

# **An Interest-based Offer Evaluation System for Semantic Matchmaking**

A Thesis

Submitted to the Faculty of Graduate Studies and Research

In Partial Fulfillment of the Requirements

for the Degree of  
Master of Science  
in Computer Science  
University of Regina

by

Wei Jiang  
Regina, Saskatchewan

June, 2011

Copyright © 2011: Wei Jiang



Library and Archives  
Canada

Published Heritage  
Branch

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque et  
Archives Canada

Direction du  
Patrimoine de l'édition

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file Votre référence*

*ISBN: 978-0-494-88529-1*

*Our file Notre référence*

*ISBN: 978-0-494-88529-1*

#### NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

#### AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

**UNIVERSITY OF REGINA**  
**FACULTY OF GRADUATE STUDIES AND RESEARCH**  
**SUPERVISORY AND EXAMINING COMMITTEE**

Wei Jiang, candidate for the degree of Master of Science in Computer Science, has presented a thesis titled, ***An Interest-based Offer Evaluation System for Semantic Matchmaking***, in an oral examination held on June 9, 2011. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner:           Dr. Yasser Morgan,  
  Faculty of Engineering and Applied Science

Supervisor:                     Dr. Samira Sadaoui, Department of Computer Science

Committee Member:         Dr. Boting Yang, Department of Computer Science

Committee Member:         Dr. Malek Mouhoub, Department of Computer Science

Chair of Defense:             Dr. Amr Henni, Faculty of Engineering & Applied Science

\*Not present at defense

# Abstract

Matchmaking is the online process through which buyers and suppliers exchange products and services. With the blooming of E-commerce, matchmaking systems are used for ease, timesaving and personalized options to help buyers find their most interested offers (products or web services).

Right now, semantic matchmaking systems are not able to recognize the differences between buyers' interest and tastes. Existing matchmaking and ranking methods are still not good enough to provide the best offer according to the individual's interests and needs. To address this weakness, we develop an offer evaluation and ranking system for semantic matchmaking. Our system determines the best offer by evaluating and sorting the request-matched offers according to the buyer's interests and preferences. The best or more interested offer represents the maximum satisfaction of the buyer. Our system extracts and analyzes the buyer's interests for multiple offer attributes to bring better results to each individual.

To evaluate and rank the candidate offers, we adapt the economical model called MultiNomial Logit (MNL) to the field of semantic matchmaking. MNL model suggests that each buyer has a taste of the population and an individual taste to make a choice. Nevertheless, in practice, the MNL model focuses on the population taste and the individual taste is eliminated. In this research, we assume the individual taste is a special reflection of some offer attribute data.

The individual taste brings some additional interests to the population taste. Following this assumption, we define a new interest model that takes into account the individual's interests and favors on multiple offer attributes.

We demonstrate the feasibility and benefits of our offer evaluation system through a detailed case study involving high dimensional offer attributes. In this study, we show how our system catches the differences between the interests of two buyers, and how it recommends a different best offer to each buyer.

# Acknowledgements

This thesis would not have been possible without my supervisor Professor Samira Sadaoui who brought her enthusiasm and inspiration to support my research work, lead me to success, and encouraged me throughout this endeavor. During my bad times, Professor Samira Sadaoui accepted me to work on a very interesting research area, and contributed her time, ideas, knowledge, and funding to make research enjoyable for me. I appreciate Professor Samira Sadaoui's help and efforts.

I would like to thank the members of my thesis committee: Dr. Malek Mouhoub, and Dr. Boting Yang for their time, helpful discussions, and inspirational comments. Also, I thank the Computer Science department for providing me with such a good environment in which to learn and grow. A special thanks to Marilyn Hepp who was always ready to help.

Lastly, I would like to specially thank my family for all their love and encouragement. For my parents who raised me with an endless love and support. For my fiance Yingjie Sun, for faithful love, support, and encouragement. And also thanks to her family. Thank you!

Wei Jiang

*University of Regina*

June 2011

# Post Defense Acknowledgements

I would like to extend a sincere thanks to my external examiner, Dr. Yasser Morgan, for his comments and suggestions to improve my thesis better. I also would like to show my gratitude to the committee chair, Dr. Amr Henni, for organizing the defense.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Post Defense Acknowledgements</b>	<b>iv</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>x</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Problem and Motivation . . . . .	1
1.2 Research Statement . . . . .	4
1.3 Thesis Organization . . . . .	7
<b>2 BACKGROUND</b>	<b>8</b>
2.1 Matchmaking System Overview . . . . .	8
2.2 MNL Model . . . . .	11
2.3 Clustering Technology SOM . . . . .	14
2.3.1 Competitive Learning . . . . .	14
2.3.2 Self-Organizing Map Algorithm . . . . .	15
<b>3 INTEREST MODEL BASED OFFER EVALUATION SYSTEM</b>	<b>19</b>
3.1 Interest Model Construction . . . . .	19



3.2	Interest Weight and Interest Rate Function Discussion . . . . .	21
3.3	Attribute Data Extraction . . . . .	24
3.4	Attribute Data Clustering . . . . .	26
3.4.1	Clustering Algorithm . . . . .	26
3.4.2	High Dimensional Data Discussion . . . . .	30
3.5	Interest Weight Calculation . . . . .	31
3.6	Interest Rate Function Generation . . . . .	33
3.7	Offers Evaluation with the Interest Model . . . . .	39
<b>4</b>	<b>DESIGN AND IMPLEMENTATION</b>	<b>41</b>
4.1	High-level Design . . . . .	41
4.2	Extracting Attribute Data . . . . .	42
4.3	Main Component Design . . . . .	46
4.3.1	Attribute Data Clustering . . . . .	46
4.3.2	Interest Weight Calculator . . . . .	49
4.3.3	Interest Rate Function Generator . . . . .	50
4.4	Implementation . . . . .	52
4.4.1	Implementation Environment . . . . .	52
4.4.2	Implementation and GUI . . . . .	53
<b>5</b>	<b>EXPERIMENTATION</b>	<b>56</b>
5.1	Case Study: Computers Purchasing Based on Multiple Attributes	56
5.1.1	Extracting Attribute Data . . . . .	57
5.1.2	Clustering Attribute Data . . . . .	57

5.1.3	Selecting Attribute Clustering . . . . .	59
5.1.4	Calculating Interest Weights . . . . .	61
5.1.5	Generating Interest Rate Functions . . . . .	62
5.1.6	Generating the Interest Models . . . . .	72
<b>6</b>	<b>CONCLUSION AND FUTURE WORK</b>	<b>77</b>
6.1	Thesis Contributions . . . . .	77
6.2	Future Work . . . . .	78
	<b>Bibliography</b>	<b>80</b>
	<b>Appendix A: SOM Implementation</b>	<b>85</b>

# List of Figures

1.1	Interest-based Evaluation System . . . . .	5
1.2	Offer Evaluation Process . . . . .	6
2.1	LVQ Learning Awards in Competitive Learning . . . . .	15
2.2	SOM Model Overview . . . . .	16
2.3	SOM Learning . . . . .	17
3.1	Formatted Matchmaking Request . . . . .	25
3.2	Hard Drive Clustering Tree . . . . .	30
3.3	CPU Clustering Tree . . . . .	31
3.4	Sigmoid Function and Interest Interval . . . . .	34
3.5	Binding Interest Rate Function . . . . .	37
4.1	An Interest-based Offer Evaluation System . . . . .	43
4.2	Example of Semantic Matchmaking Responding Message . . . . .	44
4.3	Relationships between Tables in Server Databases . . . . .	45
4.4	Offer Evaluation Sequence Diagram . . . . .	47
4.5	<i>AttributeClustering</i> Component Processing . . . . .	48
4.6	<i>InterestWeightsCalculator</i> Component Processing . . . . .	50
4.7	Interest Weights Calculation for Two Companies . . . . .	51
4.8	<i>InterestRateFunctionCreator</i> Component Processing . . . . .	52
4.9	Server Side Class Diagram . . . . .	53
4.10	Client Side Class Diagram . . . . .	54

4.11 GUI for Clustering Selection . . . . .	55
5.1 Hard Drive Clustering Tree . . . . .	59
5.2 CPU Clustering Tree . . . . .	59
5.3 Price Data Clustering Tree . . . . .	60
5.4 RAM Data Clustering Tree . . . . .	60
5.5 Company1 Interest Rate Function for Hard Drive Data . . . . .	64
5.6 Company2 Interest Rate Function for Hard Drive Data . . . . .	65
5.7 Company1 Interest Rate Function for CPU Data . . . . .	66
5.8 Company2 Interest Rate Function for CPU Data . . . . .	67
5.9 Company1 Interest Rate Function for Price Data . . . . .	69
5.10 Company2 Interest Rate Function for Price Data . . . . .	71
5.11 Company1 Interest Rate Function for RAM Data . . . . .	72
5.12 Company2 Interest Rate Function for RAM Data . . . . .	73

# List of Tables

3.1	Stock Data Example . . . . .	22
3.2	Extracting Attribute Data from the Responding Message . . . . .	26
3.3	Interest Rate Functions Shape . . . . .	36
3.4	Distances of High-Dimensional Data . . . . .	38
5.1	Example of Attribute Data . . . . .	58
5.2	Sections for both Companies . . . . .	61
5.3	$\varsigma_{HardDrive}()$ Generation for Company1 . . . . .	63
5.4	$\varsigma_{HardDrive}()$ Generation for Company2 . . . . .	64
5.5	$\varsigma_{CPU}()$ Generation for Company2 . . . . .	66
5.6	Distances of Price Data for Company1 . . . . .	68
5.7	Distances of Price Data for Company2 . . . . .	69
5.8	$\varsigma_{RAM}()$ Generation for Company1 . . . . .	71
5.9	$\varsigma_{RAM}()$ Generation for Company2 . . . . .	71
5.10	Sorted Offers for Company1 . . . . .	75
5.11	Sorted Offers for Company2 . . . . .	76

# Chapter 1

## INTRODUCTION

### 1.1 Problem and Motivation

E-commerce provides modern day societies with products and services in an electronic way. Buyers enjoy its obvious advantages, for example, convenience, lower-cost and efficiency. However, the amount of offers for products and web services are huge, and buyers are not willing to spend too much time browsing each offer. Nowadays, browsing can be hardly considered as a satisfied method to begin an e-commercial activity. Buyers prefer to interact and co-operate with intelligent systems that can help them to make a good deal.

