.1 /	2 /	2 /	1.1 /	2.1 /	3 /	0
$\Omega_2(s,1)$	1.167	1.167	1.167	1.167	1.8	1.8	1.167	1.8	2.1	
$\Omega_1(s,0)$ /	1.1 /	1.1 /	1.1 /	2.1 /	2.1 /	2.1 /	3.1 /	3.1 /	3.1 /	0
$\Omega_1(s,1)$	1.267	1.267	1.267	1.733	1.733	1.733	1.833	1.833	1.833	
$\omega_3(s)$	1	1	1	1	2	2	1	2	3	0
$\omega_2(s)$	1	1	1	1.1	1.8	1.8	1.1	1.8	2.1	0
$\omega_1(s)$	1.1	1.1	1.1	1.733	1.733	1.733	1.833	1.833	1.833	0
$Z_3{}^*(s)$	0	0	0	0	0	0	0	0	0	I
$Z_2{}^*(s)$	0	0	0	0	1	1	0	1	1	I
$Z_1{}^*(s)$	0	0	0	1	1	1	1	1	1	I

Note: I=Infeasible

5.3. Results and Discussion

5.3.1. Analytical Results

In the analysis, the following assumptions are made:

1. As an example, $P(c_i)$ is estimated by a "discrete" normal distribution with a mean μ and standard deviation σ.
2. The prices are between μ-3σ and μ+3σ. Therefore, the probability is zero outside the above price range.
3. The mean cost, μ, is set as \$400. The standard deviation σ and the traveling cost C_{travel} are varied in the following analysis.

It is possible to use these models and the backward induction algorithm to identify the preferred choices and the minimum costs. In the following, we will also compare the results for Model I and Model II.

5.3.1.1 Model I

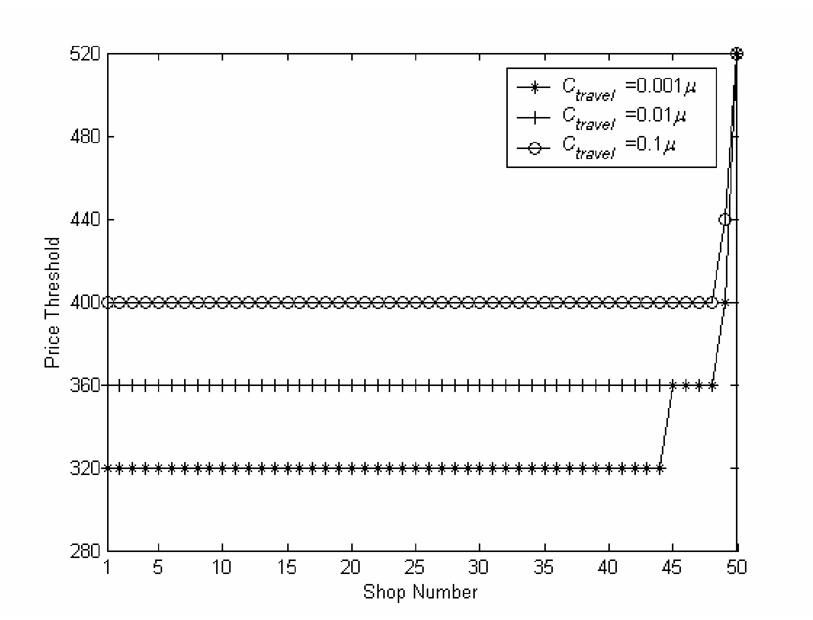


Figure 35 Preferred choices for different values of C_{travel} (σ = 0.1μ)

Figure 35 shows the preferred choices at each shop of Model I, with $\sigma = 0.1\mu$ when C_{travel} is varied. The agent prefers to travel to the next shop when the price of the product is above the lines (or price thresholds). When the traveling cost is very high, the agent should not visit other shops unless the current price is much higher than the mean price.

Figure 36 shows the average cost paid by an agent when there exists different numbers of shops. The number of shops the agent will visit may be smaller than the number of shops existing because the agent can stop visiting based on the decision policy. We have considered five different cases of C_{travel}. The average cost for Model I is calculated as follows:

$$C_{average} = \sum_{i=1}^{n} \omega_1(c_i)P(c_i) \qquad (9)$$

For instance, in the above example, the average cost is $1.1 \times {}^{1}\!/_{3} + 1.9 \times {}^{1}\!/_{3} + 1.9 \times {}^{1}\!/_{3} = 1.63$.

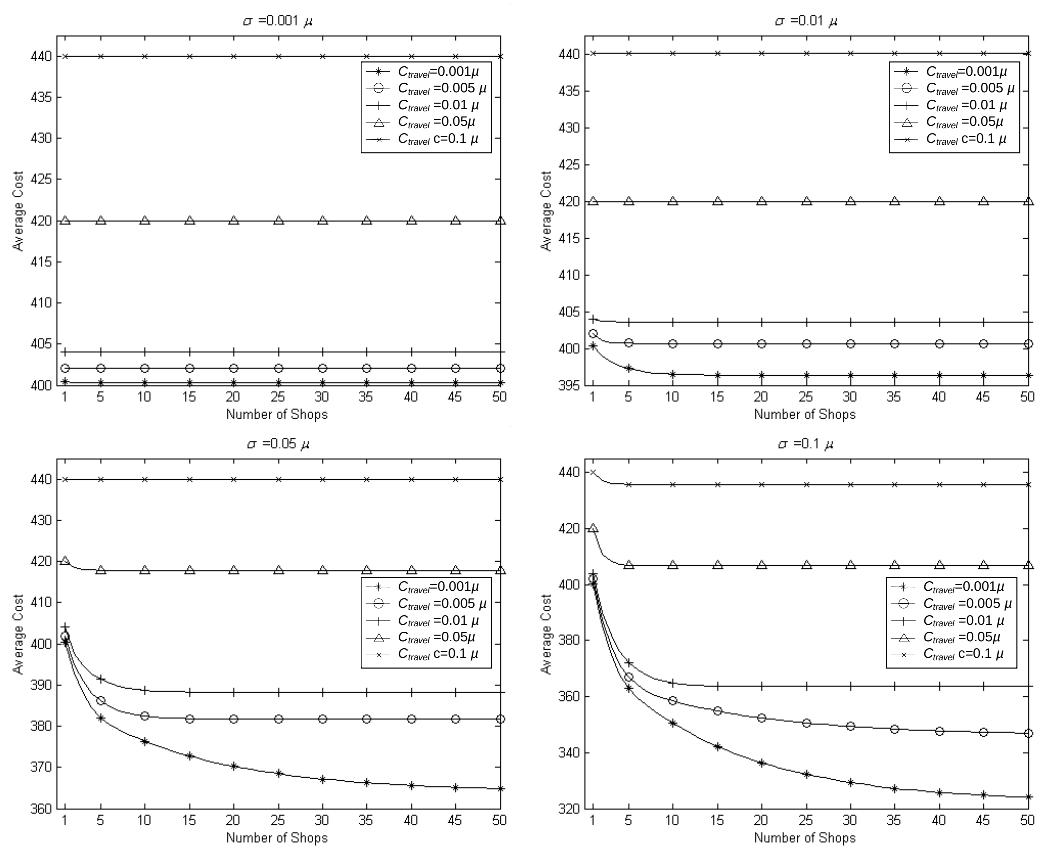


Figure 36 Average cost vs. number of shops (Model I)

According to the figures, the average cost is independent of the number of shops when the standard deviation is low. If the difference in price is very small, it is not worthwhile for the agent to travel to the next shop. In this case, the agent can simply visit the first shop and buy the product. Therefore, the average cost is equal to the sum of C_{travel} and the expected price of the product. When the price difference is large, the average cost decreases because the agent has a better chance of buying the item at a lower price. It can be seen that the average cost drops as C_{travel} falls. As C_{travel} is low, it is cost-effective to visit more shops. However, the average cost levels off when a certain threshold is reached. In other words, it is possible to get the lowest average cost after visiting a certain number of shops.

5.3.1.2 Model II

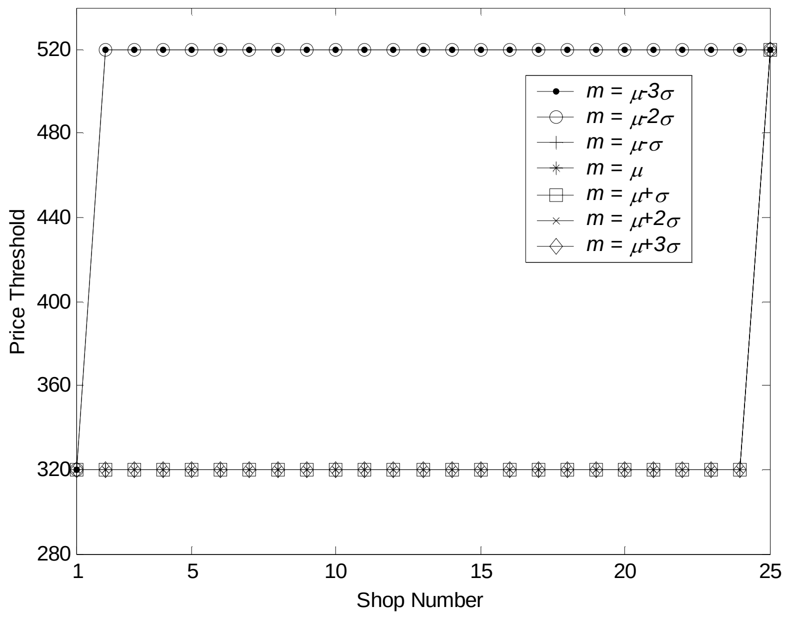


Figure 37 Preferred choices for different values of *m* with $\sigma = 0.1\mu$, $C_{travel} = 0.001\mu$ (Model II)

Figure 37 shows the preferred choices at each shop in Model II with $\sigma = 0.1\mu$ and $C_{travel} = 0.001\mu$ when the same mean price is applied as in Model I. Each line in the figure represents a different minimum price recorded. The agent prefers to travel to the next shop if the price offered by the current shop is above the line (i.e., the price threshold). As the maximum price is \$520, it can be seen that if the minimum cost (*m*) is μ-2σ or below, the agent should not travel to the next shop except at shop 1. As the minimum cost is the same as the current shop price, the current shop price becomes the main consideration in the agent's decision. Thus, the price threshold at shop 1 is the same as the one in Model I. If the minimum cost is μ-σ or above and the price offered by the current shop is greater than \$320, the agent should travel to the next shop.

In Model II, initially the minimum price should be set based on the price offered by the first shop. The average cost in Model II is calculated as follows:

$$C_{average} = \sum_{i=1}^{n} \omega_1(c_i, c_i) P(c_i) \tag{10}$$

For instance, in the aforementioned example, the average cost is $1 \times {}^1\!/_3 + 1.733 \times {}^1\!/_3 + 1.833 \times {}^1\!/_3 \approx 1.522$.

Figure 38 shows the average cost for different numbers of shops under different standard deviations σ and different traveling costs, C_{travel}. It can be seen that the average cost for Model II is very similar to that for Model I.