Buyers need to employ good methods to assist themselves instead of browsing the Internet resources (offers of products or web services). Matchmaking systems have been introduced to address this issue. Matchmaking is an online process through which buyers and suppliers exchange goods or services. The earliest research work is by Genesereth in 1992 [1]. In this research, a matchmaking

system is designed to fulfill the information querying, matching, and exchange in the E-commerce. Most matchmaking systems are further developed based on this idea. Since then, evaluation methods are implemented in order to provide buyers with the best matched offer. For instance, Ha [2] applies attribute values evaluation method in his matchmaking process.

With the support of semantic web, matchmaking systems can play an important role in the global e-commerce area. Products or services, distributed on the Internet, can be accessed by any semantic matchmaking systems. Research on ontology helped the semantic matchmaking systems to understand and process the purchasing requests much better [3–5]. Kawamura [3] developed a plug-in to match offers registered in the Universal Description, Discovery, and Integration (UDDI) platform. Qiu [4] uses a similarity table to understand the similarity semantic concepts. Dong-Wei [5] converts semantic concepts relationship to logics clauses. In the industry, some famous companies, like IBM, are continuing to develop semantic matchmakers, such as Ariba, I2, to help enterprisers for a better commerce.

Nevertheless, demands for a better matchmaking system are still increasing in different e-commercial marketing, like Business-to-Business (B2B) [6], Business-to-Government (B2G) and Business-to-Consumer (B2C) [7]. Semantic matchmaking systems can help the buyers to discovery the query-matched offers, but are not good enough to find the best offer. With the booming of e-commerce and e-services, buyers can obtain more and more query-matched candidate offers. The problems of booming information become a challenge for the current

matchmaking research. One of the big issues is that it becomes time consuming for buyers to evaluate all the candidate offers in order to find the best offer.

Today determining the best offer is more important than ever before. Thus, semantic matchmaking systems are trying to determine the best offer by using semantic ranking methods [8–12]. Semantic ranking algorithms sort offers by similarity rate of concepts [8], models [10], attributes [11], parameters [12], or by logic relationships [9]. They sort offers by the semantic similarity rates of the requests and offers.

Yet all existing matchmaking systems fail to bring the best offers to individuals. Indeed, although semantic matchmaking systems apply the semantic ranking methods, they have no guarantee that the system-evaluated best offer will be purchased by the buyer. Since semantic matchmakers do not focus on analyzing buyer's interests, they can not make sure their generated best offer would satisfy buyer's needs and tastes, and sometimes, matchmaking ends in a failure. Without studying buyer's interests and by only relaying on the semantic ranking methods, semantic matchmaking systems cannot recognize the differences between buyers' favors and tastes.

Researchers realize that a better matchmaking system “could quicken the trend toward personalization” [13]. Matchmaking systems based on semantic supports can help the buyer find the query-matched offers but are not good enough to find the best offer. Human beings are looking forward for a better matchmaking and ranking method.



## 1.2 Research Statement

Semantic matchmaking systems are required in E-commerce but they failed to determine the best offer according to buyer's interests and tastes. This thesis is expected to solve the unmatched-interest problems for semantic matchmaking. The goal of our work [14] is to help the buyer find the best and most interesting offer. Although some e-commerce cases are more complex than the ones used in this research, we focus on promoting a new approach to determine the best offer.

The MultiNomial Logit model (MNL) [15,16] is widely used in commerce and statistic areas to study human choice behaviors [17]. Nevertheless, MNL can only suggest a common interest for a group of people. In addition, MNL needs an appropriate sample data to generate its model function. These limitations restrain MNL model to analyze individual interests. First, an individual interest is usually different from the common interest. Second, it takes time to collect the sample data for one individual, and the results are often with a large deviation error.

We modify the MNL model to provide a new interest model for each individual based on the following assumption. We assume the individual taste is a special reflection of some offer attributes data. The individual taste brings some additional interests to the population taste for these attribute data. In order to represent individual interest changes, we define our interest model as a non-linear function. Indeed, we employ a sigmoid based function, and its shape is determined by the individual interest focus. The result of our interest model is an

interest rate of an offer, which represents the sum of individual interests for each offer attribute with their weights. This model evaluates the offers and returns the best offer w.r.t. individual's tastes. The best offer has the highest interest value to the individual.

We develop an offer evaluation and ranking system for semantic matchmakers based on our interest model (cf. Figure 1.1). The evaluation system returns the best offer by sorting the query-matched offers according to the buyer's interests and tastes. The best offer represents the maximum interests for the buyer.

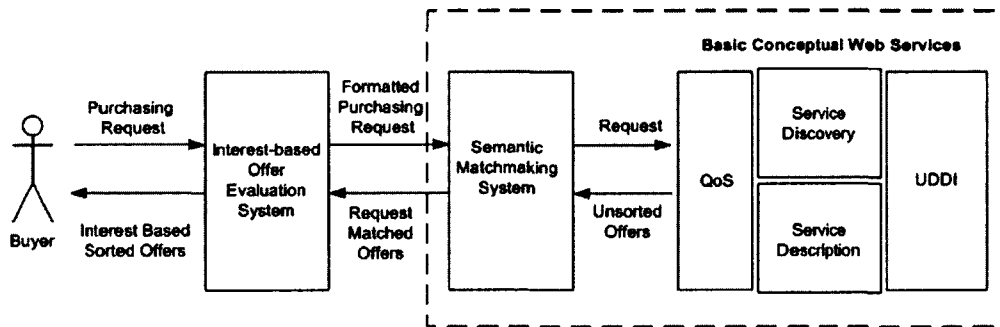


Figure 1.1: Interest-based Evaluation System

The evaluation system has been implemented in two parts: system server side and client side. In the evaluation process (cf. Figure 1.2), first the buyer submits a purchasing request which is then sent to the connected semantic matchmaker. The latter returns a list of query-matched candidate offers. To sort these offers according to the buyer's interests, our evaluation system produces the clustering tree for each offer attribute; builds the buyer's interest model based on the interest weights and interest rates of the offer attributes; applies the resulting model

to evaluate and sort the offers. We generate the attribute weights and rates according to the buyer's selected data clustering (interests). Different clustering selections will produce different interest models.

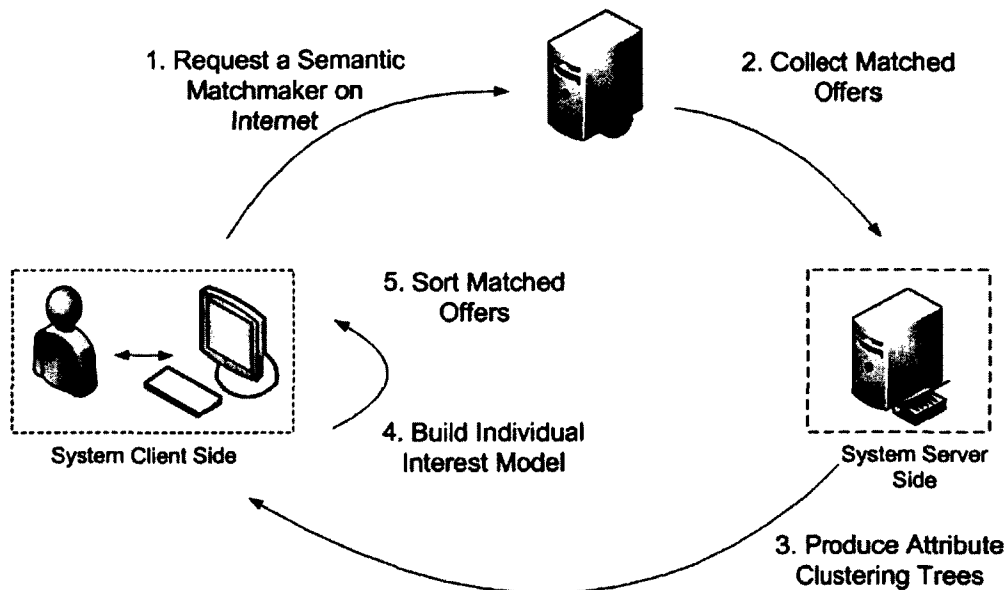


Figure 1.2: Offer Evaluation Process

Furthermore, to analyze and take into account the buyer's interest for each attribute, our evaluation system incorporates a data clustering technology called Self Organizing Map (SOM) [18, 19]. Using data clustering, our system is able to map the buyer's individual tastes to some offer attribute values, that is, the interest focus. Analyzing individual interests along with the semantic matching brings better results to each buyer. As a neural-network based approach, SOM is employed to cluster high-dimensional inputs onto lower-dimensional outputs. The reason of using SOM in our work is that the offer attributes may be complex

or contain high-dimensional data.

To better illustrate the benefits of our evaluation system, we assume the following case: there are several buyers looking for the same offer. In all existing matchmaking systems, buyers who submit the same query get the same (ranked) list of offers. In Chapter 5, through a 4-attribute application, we demonstrate how our system recommends different best offer for each buyer.

### **1.3 Thesis Organization**

The rest of the thesis is structured as follows. In Chapter 2, first we review semantic matchmaker systems and semantic ranking algorithms. Second, we explain the economic human-shopping-behavior model MNL. Finally, we present the clustering technology SOM, and its competitive learning process. In Chapter 3, we expose the phases of the offer evaluation process. The evaluation starts from collecting the offer information. After converting the information to useful data format, individual interests are analyzed during the interaction with the buyers. Based on the analysis of buyer's interest features, the best offer can be calculated. Furthermore, we will discuss how to use some technologies to simulate the individual interest model. In Chapter 4, we present the system design and implementation. In Chapter 5, we demonstrate the benefits of our system through a detailed case study which consists of purchasing computers based on several high-dimensional attributes. In Chapter 6, we report the contribution of our research and the future work.

# Chapter 2

## BACKGROUND

In this chapter, we present some well-known semantic matchmaking and ranking methods. Most current ranking methods do not take into account individual's interests. In order to develop an interest-based evaluation method, we discuss the MNL model and the clustering technology SOM.

### 2.1 Matchmaking System Overview

In the early time of e-commerce, matchmaking systems mapped buyers with suppliers using methods that rank the rates of offer attribute value fit with the requests. For example in Ha [2], the matchmaking system consists of mapping the offer attributes with the request attributes.

In the World Wide Web, requests and offers can be distributed in different places and expressed in different schemas or words even when representing the same semantic meaning. This causes the early matchmaking systems to be blind

to some potential offers for lacking the semantic supports.

To solve this problem, several semantic matchmaking models, based on ontological technologies, have been proposed [3–5]. Kawamura [3] employs the semantic matching in the Universal Description, Discovery, and Integration (UDDI) framework, a platform-independent registry of web services. In UDDI, offers are classified and stored by category register model. The matchmaking system matches all the registered services with the request using semantic filtering. Qiu [4] applies an ontology similarity table to identify all similar concepts in offers as queried. Such similarity table records all the similar concepts. Thus once a concept is in the request, all the offers with the similar-meaning concept can be measured. Dong-Wei [5] uses logical relationships to map the concepts of request and offers. Thus, an offer whether is matched to requests or not can be converted to a logical problem which can be solved easily.

However, these matchmaking systems return an unranked list of candidate offers, and no best offer is suggested by the system. The buyer needs to spend a lot of time to identify the best offer. Several semantic ranking algorithms have been proposed to evaluate the offers and determine the best offer. Since semantic offers are complex, these algorithms involve ranking different attribute types and different criteria. Ranking offers include the similarity rate of concepts [8], attributes [11, 20], models [10], parameters [12], or by logic relationship [9]. Dong-Wei [8] focuses on the concepts distance between offers and query. Huang [11] believes a best offer is an offer which contains the buyer's most wanted attribute value. Shen [20] examines the concepts and values together for ranking

the offers. Hahn [10] abstracts each offer and request into a single model. Bellur [12] analyzes the content of query and offer, and also what the buyer really wants and what the supplier can provide. Wang [9] studies the order of concepts relationships.

Different ranking criteria have been defined for these different ranking types. Dong-Wei [8] introduces a distance measurement method to get the concepts distance between the request concept to the offer concept in the ontology tree. Huang [11] employs buyer-defining attribute weights to calculate the offers rates. The highest rate offer is the best offer. Shen [20] analyzes each key word and attribute data appearing in the query and offers, and then calculates the offer matching rate with their own functions. Ranking the offers is based on the order of the matching rate. In Hahn [10], ranking is based on the semantic similarity values of the request model and offer models. Bellur [12] calculates the matching rates of parameters: query and offer, and also the buyer's needs and the supplier's products. Wang [9] lists the logical relationships with different matching levels and identify each offer with a related level.

These above ranking criteria methods map the functional properties of offers with the request's functional description. However, they cannot explain why sometimes a buyer prefers an offer different from the system evaluated best offer. To address this issue, some non-functional matching methods Yu [21] are introduced like the matching standard Quality of Service (QoS). Yu [21] argues that the buyer's choice is caused by other criteria often referred to as non-functional properties. Liu [22] utilizes an extended QoS model to rank the offers. The

extended QoS values are defined by both buyers and suppliers. Wang [23] applies the QoS metrics to produce the closest offers to the request. Most of the non-functional properties methods are linear functions which may not provide a correct solution for the best offer. In addition, most QoS can only explain the buyer's choice in general for a population. Indeed, the experiments in these research papers are based on a purchasing scenario of one user. This approach uses the population interest to represent an individual interest and dismiss the individual's specific favors and tastes.

Nevertheless, all these ranking algorithms are based on criteria of matching offer and query excluding the buyer's own criteria of interest. Today, there is no ranking algorithm that takes into account human interests and tastes on multiple attributes. The reason is that these algorithms are only based on ranking the values of query words, functional or non-functional parameters, but not on ranking offers by individual interests. We believe the buyer interest criterion is the real reason explaining why an individual selects a specific offer as the best offer. Our goal is to build the buyer's interest model and find the best offer which is the closest to the buyer's real interests, so that it can make up the weakness of current semantic matchmaking systems.

## 2.2 MNL Model

MultiNomial Logit model (MNL) is introduced by McFadden [15, 16], and it expresses the utility of a group of people choosing an item. For example, authors



in [17] use MNL model to describe the people selection of different travel route according to the different time period; Yang [24] applies the MNL model to shopping area selections. The MNL utility function Formula (2.1) for an individual in a population includes the deterministic and random components as follows [15]:

$$U_{ni} = V_{ni} + \varepsilon_{ni} \quad (2.1)$$

where  $U_{ni}$  represents the utility for buyer  $n$  selecting item  $i$ ;  $V_{ni}$  represents the taste of the population; and  $\varepsilon_{ni}$  the individual taste for the item  $i$ . Below in Formula (2.2) [15], the component  $V_{ni}$  consists of  $K$  observed deterministic features.  $V_{ni}$  contains the weight  $b_k$  for each deterministic feature  $x_{nik}$  and  $C$  denotes the set of choice items.

$$U_{ni} = \sum_{k=1}^K b_k \cdot x_{nik} + \varepsilon_{ni}, \quad i \in C \quad (2.2)$$

In order to get the feature weight  $b_k$ , MNL model uses choice probability to avoid including the random utility  $\varepsilon_{ni}$ . In the MNL model assumption,  $\varepsilon_{ni}$  is an error the buyer made in measurement, and it can be described as a random element (continuous random variables) [16] with a double exponential distributed probability. After simplifying the function, the probability for person  $n$  selecting item  $i$  is given in Formula (2.3) [16]:

$$P_{ni} = \frac{e^{V_{ni}}}{\sum_{j \in C} e^{V_{jn}}} = \frac{e^{b_k \cdot x_{nik}}}{\sum_{j \in C} \sum_{k=1}^K e^{b_k \cdot x_{nj k}}}, \quad i \in C \quad (2.3)$$

where  $x_{nj k}$  is the feature data. MNL model uses an additional statistic method, like maximum likelihood method, to get the  $b_k$  value.

MNL suggests that each buyer has a taste of the population, and an individual taste to make a choice. Individual tastes make the buyer have a personalized choice with others. Nevertheless, MNL model focuses on the population taste and the individual taste is eliminated during the weights generation. In addition, MNL model needs a sample data to get the choice probability, and these data are not easily collected for studying a single buyer choice behavior. Therefore, we can't simply apply the MNL model for determining the individual's best offer in the semantic matchmaking area.

But, we still follow the idea of MNL, and define our individual interest model. The first task for us is to know what the individual taste looks like. We assume the individual taste is a special reflection of certain offer attributes data. We call these attributes data the interest focus. Following this assumption, we can define that individual interest as a non-linear function since McFadden believes the population interest is a linear function in the MNL model. Second, we need to know what kind of non-linear function it would be. Based on the human purchasing maximum utility idea, the utility for each attribute always increases followed by either the increasing or the decreasing of the attribute values. For instance, if a buyer considers buying an item based on the price, he will only

choose the lowest price instead of other prices based on this maximum utility idea. Thus, individual interest function for each attribute should be a smooth function: either continue growing up or continue dropping down. In Chapter 3, we show how we modify the MNL model to generate a new interest model that takes into account the individual's tastes and interests for multiple offer attributes.

## 2.3 Clustering Technology SOM

The self-organizing map (SOM) is one of the most widely used clustering methods [19]. SOM applies an unsupervised neural network model which forces neurons to become sensitive to the input data after the learning process based on the competitive learning [18]. In this section, we introduce the competitive learning and the SOM basic algorithm.

### 2.3.1 Competitive Learning

Competitive learning is an adaptive process through which the neurons in the neural network gradually learn to sensitively respond to input data [18]. The neurons used in SOM model can be represented as Learning Vector Quantizations (LVQs). In each step of the competitive learning, an input data  $x$  is added into the neural network, and all the LVQs try to simulate the input data. LVQs obtain learning rewards through the competition of other LVQs based on the distances. There exists a winner node ( $LVQ_c$ ) which is the closest LVQ to the input data  $x$ .

The  $LVQ_c$  gets most learning rewards by updating itself closer to the input data  $x$ , so that it can respond more strongly to the input data. The neighbourhoods of  $LVQ_c$  may also get corresponding but less rewards based on the distance.