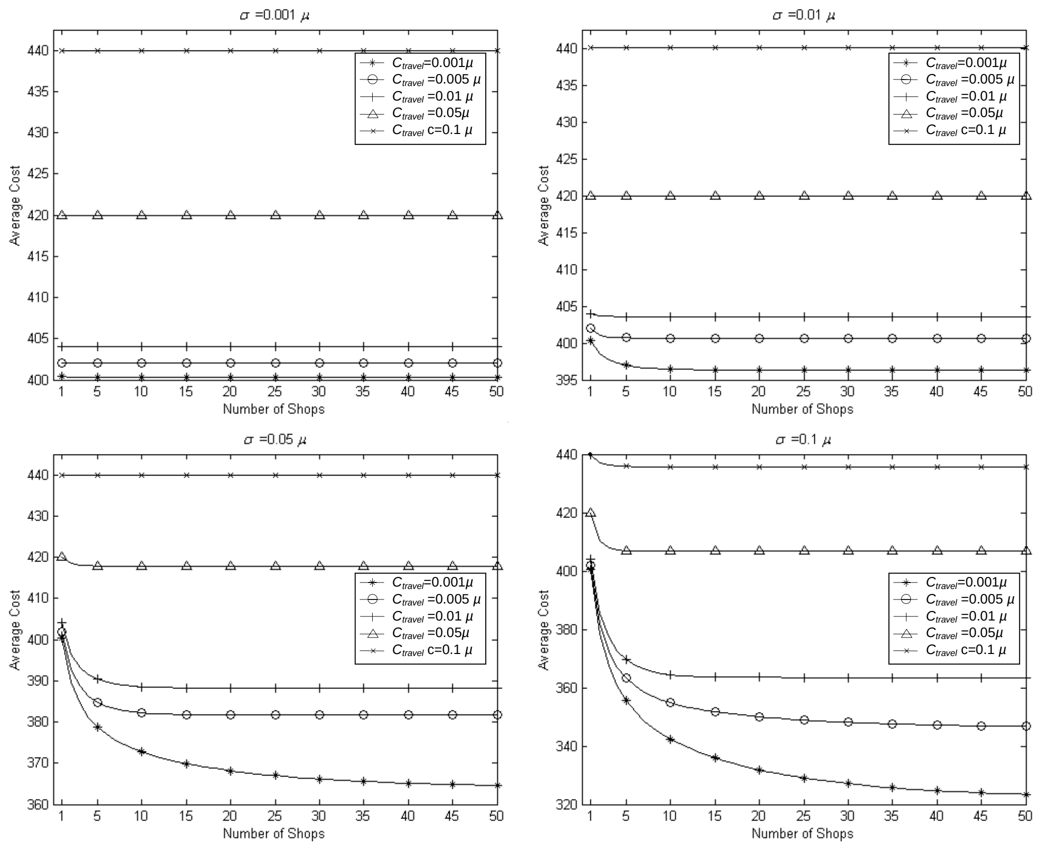


Figure 38 Average cost vs. number of shops (Model II)

5.3.1.3 Models with Real Price Distribution

We have also conducted further analysis using as the experimental data the bestselling computer books given in Best Web Buys (www.bestwebbuys.com). We also obtained the real price information in priceSCAN.com (www.pricescan.com) for the analysis. $P(c_i)$ is estimated by a "discrete" distribution with mean μ and standard deviation σ based on the price information of the bestselling computer books (see Table 8 for the first 5 of the top 20 computer books). The price range is also from μ-3σ to μ+3σ and the distribution is found by counting the number of shops in each price level.

Table 8 The computer books found from priceSCAN.com (http://www.pricescan.com)

Rank	Book name	μ (\$)	σ(\$)	No. of shop found
1	High-Speed Signal Propagation: Advanced Black Magic (2002)	85.51	9.12	17
2	High-Speed Signal Propagation: Advanced Black Magic (1993)	87.33	9.27	16
3	Performance Assessments for Adult Education: Exploring the Measurement Issues Report of a Workshop	11.12	1.93	17
4	Java: How to Program	81.88	9.25	16
5	Introduction to Algorithms	76.69	5.4	15

Note: All of the above prices are in U.S. dollars

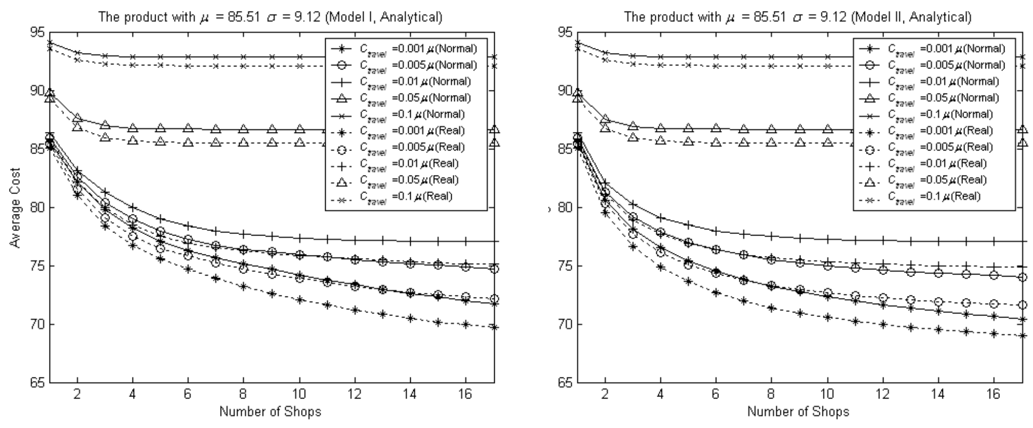


Figure 39 The average cost for different numbers of shops of purchasing the book "High-Speed Signal Propagation: Advanced Black Magic" under different price distribution and different traveling costs, C_{travel} (prices based on the information at "http://www.pricescan.com")

Figure 39 shows the average cost of purchasing the book "High-Speed Signal Propagation: Advanced Black Magic" under different price distributions and different travel costs when there are certain number of shops. The two models give the similar result that the agent pays at a lower price when the real price distribution is used. Nonetheless, the average cost as calculated with a normal distribution is only one to two dollars higher than that calculated with a real distribution. As shown in Figure 40 and Figure 41, we have calculated the average percentage difference in the average cost of the two models between the normal and real distribution when twenty books are considered. The percentage difference is below 5% and 6% for Model I and Model II, respectively. The difference rises dramatically when the number of shops visited is more than 15, because most books can be bought after visiting fifteen shops or less. The % difference is larger when the traveling cost is small. It can also be seen that the % difference increases slowly when the number of shops the agent would visit increases.

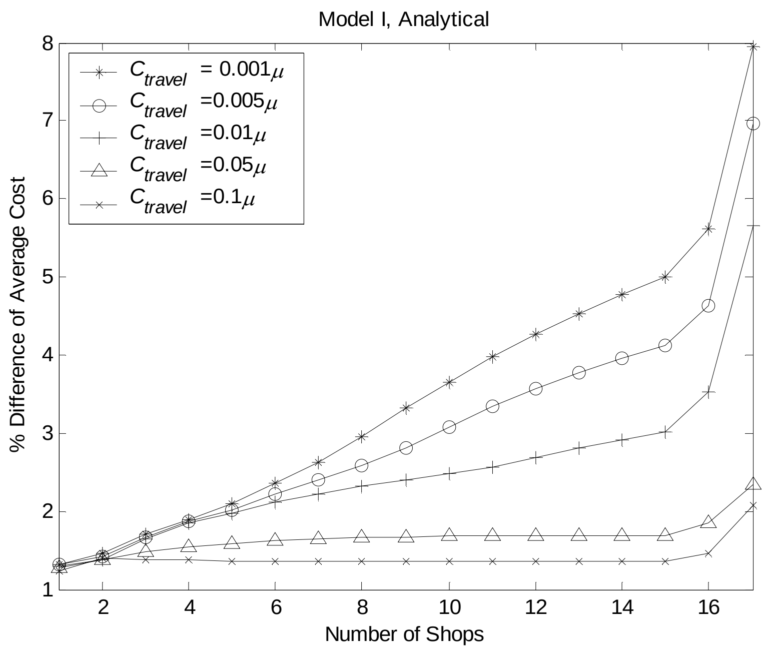


Figure 40 The % difference between normal and real price distribution (Model I, Analytical)

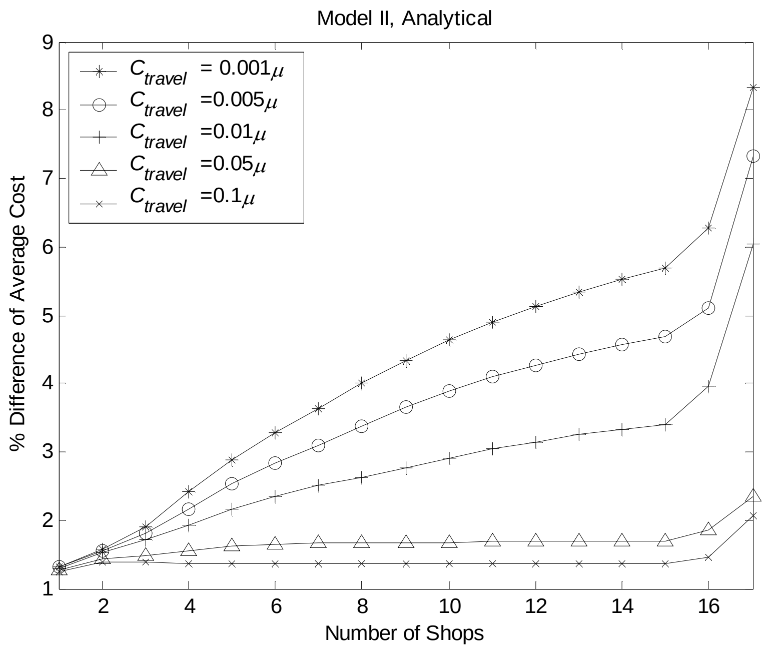


Figure 41 The % difference between normal and real price distribution (Model II, Analytical)

5.3.1.4 Comparison among Models

Besides analyzing the models individually, we will also compare the two models with the model in [6]. In [6], a mathematical model is developed to determine the number of mobile agents to be dispatched in parallel to get the price of an item for price comparison. When dispatching a mobile agent, the cost C_{travel} should be paid. After retrieving the results from all agents, the system does the price comparison. The average cost for this parallel model can be found as follows:

$$\begin{aligned} P_n(p_j) &= P_{n-1}(p_j)\sum_{k=j}^{\infty} P(p_k) + P(p_j)\sum_{k=j+1}^{\infty} P_{n-1}(p_k) \\ C_n &= nC_{travel} + \sum_{j=1}^{\infty} p_j P_n(p_j) \end{aligned} \tag{11}$$

where $P_n(p_j)$ is the probability that the lowest price is, p_j, after the n-th agent has returned and C_{travel} is the cost of sending an agent.

Referring to the above equations, the average cost decreases as the number of agent dispatched increases initially. After reaching the optimal number of agents dispatched, the average cost rises gradually. In the following comparisons, the optimal number of agents is sent in parallel to achieve the lowest average cost.

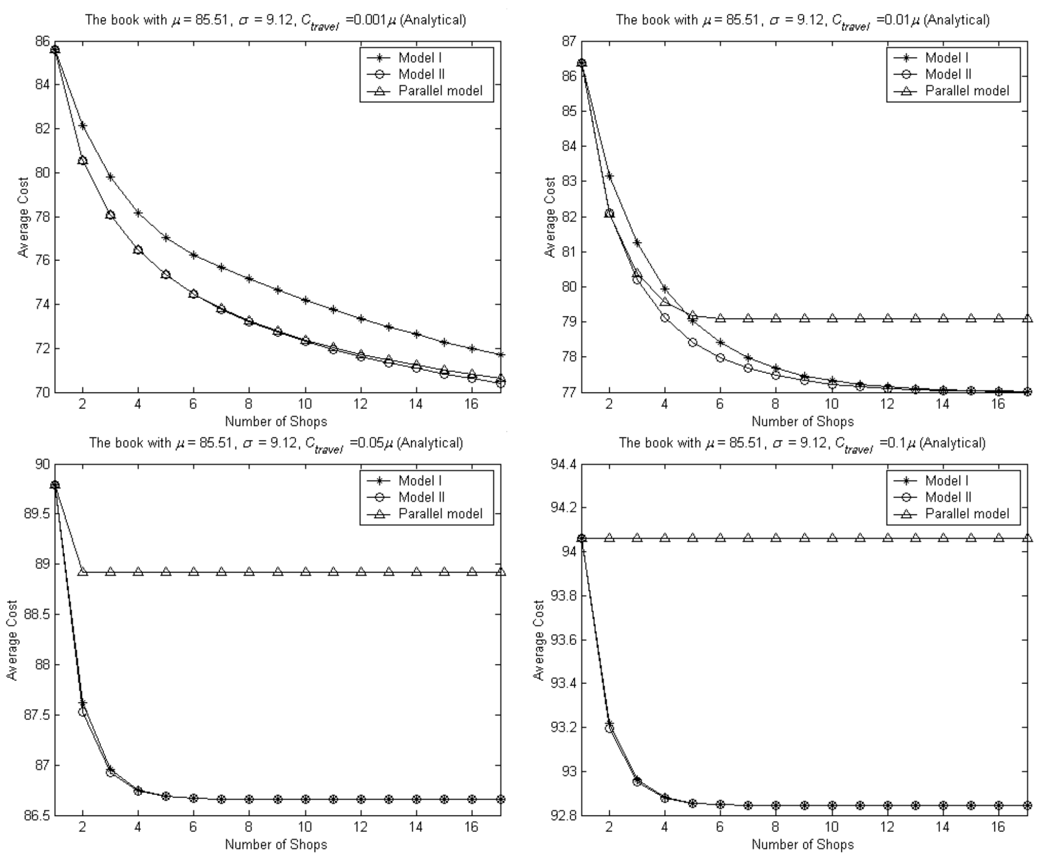


Figure 42 Average cost vs. no. of shops (3 models, Analytical)

Figure 42 shows the comparison of average cost for the parallel model, Model I and Model II. In our comparison, the best selling computer book “Final Fantasy X-2: Official Strategy Guide” is assumed to be bought. The number of shops the agent would visit is given in Table 8 and we determine the transition probability with the normal price distribution based on the corresponding mean price and standard derivation.