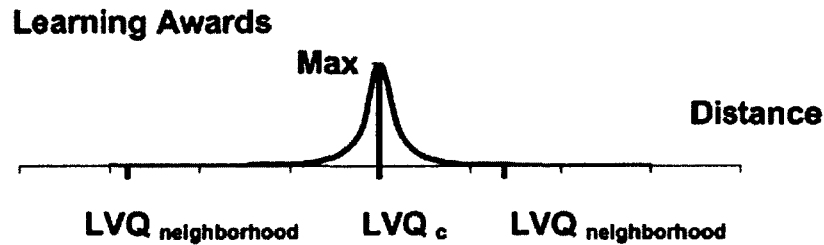


Figure 2.1: LVQ Learning Awards in Competitive Learning

As shown in Figure 2.1, LVQs gets learning rewards for the input data, but the rate of rewards decreases with the distance of  $LVQ_{neighbourhoods}$ . Such rewards make LVQs moving forward to the input data. With further training, LVQs can be more sensitive to all the input data. Data close to the same LVQ are gathered in the same clustering.

### 2.3.2 Self-Organizing Map Algorithm

The SOM algorithm employs the competitive learning method. When a data  $x$  is entered into the SOM function, the  $LVQ_c$  is determined to represent  $x$ , and then LVQs are updated closer to  $x$ . After a loop of such learning process, LVQs are good enough to represent all the input data. Finally, LVQs show data clustering patterns of the input data as the output (cf. Figure 2.2). The main algorithm will be discussed as follows.

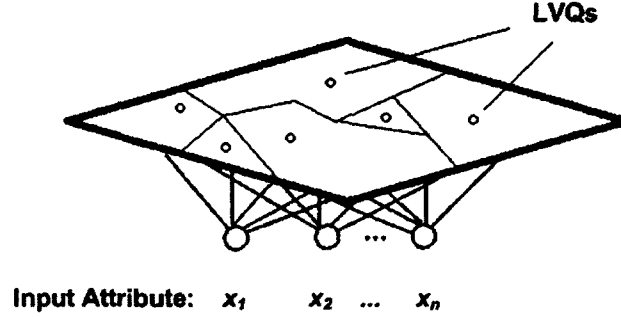


Figure 2.2: SOM Model Overview

The LVQs represent the input data  $x$  with reference vectors  $\overrightarrow{LVQ_i}$ , and the vector components correspond to the input data with a set of assigned weights. The input data  $x$  can be converted to a vector  $\vec{x}$ . Each  $\overrightarrow{LVQ_i}$  has the same dimensionality or components as the input vectors  $\vec{x}$ . For example, if  $\vec{x} = [x_1, x_2, \dots, x_n]$  is a sample of input data,  $n$  the dimension, then a reference vector of  $LVQ_i$  is  $\overrightarrow{LVQ_i} = [w_1, w_2, \dots, w_n]$ ;  $w_n$  is the assigned weight.

At each learning step, a data vector  $\vec{x}$  is entered. Distances between  $\vec{x}$  and  $\overrightarrow{LVQ_i}$  are calculated by the Euclidean distance function (cf. Formula (2.4)). The winner ( $LVQ_c$ ) is generated by formula (2.5) [18]:

$$\|\vec{x} - \overrightarrow{LVQ_i}\| = \sqrt{(x_1 - w_1)^2 + (x_2 - w_2)^2 + \dots + (x_n - w_n)^2} \quad (2.4)$$

$$\|\vec{x} - \overrightarrow{LVQ_c}\| = \min\{\|\vec{x} - \overrightarrow{LVQ_i}\|\} \quad (2.5)$$

Next, both the winner and neighborhood are updated, which make them much closer to the input data  $x$ , as shown by formula (2.6) [18]:

$$LVQ_i(t+1) = LVQ_i(t) + \alpha(t) \cdot h_i(t) (\|\vec{x} - \overline{LVQ_c(t)}\|, \|\vec{x} - \overline{LVQ_i(t)}\|) \quad (2.6)$$

where  $t$  denotes time,  $\alpha(t)$  is the learning rate and  $h_i()$  is the neighborhood function which decreases by learning time and neighborhood distance. At the start, the learning awards can allow LVQs moving widely and the width decreases during the learning time since the learning rate becomes smaller. The learning awards are based on the distance controlled by the neighborhood function. Thus, the closer LVQ to  $x$  gets more updates (cf. Figure 2.3).

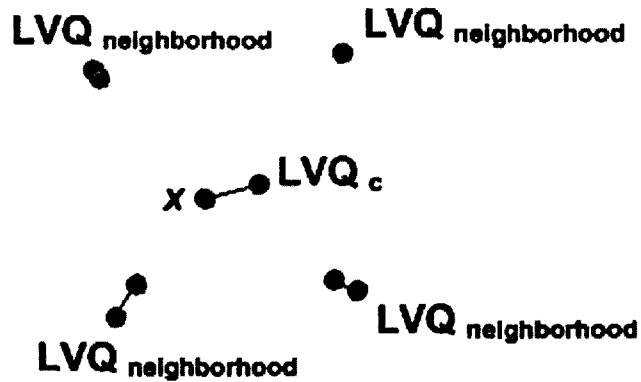


Figure 2.3: SOM Learning

In our work, we employ the clustering technology SOM so that buyers can easily point out their interests in a certain attribute data clustering. SOM helps our system to simulate buyers' interest rates for each attribute. In this

research, we consider an attribute as a high-dimensional data. SOM model is one of clustering technology which is good at clustering high-dimensional data onto lower-dimensional outputs. So that, the buyer can easily understand which high-dimensional attribute data are good enough to consider. In section 3.4.1, we propose a SOM algorithm to process a tree based clustering so that the buyer's selection can be defined in a certain level. The level represents the interest weight of an attribute.

## **Chapter 3**

# **INTEREST MODEL BASED OFFER EVALUATION SYSTEM**

In this chapter, we explain how our interest model is generated from the idea of the MNL model by taking account user's individual interests. Thanks to our interest model, the candidate offers can be evaluated and sorted according to the buyer's needs. Moreover, we show the benefits of our interest model.

### **3.1 Interest Model Construction**

As we discussed above, the MNL model suggests the utility function of a group of people choice behaviors and it needs a sample data. Individual interests can be different within the group's interests because of each individual's random interest



utility. Using only group's interests can cause more mismatched cases. Also, collecting sample data of individual choice behaviors can only generate very few appropriate data since each buyer may not purchase one kind of item frequently and the buyer's interest changes quickly. Such few data containing changeable factors can cause larger estimation error. Thus, MNL model is not good enough to be used in our work.

We need to define a new interest model for each individual by following the idea of the MNL model: an individual chooses an offer because such offer has the maximum utility or interests. At the market level, the individual taste  $\varepsilon$  is brought into the population model as a random utility and it is usually removed. Unlike the MNL model eliminating the individual taste  $\varepsilon$ , our new interest model is required to describe or at least simulate the changes of individual interest. Thus,  $\varepsilon$  is important for our interest model. In our model,  $\varepsilon$  can be considered as an additional interest caused by the individual taste, not a random value.

We adapt MNL model to generate our interest model based on following conditions. First, we consider all the offer attributes as the deterministic features  $K$ . In the MNL model, all the observed attributes are the deterministic features. Here, we assume the buyer's mentioned attributes in the request are the only deterministic features. Furthermore,  $\varepsilon$  can be decomposed into  $K$  attributes. In our work, we study how much interest rate on each attribute is influenced by the individual random utility. For example, if a buyer's salary, a random utility feature, is less than before, it may affect the total utility of such buyer having a choice. We are looking for how much the salary influences the interests of the

price deterministic attribute for the buyer. This means the evaluation of the same attributes may be different according to the tastes  $\varepsilon_{nj_k}$  of each individual (cf. Formula (3.1)).

$$U_{nj} = \sum_{k=1}^K b'_k \cdot (x_{nj_k} + \varepsilon_{nj_k}), \quad j \in C \quad (3.1)$$

In order to produce the interest model, we calculate the interest weights,  $IW$ , and simulates the interest rates  $IR$ .  $IW$  denotes the interest weight  $b'_k$  and  $IR$  generates the individual interest rate value  $(x_{nj_k} + \varepsilon_{nj_k})$ .  $IW$  describes the importance of each attribute in evaluation.  $IR$  simulates the interest rate changes by the individual taste. We propose Formula (3.2) to build our interest model  $IM$  for each individual.  $IM$  is the degree of interest of the buyer purchasing an offer  $j$  with  $K$  attributes. Based on the consumer theory, an individual seeks to maximize his utility in each purchasing behavior. In our work, buyer is looking for the offer with the maximal  $IM$ .

$$IM_j = \sum_{k=1}^K (IW_{jk} \cdot IR_{jk}), \quad j \in C \quad (3.2)$$

## 3.2 Interest Weight and Interest Rate Function

### Discussion

In this section, we discuss  $IW$  and  $IR$  functions, so that we can get a clearer picture of our interest model.

A linear function is usually used to measure the attributes' rates [1, 14, 15].

Using a linear function is an easy way to describe the buyer's interest distribution for different attributes at the category level.

However, linear functions can not define the different interest rate changes between attribute values within an attribute category. In some cases, linear utility functions cannot assign weights to attributes in order to make an offer as the best one. We use the example of Table 3.1 to explain this issue. We assume that we are looking for one stock from the following three stocks: Stock1 has lower risk and lower return, Stock2 is in the middle and Stock3 has higher risk with higher return.