According to the figures, after reaching a certain number of shops, the average cost becomes steady. This constant cost is the lowest average cost. The agents in the parallel model can achieve the lowest average cost when the system sends an optimal number of agents. The top left figure illustrates the case for a low traveling cost. We can see that the average cost for Model I is higher than that for Model II and the parallel model. The result of Model I is worse than that of the parallel model because the low traveling cost encourages the agent to travel to next shop but it cannot return to the previous shops. Note that it may have a higher probability of paying a higher price in the current shop. When the traveling cost increases, the average cost for Model I and Model II is smaller when compared with that of the parallel model. For the parallel model, a fixed number of agents are dispatched. Even though the minimum price is found, a fixed amount of traveling cost is incurred. For Model I and Model II, only one agent is sent and it can decide to stop traveling if a certain price is reached. The average cost for the two models is therefore smaller owing to the lower traveling cost. When comparing Model I and Model II, the lowest average cost for Model II can be achieved after visiting a smaller number of shops. The bottom right figure shows the result where the traveling cost is very high. We can see that the average cost for the two models are almost the same. In fact, the agent in Model II pays only a few cents less than the agent in Model I. This is because the traveling cost is more significant than the minimum price found (i.e., *m*) in Model II. In general, Model II performs better because the agent can always pay the lowest price found from the previous shops.

5.3.2. Experimental Results

We have applied the above models to carry out an analysis. Furthermore, we have also validated the analytical results by means of simulations. In the simulations, there are a number of shops based on the information obtained from Best Web Buys. An agent is created to visit the shops and makes the purchase decision according to the policy determined by the models. For evaluation purposes, 1000 simulations were performed to obtain the experimental results.

5.3.2.1 Models with Real Price Distribution

In the simulations, we determine two decision policies: one with the normal distribution and the other with the real price distribution. In the simulation, an agent uses the two decision policies to purchases twenty books. From this, we find the average cost of each simulation and the percentage difference in the average cost between the two distributions. We then take the average based on how many shops the agents must travel to. The differences between the two distributions are shown in Figure 43 and Figure 44. The simulation result is similar to the analytical result. It can be seen that the percentage difference is higher when the number of shops visited increases. The simulation results show that the percentage difference in average cost is below 2.5% for the two models. This indicates that the result of using a normal price distribution is very similar to the result of using a real price distribution. In other words, the normal price distribution is a good estimation of the real price distribution. This is not likely found in the real world.

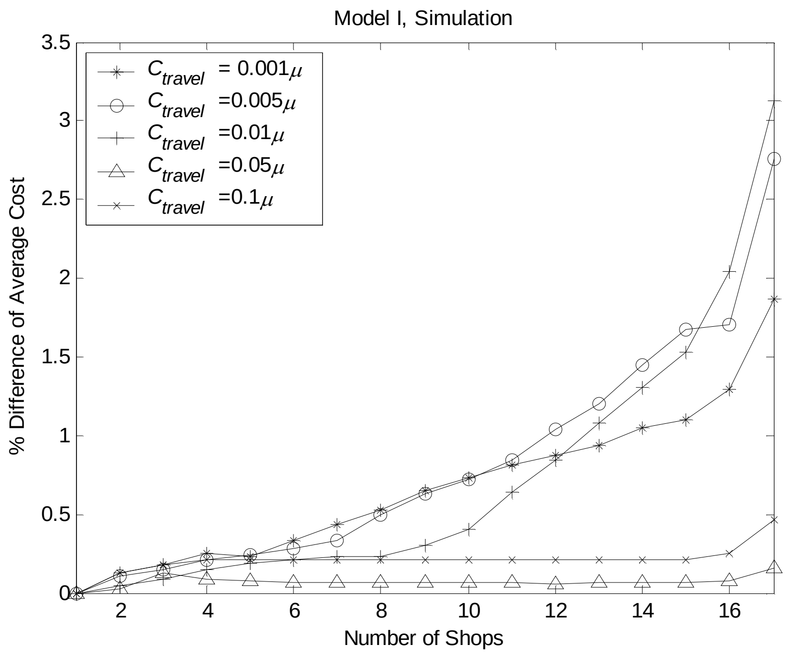


Figure 43 The % difference between normal and real price distribution (Model I, Simulation)

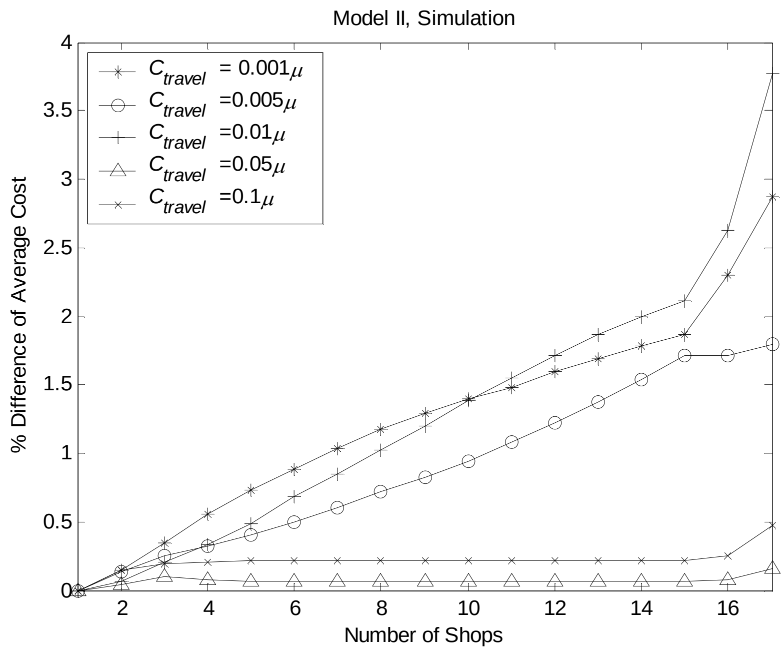


Figure 44 The % difference between normal and real price distribution (Model II, Simulation)

5.3.2.2 Comparison of the Models

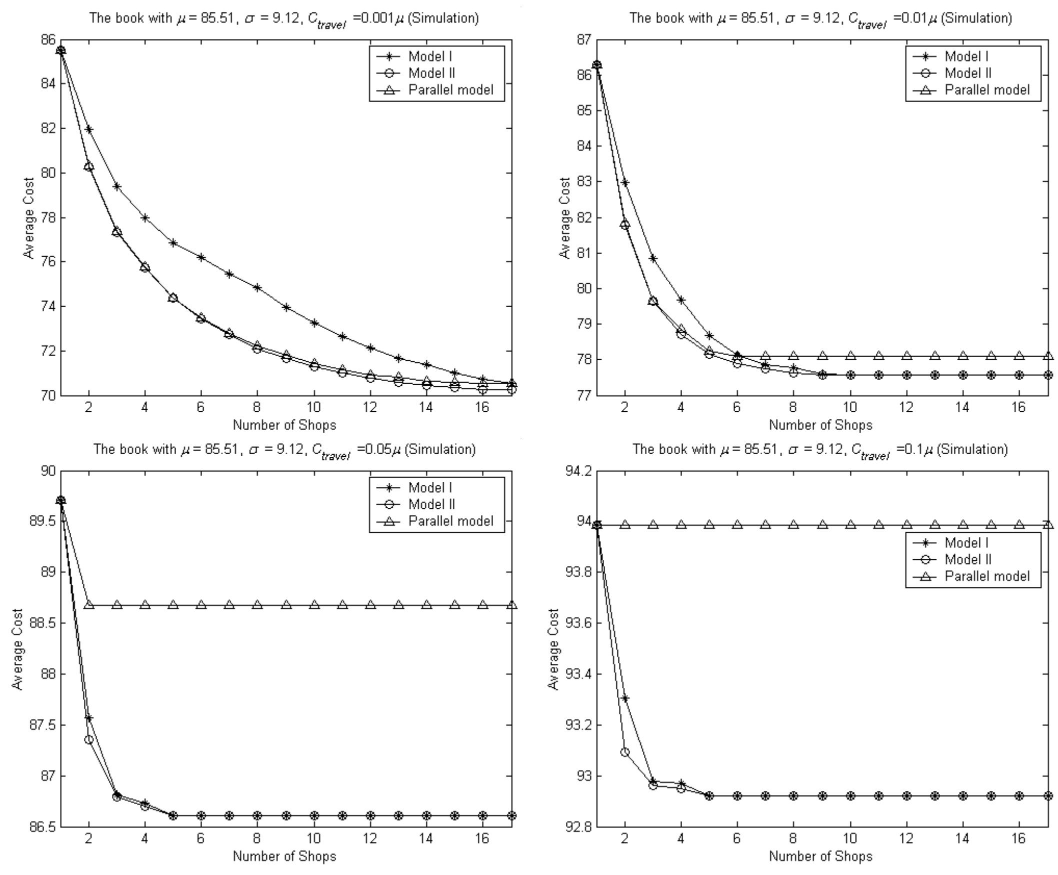


Figure 45 Average cost vs. no. of shops (3 models, Simulation)

In the simulations, the optimal number of agents to be dispatched in the parallel model is calculated using Eqn. 11. As the normal price distribution can give a similar result as the real price distribution, the normal price distribution is used to determine the decision policy. The result obtained is similar to the analytical one.

5.3.3. Discussions

As shown in Figure 35, the agent prefers to travel to the next shop when the price of the product is above the price thresholds. The price thresholds increase when approaching the last shop but they remain constant after visiting the first few shops. For simplicity, the figures show the relationship between the price thresholds in the first few shops, mean price (μ), standard deviation (σ) and traveling cost (C_{travel}). Figure 46 presents the price threshold with different μ and σ in Model I. We can see that the price threshold is independent of μ.

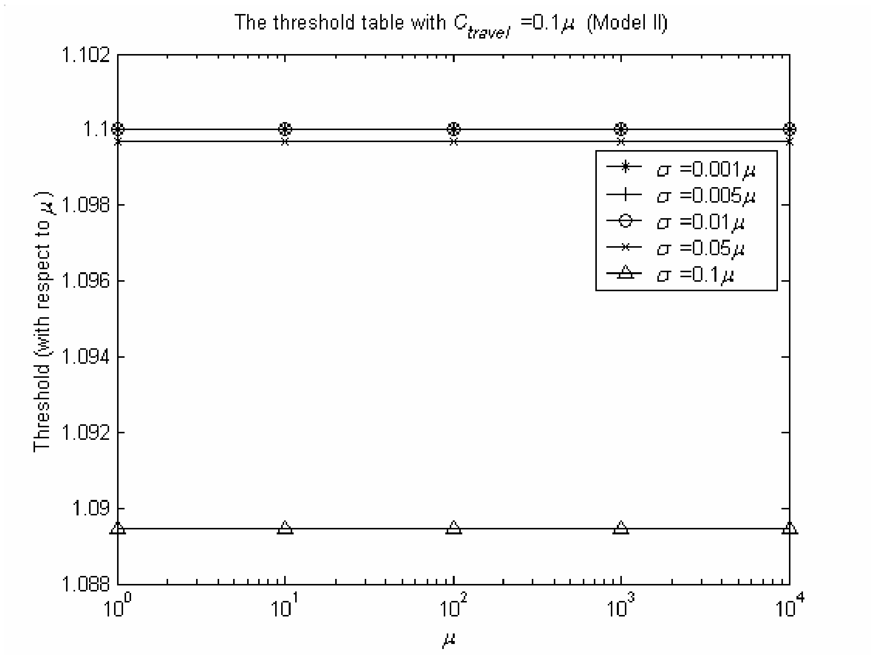


Figure 46 The price threshold using different μ and σ (Model I)

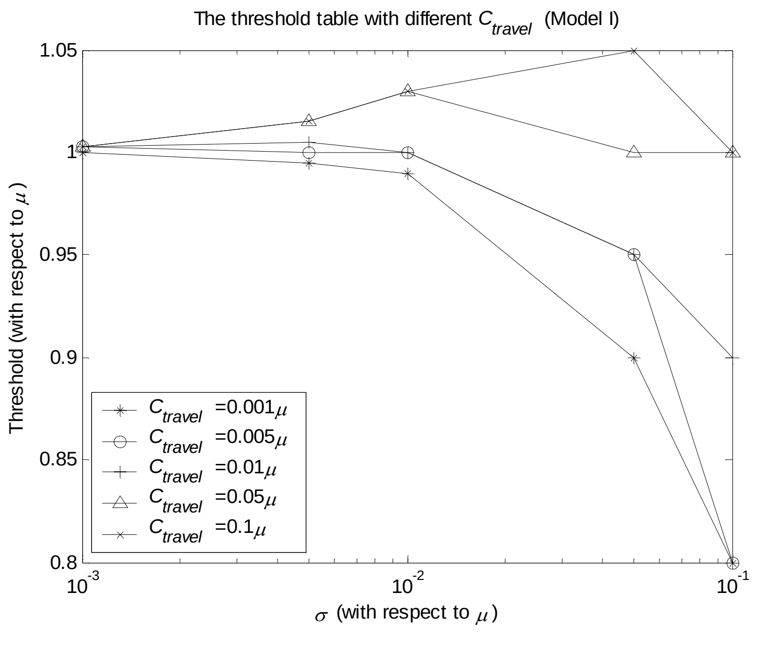


Figure 47 The price threshold using different σ and C_{travel} (Model I)

Figure 47 shows the relationship between the price thresholds, σ and C_{travel} with respect to μ in Model I. When the standard deviation is very low, the agent prefers to buy the product with the mean price. If the standard deviation is high and the traveling cost is 0.01μ or below, the agent has a higher probability of paying a lower price for the product. Furthermore the low traveling cost encourages the agent to visit more shops. This means that it can continue shopping until the price is below the mean price. When the traveling cost is high, the agent chooses to stop even though the price is above the mean price.

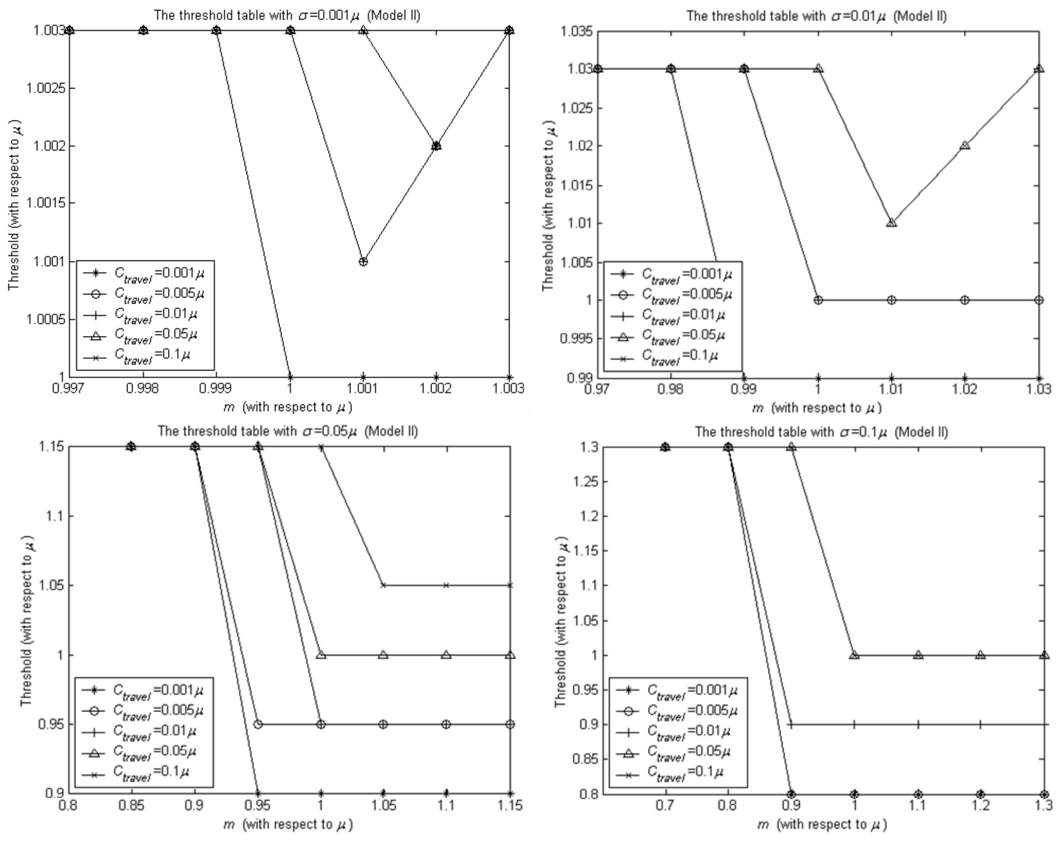


Figure 48 The threshold table with different *m*, σ and C_{travel} (Model II)

Figure 48 shows the price threshold in Model II. Model II allows the agent to buy the product in the previous shop which offers a lower price. In general, the agent can stop at the shop which provides a higher price if the current minimum product price (*m*) is low. When *m* increases, it prefers to stop at the shop that offers a lower price. The top left figure shows the price threshold with low standard deviation. When *m* is larger than the mean price, μ., the agent prefers to buy at the price which is equal to *m* except when C_{travel} is 0.001μ. As the price difference is very small, it is not worth to continue shopping. When the price difference is large as shown in the bottom right figure, the agent has a better chance of buying the item at a lower price. Therefore, it is more cost-effective to visit more shops when the traveling cost is below 0.005μ. As shown in the figure, it will stop at the shop which offers the price 20% lower than the mean price. When the traveling cost is above 0.05μ, the agent should stop at the shop that offers the mean price if *m* is greater than the mean price.

5.4. Summary

In this chapter, we have investigated a price-comparison problem for B2C MAGICS. Based on Markov Decision theory, two mathematical models have been formulated to determine the optimal decision policy for the price-comparison problem by using the backward induction algorithm. While it may also be possible to solve the problem by using a cost function to find the optimal number of agents to send, it is of interest to investigate the Markov decision models because decisions can be made adaptively (i.e., in accordance with the system state). The models also take into account various price distributions and the traveling costs. Analytical results are presented to show how the average cost is affected by various parameters. We have used these models to carry out analyses using a normal price distribution and real price information. We have found that the normal distribution can provide a good estimation of the real price distribution in general. We also conducted a set of experiments/ simulations to validate the analytical results and have found that the price threshold is independent of the mean price. The results should give us valuable insights into the design of price comparison services for B2C MAGICS.

CHAPTER 6
FUZZY MARKOV DECISION PROCESS (*f* MDP) FOR MULTI-ATTRIBUTE PRODUCT COMPARISON

With the advance of Internet technologies, information extraction has become much easier while the volume of information being obtained has increased exponentially. Too many choices make it too difficult for agents to analyze and process each alternative in order to obtain the best solution [41]. In other words, the analysis often incurs high searching and processing costs. In order to minimize these costs, there is a strong need to build an effective price comparison model such as the one presented in Chapter 5. The model tries to determine the optimal decision policy in order to buy a product at the minimum expected cost under certain assumptions. However, in many buying scenarios, a consumer may need to evaluate multiple attributes before making a final decision.

When a software agent is used to handle product comparison, Artificial Intelligent (AI) technologies can be used to consider multiple attributes in order to find the best offer. In this "complex" buying process, some attributes might have precise values and the decision can be modeled by using Boolean Logic. For example, "If the price of a book is higher than $300, the user will not buy it."

On the other hand, some attributes can only be described in imprecise terms. For example, “If the size of a T-shirt is small, then the user will buy it.” In this case, “size” is a linguistic variable and “small” is its linguistic value. To evaluate products with these kinds of attributes, Fuzzy Logic can be used [28][43].

In this chapter, we will extend the previous model (described in Chapter 5) with the aim of handling multiple attributes. An innovative model so-called “*f*MDP” (Fuzzy Markov Decision Process) will be introduced for multi-attribute product comparison. In particular it involves the integration of Fuzzy Logic and Markov Decision Theory.

6.1. Concept of Fuzzy Logic

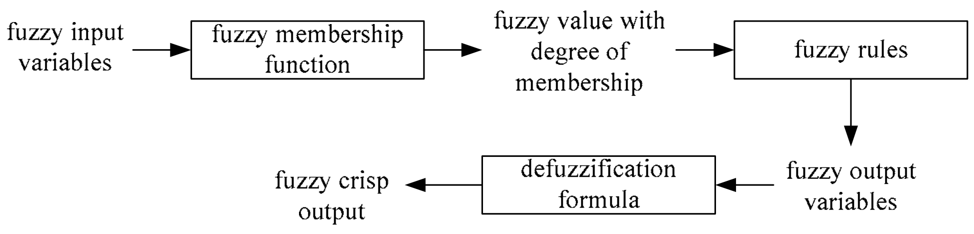


Figure 49 The general fuzzy system

The *f*MDP model proposed in this chapter is based on the integration of two theories. One is Markov Decision Theory (MDT) which is discussed in Chapter 5. The other one is Fuzzy Logic. Fuzzy Logic is introduced by Zadeh in the 1960s as a mean to handle the grey area between “completely yes” and “completely no” [44][45]. The theory of fuzzy sets is an extension of the conventional set theory by introducing the idea of the “Degree of membership”. Under this theory, any items can be described either by discrete values or continuous values. In a general fuzzy system (see Figure 49), the fuzzy input variables have a set of fuzzy values which are defined as fuzzy sets in the corresponding universes of discourse. Every fuzzy value can be mapped into a corresponding degree of membership value by using the so-called fuzzy membership functions. In a typical decision support system, fuzzy rules are used to map the input instance to the output outcome. These rules are often expressed in “if-then” format with “and”, “or” and “not” operators. To convert the fuzzy output variables into a single crisp output, the process of defuzzification is needed by combining the variables with different weights.

6.2. *f*MDP Formulation

In this chapter, we formulate a Fuzzy Markov Decision Process (*f*MDP) for multi-attribute product comparison. It can be viewed as the extension of the price-comparison model discussed in Chapter 5. As illustrated in the last chapter, Model II gives a better performance than Model I. Hence, *f*MDP will be developed based on Model II. The general scenario of the model is described as follows. Assume that there are N shops and that a shopping agent will visit them in sequence. At each shop, the shopping agent must decide whether to buy the item or to travel to the next shop by considering the product attributes. If the shopping agent decides to buy the item, it has two options to choose. The first option is to buy at the current shop and hence it will not visit other shops. The second option is to return to the previous shop that provides a better offer. In the case of returning to the previous shop, a constant traveling cost C_{travel} will be paid. Otherwise, it will move to the next shop and pay a traveling cost C_{travel}. At the next shop, the shopping agent will make the same decision again. By following the approaches/notations used in [31][32], we formulated the *f*MDP model as follows. Table 9 gives the notations used in the model.

Table 9 Notations used in the *f*MDP model

Variable	Meaning
M	The total number of attributes
A_i	The fuzzy set of the *i*-th attribute
V_i	The set of linguistic values describing attribute
X_i	A universe of discourse
$\mu_{\underline{A_i\_v}}$	The fuzzy membership function of the *i*-th attribute with a linguistic value *v*
ς	The attribute score of a product
φ	The maximum attribute score recorded
κ	The preference category (*DL*, *ID*, *I*, *L*)
t	The decision point *t*
S	The set of states
s	The system state
Θ	The end state
Z	The set of choices
z	The selected choice
ρ	The state transition probability
O_t	The overall attribute score at *t*
C_{travel}	The traveling cost
Ω_t	The average accumulated cost at *t*
ω_t	The minimum average accumulated cost at *t*

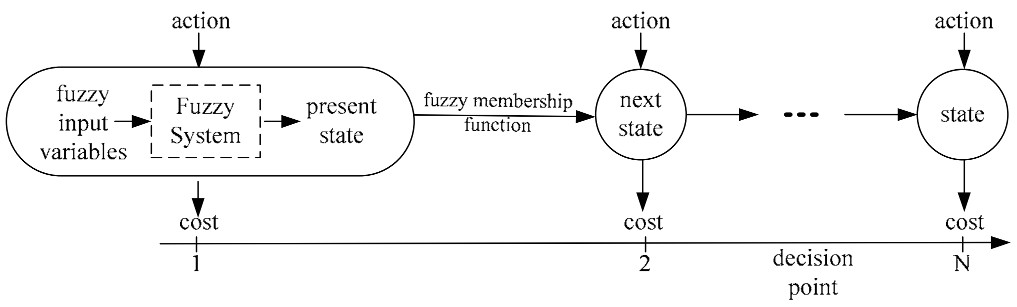


Figure 50 Architecture of Fuzzy Markov Decision Process (fMDP) for Multi-attribute Product Comparison

The fMDP model considers products with imprecise attributes and provides a decision rule based on the Markov Decision Theory and the fuzziness measurement of the set members. Figure 50 shows the architecture of the fMDP model. In this model, the fuzzy outputs are used to represent the states of MDP and the fuzzy membership function is applied when defining the transition probability. We assume that there are M attributes for comparison. The i-th attribute is denoted as A_i where i = 1, 2, … M. Instead of using the exact measurement, linguistic terms are used to describe the requirement attributes for the product. Corresponding to each product attribute, its linguistic values are defined by fuzzy sets with relevant membership functions.