Table 3.1: Stock Data Example

<i>Stock ID</i>	<i>Risk</i>	<i>Return</i>
1	-0.1	0.1
2	-0.3	0.3
3	-1	1

Here, we assume a buyer would choose Stock2 as the best offer. First, we try to use a linear utility functions to describe the buyer's choice by finding the weights for "Risk" and "Return" attributes. However, we demonstrate below that these weights do not exist to make Stock2 as the best offer.

$$\begin{aligned}
& \left\{ \begin{array}{l} \text{Stock2 is better than Stock1} \\ \text{Stock2 is better than Stock3} \end{array} \right. \\
\Rightarrow & \left\{ \begin{array}{l} (-0.3)w_{Risk} + 0.3w_{Return} > (-0.1)w_{Risk} + 0.1w_{Return} \\ (-0.3)w_{Risk} + 0.3w_{Return} > (-1)w_{Risk} + 1w_{Return} \end{array} \right. \\
\Rightarrow & \left\{ \begin{array}{l} 0.2w_{Return} > 0.2w_{Risk} \\ 0.7w_{Risk} > 0.7w_{Return} \end{array} \right. \\
\Rightarrow & w_{Risk} \text{ and } w_{Return} \text{ do not exist}
\end{aligned}$$

The result proves that linear utility function has its limitation: it cannot make sure that each offer has a chance to be the best offer. In order to solve this problem in this stock case, there are two presumptions for the solution. First, interest weights are changeable for different attribute values. In this case, the  $w_{Risk}$  for Stock2 may be lower than others, or  $w_{Return}$  for Stock2 may be higher. However, setting a set of changeable weights to one attribute is complex and hard in real life cases. Second, the utility function is not a linear function. Interest rate function  $IR$  for attribute *Risk* and *Return* should be a non-linear function. The attribute values for Stock2 goes through the non-linear functions, so that the sum of attribute utilities is higher than its linear function utility result. This non-linear function causes the final utility of Stock2 to be higher than other stocks.

We believe the second presumption is more reasonable since it is close to our

early assumption about individual tastes in Chapter 2 section 2.2. Individual tastes only cause attribute values in the interest focus area having an additional non-linear changes.

### 3.3 Attribute Data Extraction

In order to fulfill the matching, matchmaking systems utilize some semantic languages to understand and formulate the requests and offers. Some languages are widely used, especially in matching web services, such as Web Services Description Language (WSDL) [25], Semantic Web Services Language (SWSL) and Web Service Modeling Language (WSML) [26], which offer a high degree of flexibility and expressiveness.

**Example 1** *We have here a company, called Company1, which has the following inventory query:*

*Request computers with CPU > 1.5 GHz, Hard Drive > 100 GB.*

*Assuming Company1 uses WSDL language, the purchasing request is then specified in WSDL as shown in Figure 3.1. The formatted request is sent to the URL of the connected semantic matchmaking system. An operation “FindCandidateOffers” in the matchmaking system is called to match offers with the attribute constraints. The output message is declared and will be sent to our evaluation system.*

```

<wsdt:types> ...
  <xsd:element name="tns:MatchMakingRequest">
    <xsd:element name="Attribute" type="xsd:array" >
      <xsd:element name="CPU" type="xsd:array" greatThan="1.5 GHz" />
      <xsd:element name="Hard Drive" type="xsd:int" greatThan="100 GB" />
    </xsd:element>
  </xsd:element>
  <xsd:element name="tns:MatchMakingResponse">
    <xsd:element name="CandidateOffer" type="xsd:array" >
      <xsd:element name="AttributeData" type="xsd:array" />
    </xsd:element>
    <xsd:element name="EvaluateRequest">
      <xsd:element name="Attribute" type="xsd:array" >
        <xsd:element name="AttributeName" type="xsd:string" />
        <xsd:element name="AttributeDimension" type="xsd:int" />
        <xsd:element name="AttributeSortOrder" type="xsd:string" />
        <xsd:element name="AttributeBestAttributeData" type="xsd:int" />
      </xsd:element>
    </xsd:element>
  </xsd:element>
</wsdt:types>
<wsdt:portType name="FindCandidateOffersPortType">
  <wsdt:operation name="FindCandidateOffers">
    <wsdt:input message="tns:MatchMakingRequest" />
    <wsdt:output message="tns:MatchMakingResponse" />
  </wsdt:operation>
</wsdt:portType>
<binding name="FindCandidateOffersBinding">
  <soap:binding ... />
  <operation name="FindCandidateOffers">
    <soap:operation soapAction="SemanticMatchMakingSystemURL"/>
    <input> <soap:body ... /> </input>
    <output> <soap:body ... /> </output>
  </operation>
</binding>
<wsdt:service name="FindCandidateOffers">
  <wsdt:port binding="tns:FindCandidateOffersBinding" name="FindCandidateOffers">
    <soap:address location="EvaluationSystemServerURL" />
  </wsdt:port>
</wsdt:service>

```

Figure 3.1: Formatted Matchmaking Request

Getting attribute data is the first step for the whole evaluation process. In this thesis, we only take value-based attributes as the  $K$  attributes of our interest model. Other complex type attributes will be covered in our future research. Here, we use simple examples to promote our idea. The message sent

from the semantic matchmaking system should include the candidate offers and all evaluation-need attributes information. It is easy to extract these data from the message because most of the message is a XML based message, like SOAP. For each attribute data, our system extracts all its values from the candidate offers and stores them in a single table.

**Example 2** *Table 3.2 shows some examples of candidate offers responding to the purchasing request in Example 1. In this case, two attributes are used: CPU and Hard Drive, and CPU contains high dimensional data values.*

Table 3.2: Extracting Attribute Data from the Responding Message

<i>Supplier</i>	<i>CPU Table (GHz)</i>	<i>Hard Drive Table (GB)</i>
A	2.8	960
B	2.66	1000
C	2.4	1024
D	(3.2, 3.2)	500
...	...	...

## 3.4 Attribute Data Clustering

### 3.4.1 Clustering Algorithm

After determining all the attributes from the purchasing request, our system can now cluster the values of each attribute. The purpose of this clustering is to be able to determine the buyer's interest focus in each attribute. We believe the

buyer's selection is consistent with buyer's interest focus, when buyer is asked to give the most interested attribute data. Thus, the buyer can select one of the clustering to represent his interest focus. In Algorithm 1, we define a clustering function called *ClusterAttribute()* which is based on the Self Organizing Map (SOM) method [18]. We apply SOM to recursively divide a large clustering into three sub-clustering until there are less data in each sub-clustering. We include the function *IsLargeEnough()* to check whether the current clustering  $A_i$  is still large enough or not. If the range of  $A_i$  is greater than the minimal attribute clustering range, then we need to cluster  $A_i$ .

---

**Algorithm 1:** *ClusterAttribute(DataAttribute: Array)*

---

**input** : An array of attribute data

**output:** A tree of attribute clustering

```

1 MinRange ← Min(DataAttribute);
   // Calculate the minimum range
2 SOM(DataAttribute, A1, A2, A3);
   // Cluster data into 3 groups
3 ArrangClustering(A1, A2, A3);
   // Arrange clustering in ascending order
4 foreach element Ai do
5   | if IsLargeEnough(Ai, MinRange) then
6   |   | ClusterAttribute(Ai);
   |   | // Cluster each sub-group

```

---



---

**Algorithm 2:** SOM(*DataAttribute*: Array,  $A_1$ : Array,  $A_2$ : Array,  $A_3$ : Array)

---

**input** : An array of attribute data

**output:** Three sub-clustering for attribute data

```
1  $t \leftarrow 1$ ;  
   // Initialize learning time  
2 CreateLVQ(DataAttribute,  $LVQ_1$ ,  $LVQ_2$ ,  $LVQ_3$ );  
   // Create random  $LVQ_i$   
3 for Alpha( $t$ ) is not too small do  
   // Decrease Alpha( $t$ ) by time  $t$   
4   Clear  $A_i$ ;  
5   for  $x \leftarrow \text{Pop}(\textit{DataAttribute})$  do  
     // Pop  $x$  from data set  
6      $\|x - LVQ_c\| \leftarrow \text{Min}(\|x - LVQ_i\|)$ ;  
     // Find the closest  $LVQ_c$  to  $x$   
7     Add( $x$ ,  $A_c$ );  
     // Put  $x$  into the closest cluster  
8      $LVQ_i(t+1) \leftarrow LVQ_i(t) + h(t)[x(t) - LVQ_i(t)]$ ;  
     // Update  $LVQ_i$  to be close to  $x$   
9    $t \leftarrow t + 1$ ;  
   // increase time  $t$ 
```

---

In Algorithm 2, we propose an algorithm for SOM: during the learning time  $t$ , a Learning Vector Quantization ( $LVQ$ ) is generated for each sub-clustering. The function  $SOM()$  uses competitive-learning given in line 6 and updates  $LVQ$  during the learning time in line 8 [18].  $CreateLVQ()$  is a function that generates three random  $LVQ_i$  in the range of  $DataAttribute$ ;  $Alpha(t)$  controls the learning loop. It is the learning rate function which is decreased by learning time  $t$ ;  $LVQ_c$  is the closest  $LVQ$  to the selected data  $x$ ;  $h()$  is the “neighborhood” function that updates  $LVQ$  during the learning time [18]. In our work, the neighborhood function  $h(t)$  decreases gradually during the learning loop and neighborhood distance. It is defined as follows:

$$\begin{aligned}
 f_{FarthestNeighborhoodDistance}(x, LVQs) &= MAX(\|x - LVQ_n\|), \quad LVQ_n \in LVQs \\
 h(t)[x(t) - LVQ_i(t)] &= e^{-10 \cdot \frac{Distance(x, LVQ_i)}{f_{FarthestNeighborhoodDistance}(x, LVQs)}}
 \end{aligned} \tag{3.3}$$

In Appendix A, we give the implementation of  $SOM()$  in C#.

**Example 3** *Figure 3.2 shows the Hard Drive clustering tree generated by function  $ClusterAttribute()$ . Since the minimal range of Hard Drive clustering tree is 20, we should then decompose all the clustering that are larger or equal to 20. For example in level 3, the range of the cluster [620, 640] is 20, thus we divide it into two sub-clusters because there are only two values.*

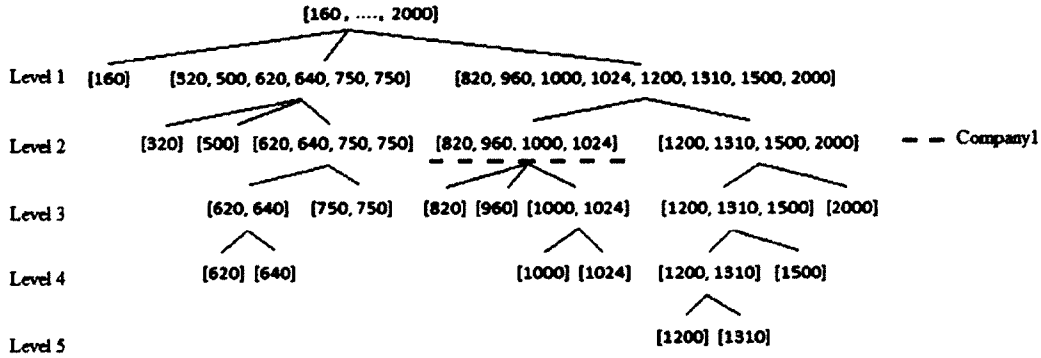


Figure 3.2: Hard Drive Clustering Tree

### 3.4.2 High Dimensional Data Discussion

The *ArrangeClustering()* function is used to order the clustering  $A_i$  in an ascending or descending order. For high dimensional data, it sorts the clustering based on the distance (cf. Formula 2.4).

To sort high dimensional clustering, our system needs to know the sorting order for each clustering. Since there are only value based attributes in this research, attribute sorting order can be either the higher the better or the lower the better. For instance, when buying a computer, price attribute can be the lower the better attribute, while CPU can be the higher the better. Then comparing all the attribute data, our system can find the best attribute value according to the sorting order. Ranking the clustering  $A_i$  can be based on the distance between  $LVQ_i$  to such best attribute value.

**Example 4** The three clustering  $A_1 = [2.2, \dots, 2.8]$ ,  $A_2 = [(1.8, 1.8), (1.9, 1.9)]$ , and  $A_3 = [(3, 3), (3.2, 3.2)]$  are sorted in Level1 in Figure 3.3. The distance

between each clustering related  $LVQ$  to the best CPU value (3.2,3.2):  $LVQ_1$  of 3.0489,  $LVQ_2$  of 1.6232,  $LVQ_3$  of 0.4295(the closest). Since  $LVQ_3$  is greater than  $LVQ_2$  and  $LVQ_2$  is greater than  $LVQ_1$ , three clustering are sorted as A1, A2 and A3 in an ascending order. The minimal range for CPU clustering tree is 0.03.

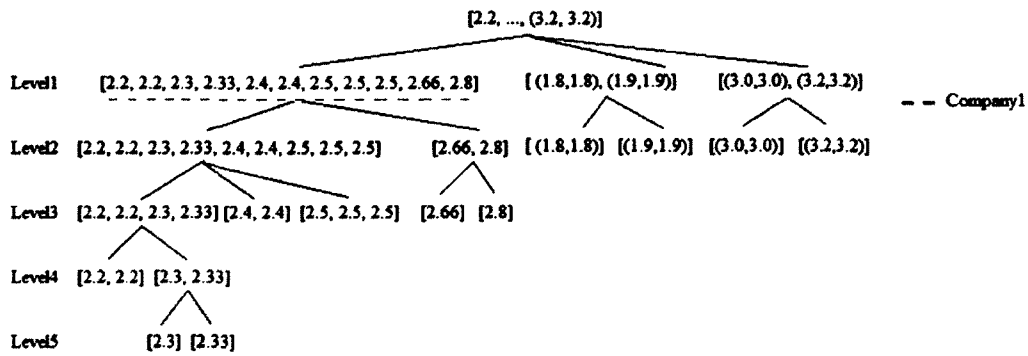


Figure 3.3: CPU Clustering Tree

### 3.5 Interest Weight Calculation

An interest weight denotes the degree of buyer's interest focus on an attribute in a matching. Assuming buyer's selection is consistent with buyer's interest focus,  $IW$  should be related to the depth and range of the buyer's selection. The smaller and deeper clustering selected shows the more interest focus on such attribute. In order to produce the  $IW$ , we use the interest-weight coefficient  $IW\_coe$  which contains the buyer's selection information in one attribute. We create Formula 3.4 to compute the  $IW\_coe$  of an attribute called  $k$ :  $K$  is the attribute set,

*DataAttribute* is the whole attribute value set of  $k$ , *SelectedClustering* is the buyer's selected clustering for  $k$ , *SelectedLevelTree* is the selected clustering level, and *TotalLevelTree* is the number of levels of the clustering tree. We define the function *Length()* to return the length of the attribute clustering, which is the absolute difference between two clustering endpoints.

$$IW\_coe_k = \frac{Length(DataAttribute_k)}{Length(SelectedClustering_k)} \cdot \frac{SelectedLevelTree_k}{TotalLevelTree_k} \quad k \in K \quad (3.4)$$

The coefficient has to be compared with the other attributes' coefficients (cf. Formula 3.5), so that system can identify how buyer's interest focus distribute among  $K$ . An attribute with a larger interest weight coefficient can get a larger interest weight.

$$IW_k = \frac{IW\_coe_k}{\sum_{k'=1}^K IW\_coe_{k'}} \quad k \in K \quad (3.5)$$

**Example 5** We suppose the company in Example 1 chooses the CPU clustering [2.2, ..., 2.8] (length: 0.6) as their interest focus on CPU. This selection is at level 1 of the 5-level CPU tree and all CPU data are in the set [2.2, ..., (3.2, 3.2)] (length: 3.35). So, the CPU interest weight coefficient can be calculated as 1.1167 with Formula 3.4.

$$\begin{aligned}
IW_{coe_{CPU}} &= \frac{Length(DataAttribute_{CPU})}{Length(SelectedClustering_{CPU})} \cdot \frac{SelectedLevelTree_{CPU}}{TotalLevelTree_{CPU}} \\
&= \frac{3.35}{0.6} \cdot \frac{1}{5} = 1.1167
\end{aligned}$$

We perform the same calculation for the Hard Drive attribute with a coefficient of 3.6078. According to Formula 3.5, we can get the attribute interest weights. For example, CPU and Hard Drive interest weight in this 2-attribute example is the following:

$$\begin{aligned}
IW_{CPU} &= \frac{IW_{coe_{CPU}}}{IW_{coe_{CPU}} + IW_{coe_{HardDrive}}} \\
&= \frac{1.1167}{1.1167 + 3.6078} \approx 0.2364
\end{aligned}$$

$$IW_{HardDrive} = \frac{3.6078}{1.1167 + 3.6078} \approx 0.7636$$

Based on these interest weights, we can conclude that attribute Hard Drive is much more important than CPU. Such interest feature will help our system to calculate the best offers for the company.

## 3.6 Interest Rate Function Generation

As we discussed earlier in section 3.2, a linear function may not have appropriate weights to make an offer as the best offer. To address this issue, we set our IR function as a non-linear function. In this thesis, we use the sigmoid function

$\varsigma_k(x) = \frac{1}{1+\exp^{-x}}$  to simulate each buyer's interest. We believe the sigmoid function is the closest function to human natural interest change.

In our sigmoid function,  $x$  denotes the value of an attribute and  $y$  its interest value. In Figure 3.4, we show that  $x$  of the sigmoid function has less changes in the two intervals  $[-\infty, -2]$  and  $[2, +\infty]$ . The Sigmoid function in these two intervals can be considered as a linear function with an acceptable standard error. Meanwhile the interval  $[-2, 2]$  is a quickly changeable area. We choosing  $\pm 2$  as the boundaries as promoted by [27].

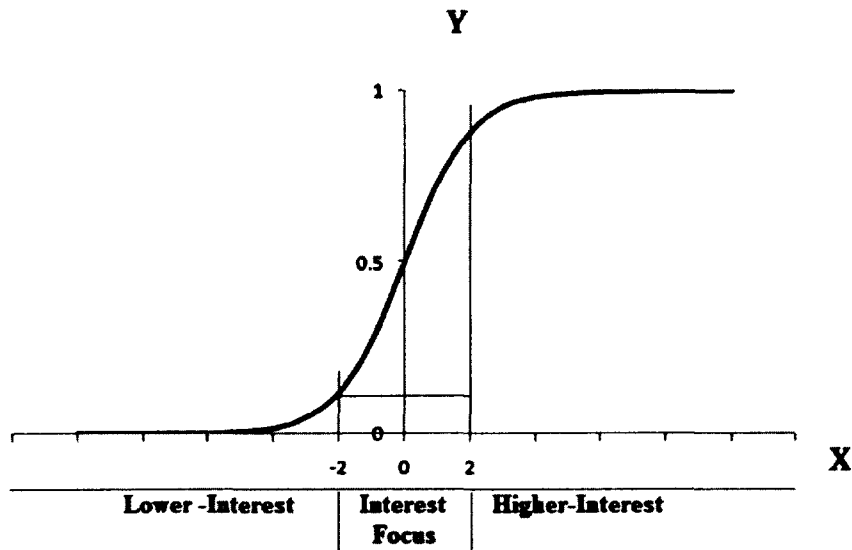


Figure 3.4: Sigmoid Function and Interest Interval

We employ the sigmoid function to simulate buyer's interest rate for each attribute. We divide buyer's interests change into three interest area: lower-interest area, higher-interest area and interest focus, in order to map all the

attribute interests with the three intervals in the sigmoid function. In order to represent IR function as a sigmoid based function, we need to bind the buyer's interest focus, i.e. the selected clustering, into the quickly changeable interval  $[-2, 2]$ .