$\underline{A_i}$ is the fuzzy set of the i-th attribute and V_i is a set of its linguistic values. The fuzzy set is defined as:

$$\underline{A_i\_v} = \{(x, \mu_{\underline{A_i\_v}}(x)) \mid x \in X_i, \mu_{\underline{A_i\_v}}(x) \in [0,1]\} \qquad (12)$$

where $\mu_{\underline{A_i\_v}}(x)$ is the fuzzy membership function which maps the members of X, x, to a membership ranging between 0 and 1. For example, in describing the size of a T-shirt, the term “small”, “medium” and “large” are used. The “length of arm" is defined as the degree of size. Figure 51 shows the membership functions defined by Eqn. 13 and the way of determining the membership values with the length of arm. For example, if the length of arm is 110, it can be interpreted as 0.135 degree of "Small" and 0.607 degree of "Middle".

A_1 = size, V_1 = {small, medium, large} and X_1 = length of arm

$$size = \{small, medium, large \mid \forall \mu_{\underline{size\_small}}(x), \mu_{\underline{size\_medium}}(x), \mu_{\underline{size\_large}}(x) \in [0,1]\}$$

$$\begin{aligned} \mu_{\underline{size\_small}}(x) &= \begin{cases} 1 & \text{if } 0 \le x < 100 \\ N(x,100,5) & \text{if } 100 \le x \le 115 \\ 0 & \text{otherwise} \end{cases} \\ \mu_{\underline{size\_medium}}(x) &= N(x,115,5) \\ \mu_{\underline{size\_large}}(x) &= \begin{cases} 1 & \text{if } x > 130 \\ N(x,130,5) & \text{if } x \le 130 \end{cases} \end{aligned} \tag{13}$$

where $N(x,\alpha,\beta) = e^{\frac{(x-\alpha)^2}{-2\beta^2}}$

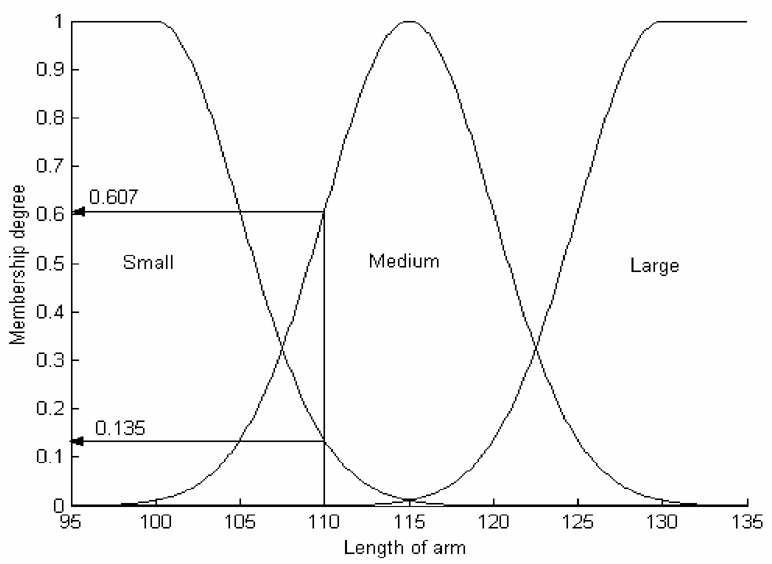


Figure 51 Fuzzy membership functions for size

The aim of using Fuzzy Logic is to evaluate the buyer preference for each product using multiple attributes. Based on the degree of membership of each attribute, the buyer preference can be calculated. After comparing the user requirement and product attributes, the final output score ς is calculated using the following defuzzification formula with the matched attributes as follows:

$$\varsigma = \frac{\sum_{i=1}^{K} \mu^2_{A_i\_v}}{M} \in [0,1] \tag{14}$$

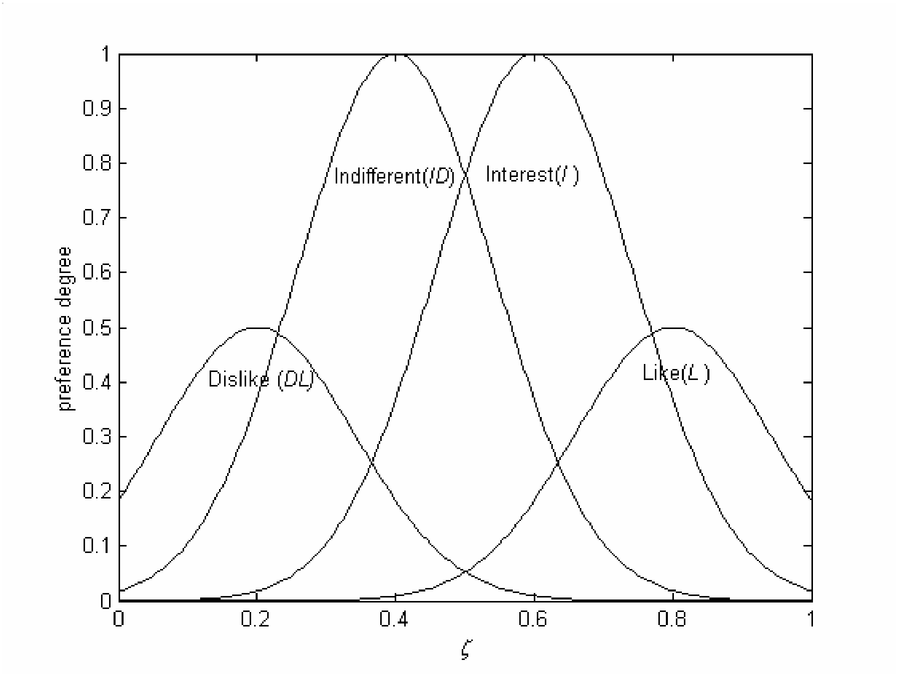


Figure 52 Preference score distribution

The "Preference" (*Pref*) is classified into 4 categories. They are "dislike" (*DL*), "indifferent" (*ID*), "interest" (*I*) and "like" (*L*) by considering the attribute score ς.

$$\begin{aligned} Pref_{dislike}(\varsigma) &= N(z,0.2,\beta)\times 0.5 \\ Pref_{indifferent}(\varsigma) &= N(z,0.4,\beta) \\ Pref_{interest}(\varsigma) &= N(z,0.6,\beta) \\ Pref_{like}(\varsigma) &= N(z,0.8,\beta)\times 0.5 \end{aligned} \tag{15}$$

where β is the standard deviation of the preference category distribution. As shown in Figure 52, we assume that the probability of the extreme cases is lower and the preference degree of categories *DL* and *L* is lower than that of *ID* and *I*.

Given the user requirement and the input data set, i.e., product attributes, the data set can be mapped to the relevant category. We obtain the product preference in each category, and hence the category, that the product belongs to, by selecting the category in which the maximum preference score is achieved.

In this product comparison problem, the shopping agent has to decide whether to buy the item or visit to the next shop by considering the product attributes. The decision points are 1, 2, … *t*, … *N*, where *N* is the total number of shops. The current decision point is denoted as *t* and the next one is denoted as *t*+1. The set of states at each shop is denoted as $S = \{\varsigma,\kappa,\varphi\}\cup\{\Theta\}$ where κ = {*dislike*(*DL*), *indifferent*(*ID*), *interest*(*I*), *like*(*L*)}. The system states are defined by the attribute score of the product (ς), the preference category (κ), the maximum attribute score recorded from the previous shop (φ) and the end state (Θ). If the shopping agent decides to buy at the current shop, it will move to the “end” state.

The set of choices are the same as that in the previous price-comparison models. The transition probability of the *f*MDP model is also similar to that of the price-comparison Model II stated in Chapter 5. The main difference is that *Pref* instead of the price probability is used. It is given by the following expression for *t* = 2 , … *N*-1. As a future work, we may also consider using incremental learning to update the transition probabilities. For example, for updating purposes, we can ask each consumer to fill in a questionnaire to report their level of satisfaction with the product chosen by the agent. The preference score distribution can also be updated using other intelligent computing techniques such as data mining and artificial intelligence .

$$\rho(s_{t+1} \mid s_t, z_t) = \begin{cases} Pref_{\kappa_t}(\varsigma_t) \times Pref_{\kappa_{t+1}}(\varsigma_{t+1}), & \max(\varsigma_t, \varphi_t) = \varphi_{t+1}, z_t = 1 \\ 1, & (s_t \in \{\varsigma, \kappa, \varphi\}, s_{t+1} = \Theta, z_t = 0) or (s_t = s_{t+1} = \Theta, z_t = 1) \\ 0, & \text{otherwise,} \end{cases}$$
$$\rho(s_2 \mid s_1, z_1) = \begin{cases} Pref_{\kappa_t}(\varsigma_1) \times Pref_{\kappa_{t+1}}(\varsigma_2), & \varsigma_1 = \varphi_2, z_1 = 1 \\ 1, & (s_1 \in \{\varsigma, \kappa, \varphi\}, s_2 = \Theta, z_t = 0) or (s_1 = s_2 = \Theta, z_1 = 1) \\ 0, & \text{otherwise,} \end{cases} \quad (16)$$

Denote $O_t(s, z)$ as the overall attribute score for visiting each shop if the system state is $s \in S$ at t and the selected action is $z \in Z(s)$ at t. It is defined by the following:

$$O_1(s, z) = \begin{cases} \varsigma - C_{travel}, & \text{if } s \in \{\varsigma, \kappa, \varphi\}, z = 0 \\ -2 \times C_{travel}, & \text{if } s \in \{\varsigma, \kappa, \varphi\}, z = 1 \\ 0, & \text{if } s = \Theta \end{cases}$$
$$O_t(s, z) = \begin{cases} \varsigma, & \text{if } s \in \{\varsigma, \kappa, \varphi\}, s \geq \varphi, z = 0 \\ \varphi - C_{travel}, & \text{if } s \in \{\varsigma, \kappa, \varphi\}, s < \varphi, z = 0 \\ -C_{travel}, & \text{if } s \in \{\varsigma, \kappa, \varphi\}, z = 1 \\ 0, & \text{if } s = \Theta \end{cases} \quad (17)$$
$$O_N(s, z) = \begin{cases} \varsigma & \text{if } s \in \{\varsigma, \kappa, \varphi\}, s \geq \varphi \\ \varphi - C_{travel} & \text{if } s \in \{\varsigma, \kappa, \varphi\}, s < \varphi \end{cases}$$

where $t = 2, \ldots N-1$.

The overall attribute score is determined by using three equations corresponding to the first decision point, the last decision point and the decision points in between, as explained below. If the decision is “end”, the attribute score of the product found at that shop is obtained. If the decision is “move”, the traveling cost will be included. At the first shop, one more traveling cost C_{travel} is added because the cost of traveling to the first shop should be included. If the shopping agent reaches the last shop, it will stop and the overall attribute score should be equal to the attribute score being offered at the last shop.

6.3. Analytical Results and Discussions

Based on the above model and the backward induction algorithm [31][32], this section presents and discusses analytical results. In the analysis, the following assumptions are made:

- $Pref_{\kappa}(\varsigma)$ is estimated by a Gaussian distribution with standard deviation β and a set of mean values for each preference category.
- The mean of preference category is set as 0.2 for $\kappa = DL$, 0.4 for $\kappa = ID$, 0.6 for $\kappa = I$ and 0.8 for $\kappa = L$. Figure 53 shows the graphic representation of the transition probability when $s_t, s_{t+1} \in \{\varsigma, \kappa, \varphi\}$.

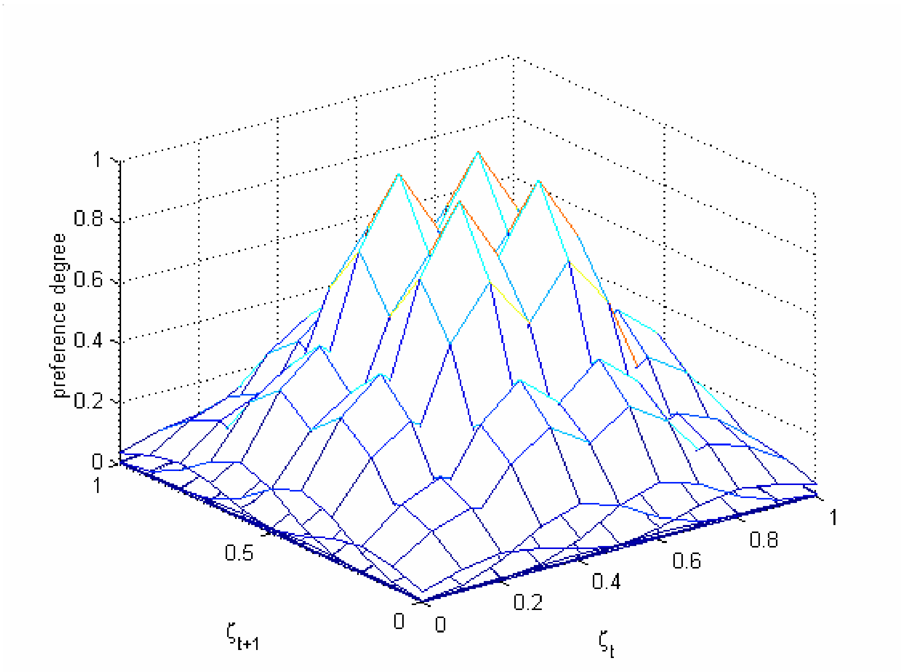


Figure 53 The transition probability

- The attribute score ς is set to be 0.1, 0.3, 0.5, 0.7 and 0.9 for the subsequent analysis. The standard deviation β of the preference distribution and the traveling cost C_{travel} are varied in the following analysis.

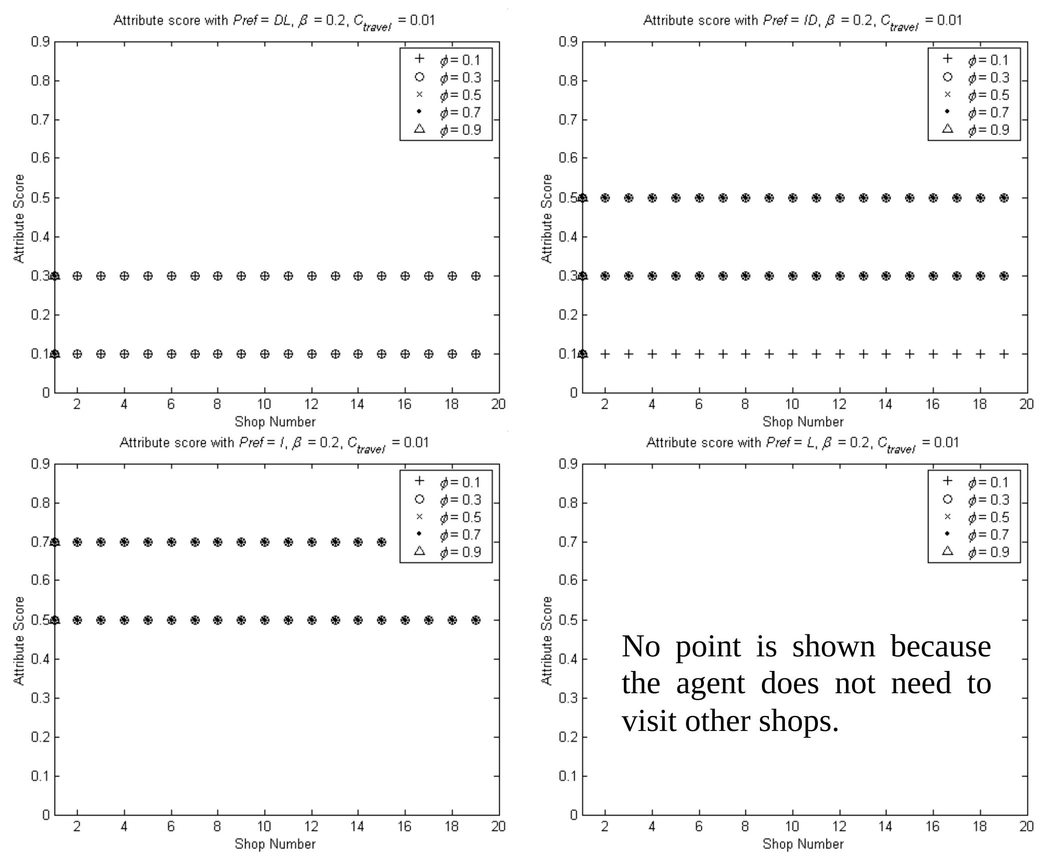


Figure 54 Preferred choices for each preference categories with C_{travel} = 0.01

Figure 54 shows the preferred choices for different preference categories with β = 0.2 and C_{travel} = 0.01. Each symbol in the figure represents a different maximum attribute score being recorded. When there is a symbol in the graph with a particular attribute score, it means that the shopping agent prefers to travel to the next shop. For the scenario with $\kappa = DL$, if the product found in the current shop with the attribute score is smaller than 0.3 and the maximum score recorded is equal to 0.3 or below, the shopping agent will visit to the next shop. When the maximum score recorded is 0.5 or above, the shopping agent will not visit to any other shop (except shop 1). The shopping agent will visit to the next shop when the attribute score calculated in shop 1 is equal to 0.3 or below. As the shopping agent starts from the first shop, there is no attribute score for the previous shop. The attribute score at the current shop becomes the major decision factor. Thus, the preferred choice selected at shop 1 is the same even though different maximum scores are shown in the figure. For the figure with κ=*L*, no point is shown, which means that the shopping agent does not need to visit to other shop if the product being found belongs to $\kappa = L$. Note that each attribute score belongs to each preference category with different degrees of preference. The average score threshold is calculated by using the following equation:

$$\varsigma_{avg,t} = \sum_{s_t \in \{\varsigma, \kappa\}} Pref_{\kappa_t}(\varsigma_t) z(s_t) \tag{18}$$

We have considered four different cases of traveling cost C_{travel}. Figure 55 presents the shopping policy of the shopping agent under different traveling costs when the attribute score is obtained. The shopping agent prefers to visit to the next shop if the attribute score calculated in the current shop is below the threshold line. As mentioned before, the maximum score does not affect the result in the first shop. In other words, the starting point (i.e., the score threshold at the first shop) of each maximum score being recorded is the same. As shown in the top two figures (Figure 55), the threshold score remains constant even when the traveling cost decreases. When the traveling cost is high (see the bottom right figure in Figure 55), the shopping agent prefers not to visit to other shops when the maximum score is 0.7 or higher. Even the shopping agent decides to visit to the next shop, the threshold score and the maximum score are rather low (0.2 and 0.5 respectively).

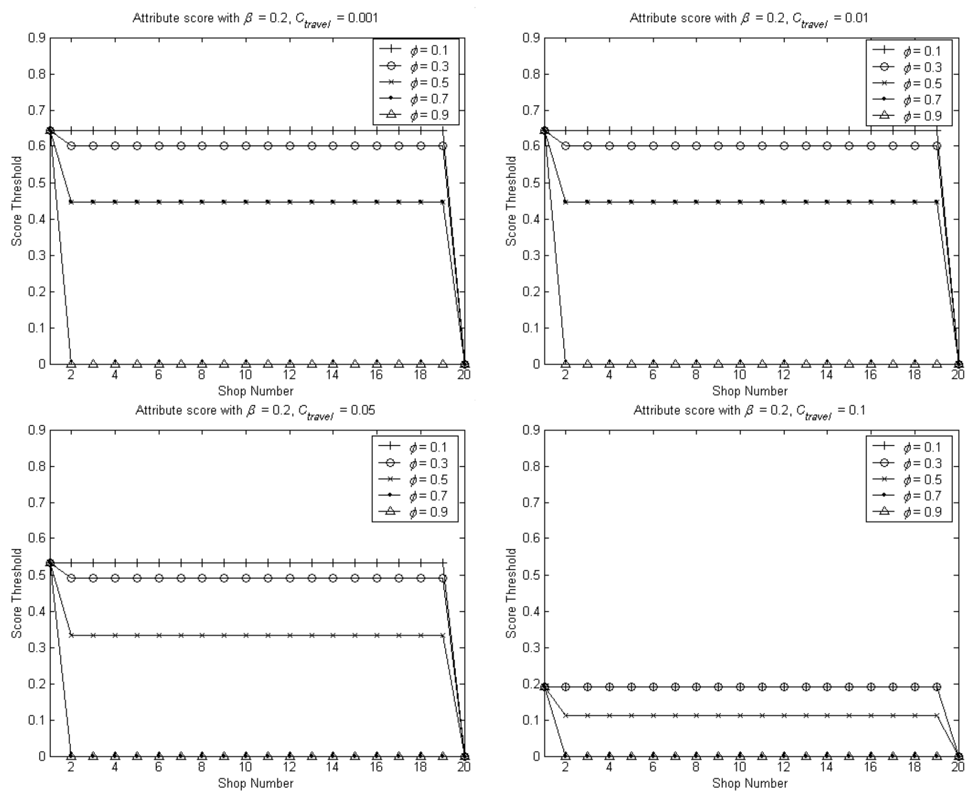


Figure 55 Overall preferred choices with different C_{travel}

Figure 56 depicts the average product score for different numbers of shops. The average product score is calculated as follows:

$$O_{avg} = \sum_{s_1 \in \{\varsigma, \kappa\}} Pref\kappa_1(\varsigma_1)\varpi(s_1) \tag{19}$$

It can be seen that the average score increases as the traveling cost C_{travel} declines. When C_{travel} is low, it is more cost-effective to visit more shops. However, the average cost levels only go up to a certain threshold. This indicates that it is possible to get the highest average score after visiting a certain number of shops. As shown in Figure 56, the variation in the average score is rather small between C_{travel} = 0.01 and C_{travel} = 0.001, the reason behind is that both traveling cost is relatively small with respect to the overall average score in these cases.

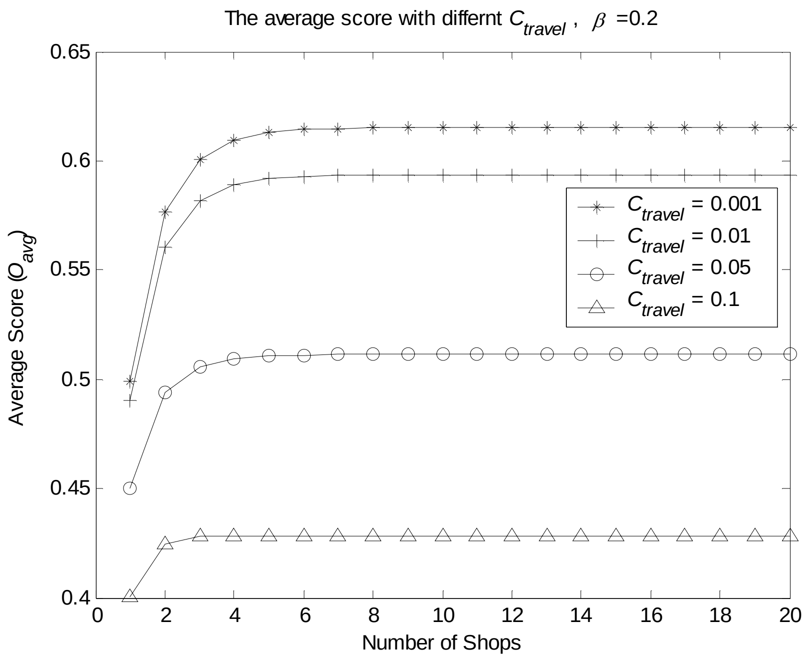


Figure 56 Average product score (O_{avg}) vs. number of shops with different C_{travel}

After showing the effect of varying the traveling cost to the score threshold and the average score, the following figures (Figure 57 and Figure 58) show how the standard deviation (β) of the preference distribution affects the system. Figure 57 shows the overall preferred choices of a shopping agent with C_{travel} = 0.01 when β is varied. As revealed in Figure 57, the shopping agent prefers to visit to the next shop if the attribute score determined in the current shop is below the score threshold. When the standard deviation β of the preference distribution increases, each attribute score belongs to more preference categories. It has no effect on the score threshold when the maximum score is 0.9. Besides, the score threshold increases if the standard deviation decreases and the maximum score is greater than 0.7.