Here, we explain how to generate the interest rate function for an attribute  $k$ , called  $\varsigma_k()$ .  $[AL, AR]$  denotes the whole attribute data and  $[L, R]$  the selected clustering. We utilize  $LVQ$  of the selected clustering as the center of the function  $\varsigma_k()$  since  $LVQ$  can be considered as a density center of  $[L, R]$ . The value of  $LVQ$  is computed with the learning loops of the  $SOM()$  algorithm (cf. line 8 in Algorithm 2).  $LVQ$  can be either inside the selected clustering  $[L, R]$  or outside. Binding  $[L, R]$  into interval  $[-2, 2]$  depends on the  $LVQ$  position situation.

If  $LVQ$  is inside  $[L, R]$ , we decompose the interest rate function into two functions:  $\varsigma_{k\_Right}()$  and  $\varsigma_{k\_Left}()$  (cf. Formula (3.6)) where:  $\alpha_{Left}$  is generated when binding  $[L, LVQ]$  into the interval  $[-2, 0]$ ;  $\alpha_{Right}$  when binding  $[LVQ, R]$  into  $[0, 2]$ .  $\alpha_{Left}$  controls the shape of the left side of the interest function, and  $\alpha_{Right}$  the right side.  $Sign$  is +1 or -1 w.r.t the ascending or descending order of the interest rate function (cf. Table 3.3).

If  $LVQ$  is outside of  $[L, R]$ , the interest rate function can be either  $\varsigma_{k\_Right}()$  or  $\varsigma_{k\_Left}()$  depending on the  $LVQ$  position: left outside or right outside of the selected clustering. For example, if  $LVQ$  is on the right side of  $[L, R]$ , we let  $\varsigma_{k\_Left}()$  control the whole shape of the function since  $\alpha_{Right}$  cannot be generated.



$$\varsigma_k() = \begin{cases} \varsigma_{k\_Left}() = \frac{1}{1+\exp(\alpha_{Left} \cdot \text{Sign} \cdot |x-LVQ|)} & x \in [AL, LVQ] \\ \text{And/Or} \\ \varsigma_{k\_Right}() = \frac{1}{1+\exp(\alpha_{Right} \cdot \text{Sign} \cdot |x-LVQ|)} & x \in [LVQ, AR] \end{cases}$$


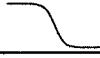
$$\alpha_{Left} \cdot |x - LVQ| = -1 \cdot (-2 - 0) \text{ when } x = L, L \text{ is binding to } -2 \quad (3.6)$$

$$\Rightarrow \alpha_{Left} = \frac{2}{|L - LVQ|}$$

$$\alpha_{Right} \cdot |x - LVQ| = -1 \cdot (2 - 0) \text{ when } x = R, L \text{ is binding to } 2$$

$$\Rightarrow \alpha_{Right} = \frac{-2}{|R - LVQ|}$$

Table 3.3: Interest Rate Functions Shape

Attribute $k$ Sorted by	Sign	Function Image
Ascending	+1	
Descending	-1	

**Example 6** *The selected Hard Drive clustering [820, 1024] for Company1 is bound into the area [-2, 2]; LVQ of 1042.6457 is the center point for the interest function. After data binding, the interest rate function shown in Figure 3.5 is smoother than the original one. Based on this function, we can easily get the*

company's interest rate for each Hard Drive attribute value.

$$\varsigma_{HardDrive}(x) = \begin{cases} \varsigma_{Left}(x) = \frac{1}{1 + \exp(0.0089 \cdot |x - 1042.6457|)} & x \in [160, \dots, 2000] \\ \varsigma_{Right}(x) \text{ does not exist} \end{cases}$$

$$\alpha_{Left} = \frac{2}{|820 - 1042.645|} = 0.0089$$

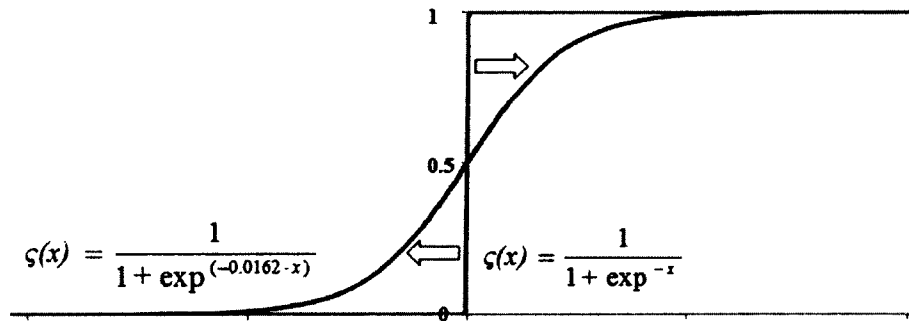


Figure 3.5: Binding Interest Rate Function

For high dimensional attribute interest rate function,  $x$  denotes the Euclidean distance between attribute data to the best attribute data instead of the high dimensional values. Using distance, we can apply the interest rate function to high dimensional data.

**Example 7** Based on the CPU attribute tree, the selection  $[2.2, \dots, 2.8]$  has the LVQ of  $(2.2654, 0.2979)$ . Here, CPU clustering  $(3.2, 3.2)$  is the best attribute

data.

Table 3.4: Distances of High-Dimensional Data

	$x$	$BestAttributeData$	$Distance(x, BestAttributeData)$
L	2.2	(3.2, 3.2)	3.3526
R	2.8	(3.2, 3.2)	3.2249
LVQ	(2.2654, 0.2979)	(3.2, 3.2)	3.0489

According to the distance calculation by Formula (2.4) in Table 3.4, LVQ is closer to the best attribute data (3.2, 3.2) than any attribute data in the selected clustering, that is right outside of  $[L, R]$ . Thus, the following interest rate function for attribute CPU has only  $\alpha_{Left}$ . We will give more details regarding CPU's IR functions in Chapter 5.

$$\varsigma_{CPU}(x) = \begin{cases} \varsigma_{Left}(x) = \frac{1}{1 + \exp(\alpha_{Left} \cdot \text{Sign} \cdot |Distance(x, BestAttributeData) - Distance(LVQ, BestAttributeData)|)} \\ \varsigma_{Right}(x) \text{ does not exist} \end{cases}$$

$$= \frac{1}{1 + \exp^{6.5848 \cdot |Distance[x, (3.2, 3.2)] - 3.0489|}} \quad x \in [2.2, \dots, (3.2, 3.2)]$$

$$\alpha_{Left} = \frac{2}{|Distance(L, BestAttributeData) - Distance(LVQ, BestAttributeData)|}$$

$$= \frac{2}{|3.3526 - 3.0489|} = 6.5848$$

### 3.7 Offers Evaluation with the Interest Model

Once  $IW$  and  $IR$  functions are set up, our interest model can evaluate the candidate offers. The interest model calculates the total interest rate of each offer, and finds the best offer based on the interest rate  $IM$ .

**Example 8** After our system gets the interest weights (cf. Example 5) and interest rate functions for the two attributes (cf. Example 6 and 7), it generates the interest model ( $IM$ ) for Company1 as follows.

$$\begin{aligned}IM_{Company1}(Supplier) &= IW_{CPU} \cdot IR_{CPU}(CPU) + IW_{HardDrive} \cdot IR_{HardDrive}(HardDrive) \\ &= 0.2364 \cdot \varsigma_{CPU}(CPU) + 0.7636 \cdot \varsigma_{HardDrive}(HardDrive)\end{aligned}$$

Here, we show below how the interest model calculates the interest rate and find the best offer only from the three suppliers A, B, and C.

$$\begin{aligned}IM_{Company1}(Supplier A) &= 0.2364 \cdot \varsigma_{CPU}(2.8) + 0.7636 \cdot \varsigma_{HardDrive}(960) \\ &= 0.2364 \cdot 0.2388 + 0.7636 \cdot 0.3225 = 0.3027\end{aligned}$$

$$\begin{aligned}IM_{Company1}(Supplier B) &= 0.2364 \cdot \varsigma_{CPU}(2.66) + 0.7636 \cdot \varsigma_{HardDrive}(1000) \\ &= 0.2364 \cdot 0.2153 + 0.7636 \cdot 0.4054 = 0.3605\end{aligned}$$

$$\begin{aligned}
 IM_{Company1}(SupplierC) &= 0.2364 \cdot \varsigma_{CPU}(2.4) + 0.7636 \cdot \varsigma_{HardDrive}(1024) \\
 &= 0.2364 \cdot 0.1620 + 0.7636 \cdot 0.4582 = 0.2845
 \end{aligned}$$

*So, we got an interest rate of 0.3605 for Supplier B's offer, 0.3027 for A's, 0.2845 for C's. So, we can conclude that Supplier B has a higher chance than Supplier A and C to become the Company1's partner. With our interest model, we can evaluate all the candidate suppliers' offers.*

*We also see that using a linear function to evaluate this three-offer cases can not make Supplier B's offer better than offer of A or C because of linear function limitation. Based on the analysis of the company's interests, our interest model result seems to be more reasonable.*

## Chapter 4

# DESIGN AND IMPLEMENTATION

In this chapter, we design and implement our interest-model-based offer evaluation and ranking system. We provide a platform so that features of the individual's interests can be exposed during the interaction with our system. Also, our system uses the features to create the interest model, and then rank the offers. Finally, our system recommends the best matched offer result.