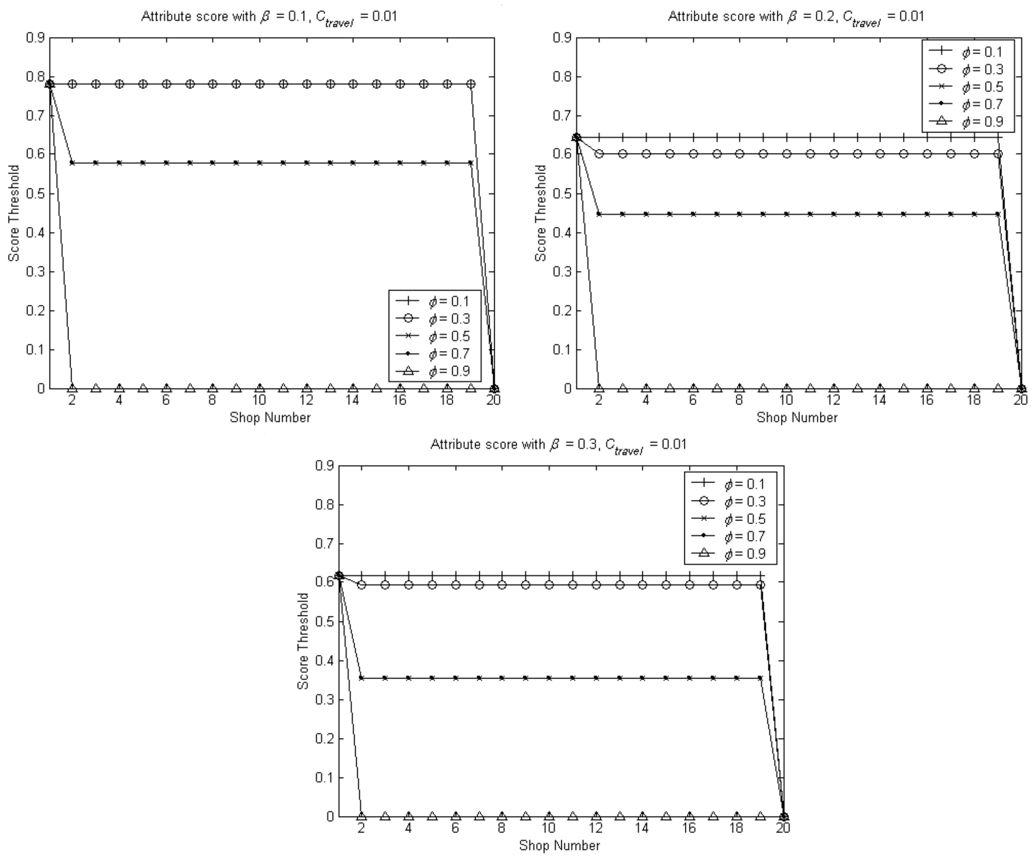


Figure 57 Overall preferred choices with different *β* of the preference distribution

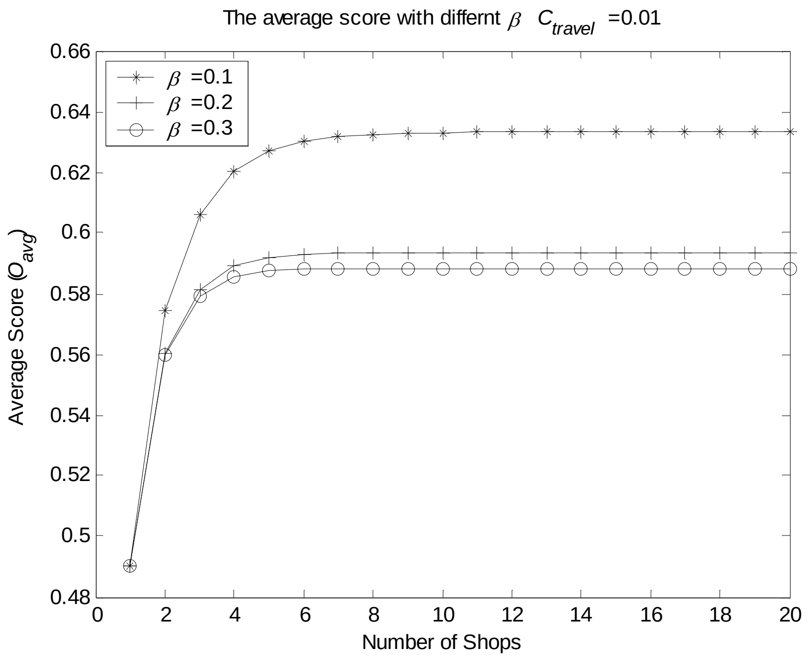


Figure 58 Average product score (O_{avg}) vs. number of shops with different *β* of the preference distribution

Figure 58 shows the average product score by using Eqn. 19. When the standard deviation β rises, the preference degree of the product attribute score will drop. In general, the average score with a high β is lower than that with a low β.

6.4. Comparison with Model II

Besides analyzing the fMDP model with different parameters, we will compare the results of fMDP and Model II given in Chapter 5. As fMDP can handle multiple attributes while Model II can only work on price, the comparison will be performed with the aim of buying the product at the lowest cost.

We obtain the real price information extracted in priceSCAN.com (www.pricescan.com) for conducting the simulation. In Model II, the decision policy for an agent to buy a product is based on the price distribution with the price range between μ-3σ and μ+3σ. The price distribution is estimated by a "discrete" distribution with a mean μ and standard deviation σ based on the price information obtained from priceSCAN.com. In the fMDP model, the price of a product is the only attribute considered in the product comparison, when A_1 = price, V_1 = {low, medium, high} and X_1 = price of a product.

$$price = \{low, medium, high \mid \forall \mu_{\underline{price\_low}}(x), \mu_{\underline{price\_medium}}(x), \mu_{\underline{price\_high}}(x) \in [0,1]\}$$

$$\begin{aligned}
&\mu_{\underline{price\_low}}(x) = \begin{cases} 1 & \text{if } 0 \le x < \mu - 2\sigma \\ N(x, \mu - 2\sigma, \sigma) & \text{if } \mu - 2\sigma \le x \le \mu \\ 0 & \text{otherwise} \end{cases} \\
&\mu_{\underline{price\_medium}}(x) = N(x, \mu, \sigma) \qquad\qquad \text{where } N(x,\alpha,\beta) = e^{\frac{(x-\alpha)^2}{-2\beta^2}} \qquad (20) \\
&\mu_{\underline{price\_high}}(x) = \begin{cases} 1 & \text{if } x > \mu + 2\sigma \\ N(x, \mu + 2\sigma, \sigma) & \text{if } x \le \mu + 2\sigma \end{cases}
\end{aligned}$$

The goal of the shopping agent is to buy the product with the best (i.e., the lowest) cost. The final output score ς of each product can be found by Eqn. 14. After applying ς to Eqn. 15, the category of each product is obtained by finding the maximum preference score.

For experimental purposes, a Matlab program has been written to simulate the buying process. A shopping agent is created to visit the shops and will make a purchase decision according to the policy determined by the models. For system evaluation purposes, 10000 simulations have performed to obtain the experimental results. The shopping agent using the fMDP model makes decision according to the attribute score and the preference category. On the other hand, the agent using Model II makes decision based on the price value. As it is difficult to compare them using two kinds of units, we deliberately set the fMDP model to record the price of the product offered by a shop. The comparison of prices between the two models is shown as follows. To ensure a fair comparison, the travelling cost C_{travel} paid by the shopping agent of the two models is the same. We assume that there are 17 shops for the shopping agents to visit, the shopping agents may visit less than or equal to 17 shops because they might stop travelling according to their decision policies. The average cost is plotted against the number of shops they have visited.

The average costs paid by the shopping agent of the two models are presented in Figure 59 and Figure 60. Overall speaking, the performance of the shopping agent using the fMDP model is better than the one using Model II especially when the traveling cost is high, while the performance is more or less the same when the traveling cost is relatively low. In general, the average cost increases when the number of shops visited increases mainly because more traveling cost is included. In some cases, the average cost drops to zero after visiting a certain number of shops. It is because the shopping agents do not decide to visit more than a specific number of shops according to their decision policies. At a specific number of shops being visited, the traveling cost paid by the two agents must be the same. It reflects that the average product price paid by the agent using the fMDP model is lower than that of Model II. Although the shopping agent using Model II usually stops traveling earlier than the fMDP agent (especially when the traveling cost is high), it does not mean that it is more efficient, as the main purpose of our study is to investigate the overall performance of the two kind of shopping agents given a predefined resources (such as the number of shops to be visited). That means, whether an agent stops traveling earlier is not our main concern. One interesting finding in Figure 60 is that, when the traveling cost is lower than or equal to 0.01μ, even though the shopping agent using Model II will not buy the product until it has visited 15 shops (or more), its overall average cost is still higher than that of the agent using the fMDP model. Generally speaking, fMDP performs better as compared with Model II in most of the cases. Moreover, its main advantage is that it can support multiple attributes comparison, while this can never be achieved by using Model II.

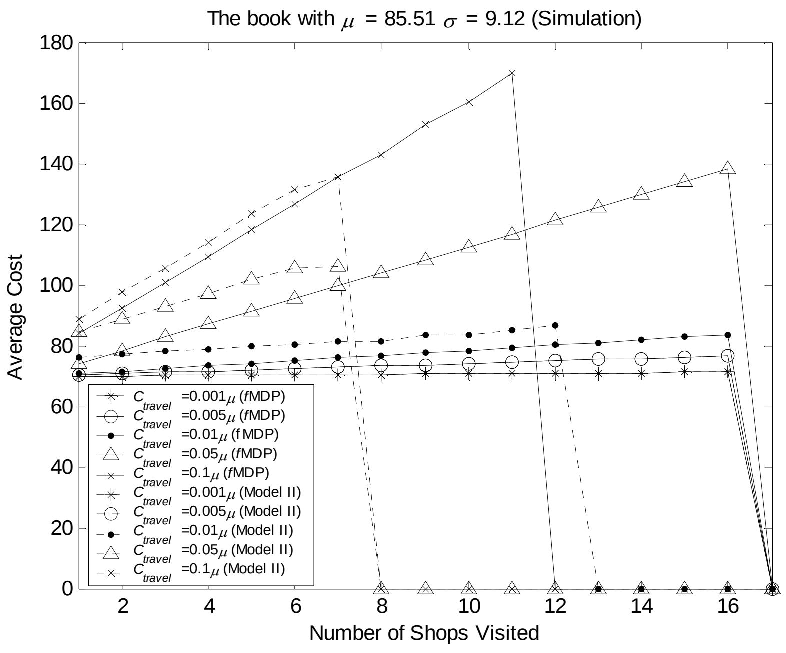


Figure 59 The comparison between fMDP and Model II with high mean (μ) value

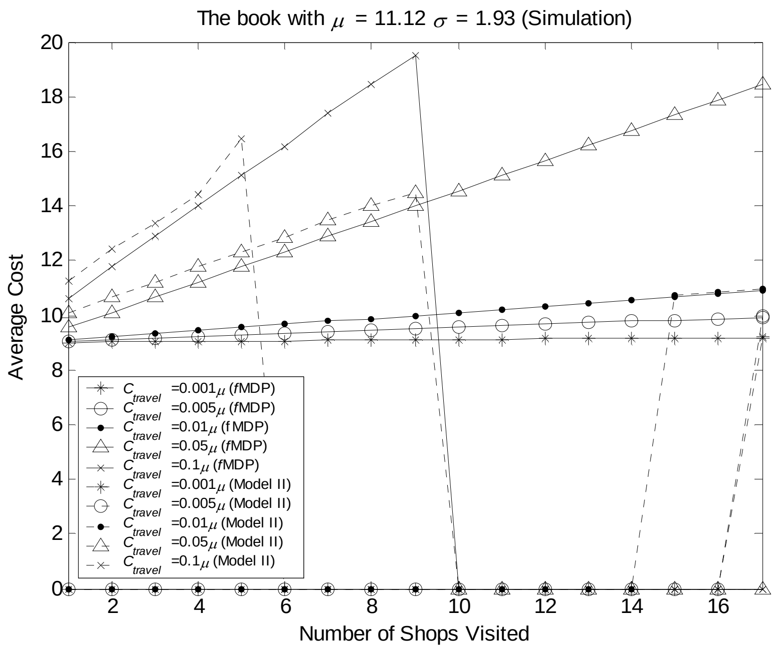


Figure 60 The comparison between fMDP and Model II with low mean (μ) value

6.5.Summary

In this chapter, we have proposed an innovative and intelligent agent-based multi-attribute product comparison model called fMDP. This model integrates Markov Decision Theory and Fuzzy Logic. The fMDP model is formulated to determine the optimal decision policy for a shopping agent to decide whether to buy the product or not based on multiple shopping attributes. The policy provides an effective way for the shopping agent to buy the best-suited product. When the overall preferred choices keep constant except at the first and the last shop, the optimal decision policy is independent on the number of shops being visited but dependent of the preference distribution and the traveling costs. According to the simulations, the fMDP model performs better as compared with Model II proposed in Chapter 5 in most of the cases. This is because unlike the previous model, it can support fuzzy shopping and comparison of multiple attributes. The analytical and experimental results indicate that fMDP is useful for supporting multi-attribute product comparison services for MAGICS.

CHAPTER 7
CONCLUSION

Inspired by existing mobile agent-based e-commerce systems, we have developed an e-commerce system for supporting consumer-oriented e-commerce with mobile agents. The system is called the B2C Mobile AGent-based Internet Commerce System (MAGICS). Besides complementing the current Web-based Internet commerce system by using mobile agents, B2C MAGICS can be used to conduct the consumer buying process on behalf of a user. In this thesis, we focus on investigating the basic architecture and the evaluation stage of the buying process. In Chapter 3, we have presented the basic architecture and the protocols for B2C MAGICS. In particular, a prototype has been built to demonstrate the basic functions.

In Chapter 4, we have proposed a communication framework based on XML for facilitating inter-agent communication. The MAGICS message is defined to include 3 kinds of information, which are basic, functional and additional information. Using the inheritant feature of object-oriented programming, we have also presented the XML schema for defining general product information which is used in the consumer buying process. By doing so, the particular product information can be defined by extending the general product information.

Besides investigating inter-agent communication, we have also studied how a mobile agent can communicate with a shop efficiently during product comparison. In Chapter 5, we have investigated a price-comparison problem for B2C MAGICS for buying standard goods. Two mathematical models based on Markov Decision Theory have been formulated to solve the problem. We have used the backward induction algorithm to determine the optimal decision policy for the shopping agent to make decision in each shop so as to pay the least cost. Based on the models, we have carried out analyses using a normal price distribution and a real price distribution to determine the decision policies. We have found that the normal distribution can provide a good estimation of the real price distribution in general. The results should give valuable insights for supporting price comparison services for B2C MAGICS.

In Chapter 6, we have proposed a Fuzzy Markov Decision Process called *ƒ*MDP buying non-standard goods in B2C MAGICS. It is extended from the price-comparison model discussed in Chapter 5 so as to support multi-attribute product comparison. The *ƒ*MDP model has been formulated to determine the optimal decision policy for a shopping agent to buy a best-suited product based on multiple shopping attributes. We have carried out analyses demonstrating how the system is affected by different parameters. According to the simulations, the *ƒ*MDP model gives a better performance and is more general as compared with Model II proposed in Chapter 5. Moreover, the major advantage is that it can support fuzzy requirements and multiple attributes.

Part of the research results of this thesis have been published as follows:

Perry P. Y. Lam, and Henry C. B. Chan, "A Markov-decision-based price comparison problem for Mobile AGent-based Internet Commerce System (MAGICS)," in *Proceedings of 6th International Conference on E-Commerce Technology* (*CEC 2004*), pp. 34-41, Jul. 2004.

Perry P. Y. Lam, Lei Ye, and Henry C. B. Chan, "Business-to-consumer and business-to-business Mobile AGent-based Internet Commerce System (MAGICS)," in *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics* (*SMC 2004*), Oct. 2004.

REFERENCES

[1] Aglets [Online]. Available: http://www.trl.ibm.com/aglets/

[2] Firefly [Online]. Available: http://www.firefly.com

[3] A. B. Brody, and E. J. Gottsman, “Pocket BargainFinder: a handheld device for augmented commerce,” in *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing* (*HUC’ 99*), pp. 44-51, Sept. 1999.

[4] K. Cagle, D. Gibbons, D. Hunter, N. Ozu, J. Pinnock, and P. Spencer, *Beginning XML*, Wrox Press, Birmingham, England, 2000.

[5] H. C. B. Chan, B. Lam, and R. S. T. Lee, “MAGICS shopper,” in *Proceedings of the International Conference on Internet Computing* (*IC’ 01*), vol. 2, pp. 1062-1068, Jun. 2001.

[6] H. C. B. Chan, C. K. H. Chin, and B. Lam, “Price-comparison agents for MAGICS,” in *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing* (*PACRIM’ 01*), vol. 2, pp. 744-747, Aug. 2001.

[7] H. Chan, H. Chen, T. Dillon, J. Cao, and R. Lee, “A mobile agent-based system for consumer-oriented e-commerce,” in *Proceedings of the 4th International Conference on Electronic Commerce* (*ICEC 2002*), Oct. 2002.

[8] H. C. B. Chan, “Consumer-oriented electronic commerce,” in *Internet Encyclopedia*, vol. 1, John Wiley & Sons, Hoboken, New Jersey, pp. 284-293, 2004.

[9] A. Chavez, and P. Maes, “Kasbah: an agent marketplace for buying and selling goods,” in *Proceedings of the 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* (*PAAM’ 96*), pp. 75-90, Apr. 1996.

[10] M. Chung, and V. Honavar, “A negotiation model in agent-mediated electronic commerce,” in *Proceedings of International Symposium on Multimedia Software E0 ngineering*, pp. 403-410, Dec. 2000.

[11] P. Dasgupta, N. Narasimhan, L. E. Moser, and P. M. Melliar-Smith, “MAgNET: mobile agents for networked electronic trading,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 4, pp. 509-525, Jul.-Aug. 1999.

[12] R. B. Doorenbos, O. Etzioni, and D. S. Weld, “A scalable comparison-shopping agent for the World-Wide Web,” in *Proceedings of the 1st International Conference on Autonomous Agents* (*AGENTS’ 97*), pp. 39-48, Feb. 1997.

[13] A. E. Fano, “Shopper's eye: using location-based filtering for a shopping agent in the physical world,” in *Proceedings of the 2nd International Conference on Autonomous Agents* (*AGENTS’ 98*), pp. 416-421, May 1998.

[14] S. Fong, A. Sun, and K. K. Wong, “Price watcher agent for e-commerce,” in *Proceedings of the 2nd Asia-Pacific Conference on Intelligent Agent Technology* (*IAT 2001*), pp. 294-299, Oct. 2001.

[15] S. Fong, S. Siu, and A. Sun, “WebSpy: retrieving web contents for e-business intelligence,” in *Proceedings of the 1st International Conference on Information Technology and Applications* (*ICITA 2002*),

Nov. 2002.

[16] R. J. Glushko, J. M. Tenenbaum, and B. Meltzer, “An XML framework for agent-based ecommerce,” *Communications of the ACM*, vol. 42, no. 3, pp. 106-114, Mar. 1999.

[17] D. Grimshaw, Artificial Intelligence Topics with Agents, http://www.ryerson.ca/~dgrimsha/courses/cps720/

[18] M. He, N. R. Jennings, and H. F. Leung, “On agent-mediated electronic commerce,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 985-1003, Jul.-Aug. 2003.

[19] C. M. Jonker, and J. Treur, “An agent architecture for multi-attribute negotiation,” in *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (*IJCAI’01*), pp.1195-1201, Aug. 2001.

[20] R. Kalakota, and A. B. Whinston, *Electronic Commerce: A Manager’s Guide*, Addison-Wesley, Massachusetts, 1997.

[21] R. Kerin, E. Berkowitz, S. Hartley, and W. Rudelius, *Marketing*, 7th edition, McGraw-Hill/Irwin Companies, New York, 2003.

[22] B. Krulwich, “The BargainFinder agent: comparison price shopping on the internet,” in *Agents, Bots, and Other Internet Beasties*, Macmillan Computer Publishing, Indianapolis, pp. 257-263, May 1996.

[23] K. Kuramitsu, and K. Sakamura, “Distributed object-oriented schema for XML-based electronic catalog sharing semantics among businesses,” in *Proceedings of the 1st International Conference on Web Information Systems Engineering* (*WISE’00*), vol. 1, pp. 87-96, Jun. 2000.

[24] Y. Labrou, and T. Finin, “A semantics approach for KQML a general purpose communication language for software agents,” in *Proceedings of*

the 3rd International Conference on Information and Knowledge Management (*CIKM' 94*), pp. 447-455, Nov. 1994.

[25] Y. Labrou, T. Finin, and Y. Peng, “Agent communication languages: the current landscape,” *IEEE Intelligent Systems*, vol. 14, no. 2, pp. 45-52, Mar.-Apr. 1999.

[26] D. B. Lange, “Mobile objects and mobile agents: the future of distributed computing?” in *Proceedings of the 12th European Conference on Object-Oriented Programming* (*ECOOP’ 98*), Jul. 1998.

[27] D. B. Lange, and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley, Massachusetts, 1998

[28] R. S. T. Lee, *Fuzzy-Neuro Approach to Agent Applications*, Springer-Verlag, Heidelberg, New York, 2004

[29] P. Maes, “Software agents and the future of electronic commerce,” MIT Media Laboratory Group. 1998. http://pattie.www.media.mit.edu/people/pattie/ECOM/

[30] A. Moukas, G. Zacharia, R. Guttman, and P. Maes, “Agent-mediated electronic commerce: an MIT media laboratory perspective,” *The International Journal of Electronic Commerce*, vol. 4, no. 3, pp. 5-21, 2000.

[31] M. L. Puterman, “Dynamic programming,” in *Encyclopedia of Physical Science and Technology*, vol. 4, Academic Press, pp. 438-463, 1987.

[32] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st edition, John Wiley and Sons, New York, 1994.

[33] T. Sandholm, and Q. Huai, “Nomad: mobile agent system for an

Internet-based auction house," *IEEE Internet Computing*, vol. 4, no. 2, pp. 80-86, Mar.-Apr. 2000.

[34] M. H. Sherif, *Protocols for Secure Electronic Commerce*, 2nd edition, CRC Press, Boca Raton, Florida, 2003.

[35] J. N. Sheth, B. Mittal, and B. Newman, *Consumer Behavior: Consumer Behavior and Beyond*, Dryden Press, 1998.

[36] K. M. Sim, and R. Chan, "A brokering protocol for agent-based e-commerce," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 30, no. 4, pp. 474-484, Nov. 2000.

[37] B. Sommers, Agents: Not just for Bond anymore, http://www.javaworld.com/javaworld/jw-04-1997/jw-04-agents.html

[38] M. Ströbel, "Communication design for electronic negotiations on the basis of XML schema," in *Proceedings of the 10th International Conference on World Wide Web* (*WWW' 01*) , pp. 9-20, May 2001.

[39] E. Turban, D. King, J. Lee, M. Warkentin, and H. M. Chung, *Electronic Commerce 2002 : A Managerial Perspective*, 2nd edition, Prentice Hall, Upper Saddle River, New Jersey, 2002.

[40] J. Ueyama, and E. R. M. Madeira, "An automated negotiation model for electronic commerce," in *Proceedings of the 5th International Symposium on Autonomous Decentralized Systems* (*ISADS' 01*), pp. 29-36. Mar. 2001.

[41] Y. Wan, S. Menon, and A. Ramaprasad, "A classification of product comparison agents," in *Proceedings of the 5th International Conference on Electronic Commerce* (*ICEC 2003*), pp. 498-504, Sep. 2003.

[42] M. J. Wooldridge, and N. R. Jennings, "Software engineering with agents:

pitfalls and pratfalls," *IEEE Internet Computing*, vol. 3, no. 3, pp. 20-27, May-Jun. 1999.

[43] J. Yan, M. Ryan, and J. Power, *Using fuzzy logic: towards intelligent systems*, Prentice-Hall, New York, 1994.

[44] L. A. Zadeh, "Fuzzy sets", in *Information and Control*, vol. 8, pp. 338-353, 1965.

[45] Y. Q. Zhang, "Fuzzy logic," in *Internet Encyclopedia*, vol. 1, John Wiley & Sons, Hoboken, New Jersey, pp. 841-851, 2004.

[46] F. Zhu, and S.-U. Guan, "Towards evolution of software agents in electronic commerce," in *Proceedings of the Congress on Evolutionary Computation* (*CEC 2001*), vol. 2, pp. 1303-1308, May 2001.