### 4.1 High-level Design

We developed our system with a distributed architecture as illustrated in Figure 4.1. The buyer interacts with the client side via the Graphical User Interface (GUI). After the buyer submits his request, the GUI passes the formatted request to the connected semantic matchmaking system. A responding message, which

contains the matchmaking results, is sent to our system server side.

On the server side, the *OfferManager* component: (1) analyzes the responding message to understand the attributes and offers information, (2) stores the request-matched offers in the *OfferContent* database, and (3) extracts the offer attribute values into the *OfferAttribute* database. For each attribute, the *AttributeData-Clustering* component clusters the attribute values and sends its clustering tree to the client side.

After the client side receives the clustering trees, GUI helps the buyer to select the most interested clustering. Once the buyer's selection is completed, the *OfferEvaluator* component is called to build the buyer's interest model. It first passes all the selected clustering and the whole attribute clustering trees to *InterestWeightCalculator* and *InterestRateFunctionCreator*. For each attribute, *InterestWeightCalculator* returns its interest weight coefficient and interest weight, and *InterestRateFunctionCreator* generates its interest rate function. Then, the *OfferEvaluator* component applies the generated interest model on the request-matched offers, and returns to the buyer a list of offers sorted by individual interests.

## 4.2 Extracting Attribute Data

The message sent from the semantic matchmaking system to our system server side is required in the buyer's formatted purchasing request (cf. Figure 3.1), so that the matchmaking system knows what kind of information needs to be

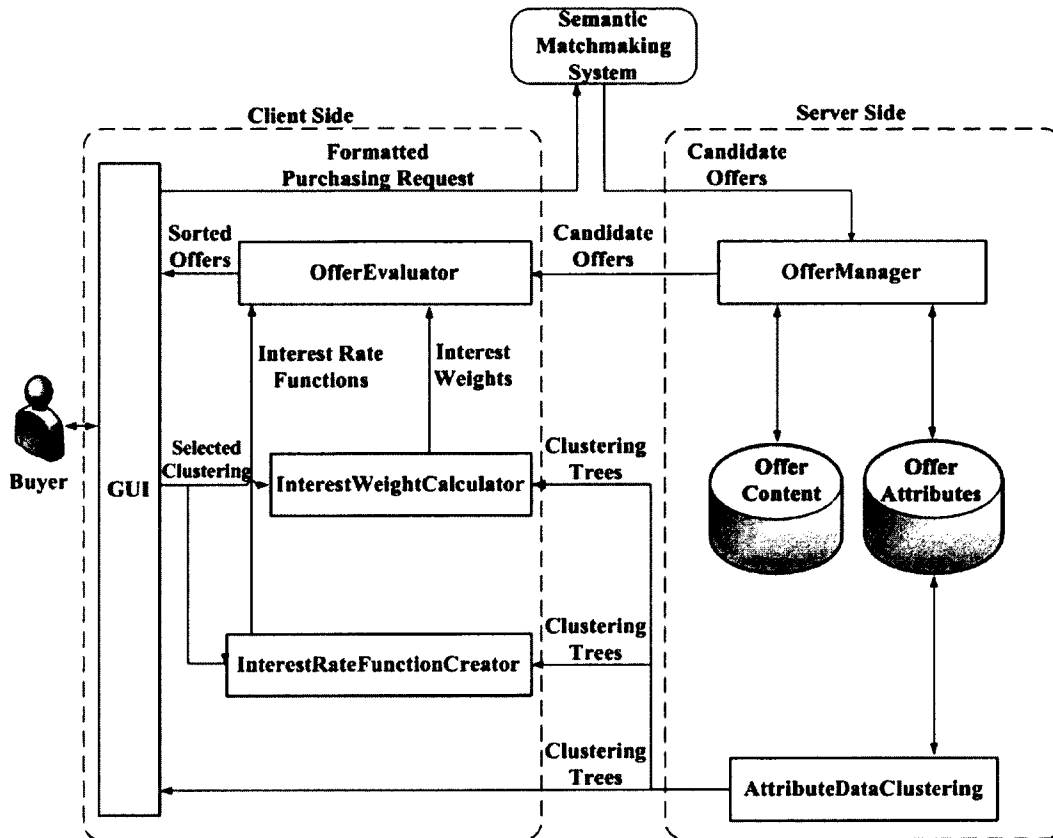


Figure 4.1: An Interest-based Offer Evaluation System

provided to support our evaluation work. Usually, the responding message is a XML-based file, such as SOAP message. The message body contains two important parts: attribute information and offers (cf. Figure 4.2). The attribute part declares how many offer attributes are considered to be the evaluation attributes  $K$  and the attribute features, including *Name*, *Dimension*, *SortingOrder*, and *BestAttributeData*, which help ranking high-dimensional attribute data. The offer part contains all the necessary offer attribute information.



```

<soap:Body xmlns:m="">
  <m:SemanticMatchmakingResponse>
    <m:EvaluateRequest>
      <m:Attribute>
        <m:AttributeName>CPU</m:AttributeName>
        <m:AttributeDimension>2</m:AttributeDimension>
        <m:AttributeSortOrder>ASC</m:AttributeSortOrder>
        <m:AttributeBestAttributeData>(3.2, 3.2)</m:AttributeBestAttributeData>
      </m:Attribute>
      ...
    </m:EvaluateRequest>
    <m:CandidateOffer Offer_ID = "1">
      <m:AttributeData>
        <CPU>2.2</CPU>
        <RAM>2</RAM>
        <HardDrive>320</HardDrive>
        <Price>420</Price>
      </m:AttributeData>
    </m:CandidateOffer>
    ...
  </m:SemanticMatchmakerResponse>
</soap:Body>

```

Figure 4.2: Example of Semantic Matchmaking Responding Message

Component *OfferManager* is the first component taking over the responding message. The first task is to understand the attribute input parameters. The name of the attribute defines from which XML tags the system can get the attribute values. The dimension of attribute data defines how many dimensional values are needed to extract a complete offer attribute data from the responding message. The rest of the attribute feature parameters control the clustering arrangement in the tree, like sorting order, and the best attribute value. They will be passed to other related components.

With the information of attribute name and dimension, *OfferManager* can

collect the attributes values one by one from the candidate offers. Creating tables and sorting the candidate offers into the *OfferContent* database is the next step. Some offers may contain more attribute contents than what the buyer required, so we need to store each offer completely in the database. Once the interest model is generated, offers in database will be evaluated through the interest model.

Also, attribute data need to be extracted from each offer as the input of other components. System sorts the attribute data separately into each attribute table created in *OfferAttribute* database. Related Attribute tables, SOM Clustering tables and Attribute Tree tables are also created for *AttributeDataClustering* component. Figure 4.3 shows the relationships between these tables.

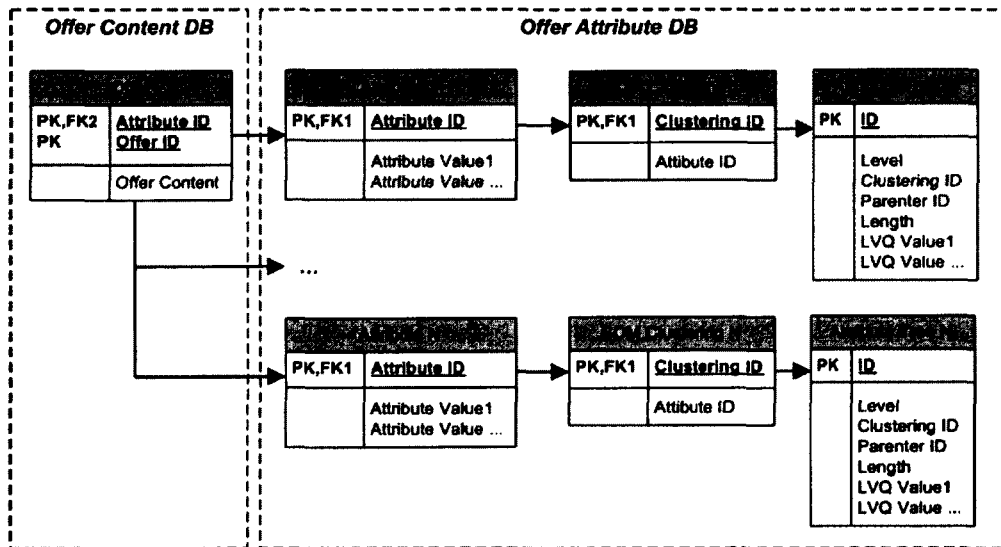


Figure 4.3: Relationships between Tables in Server Databases

## 4.3 Main Component Design

On the server side, *AttributeDataClustering* component is also a very important part. It provides the attribute clustering trees to help the buyer find his interest focus distribution. The other important components for creating an interest model are on the client side. In Figure 4.4, we show for instance the offer evaluation process on the client side. In the next sections, we will introduce these main components: *AttributeDataClustering*, *InterestWeightCalculator* and *InterestRateFunctionCreator*.

### 4.3.1 Attribute Data Clustering

The goal of the *AttributeDataClustering* component is to cluster the values of each attribute and send each clustering tree to the client side. With these clustering trees, the buyer can select one of the attribute clustering to represent his interested focus, and our system is able to get these features. The *AttributeDataClustering* component applies Algorithm 1 defined in section 3.4.1, and connects with the Attribute tables, SOM Clustering tables, and the Attribute Tree tables (cf. Figure 4.5).

The first step is to get all the attribute information from the matchmaking responding message, so that system can pick up attribute data for each attribute table. Some attribute data can be high dimensional. Thus, attribute dimension is also very important when the *AttributeDataClustering* component takes the attribute data from Attribute tables.

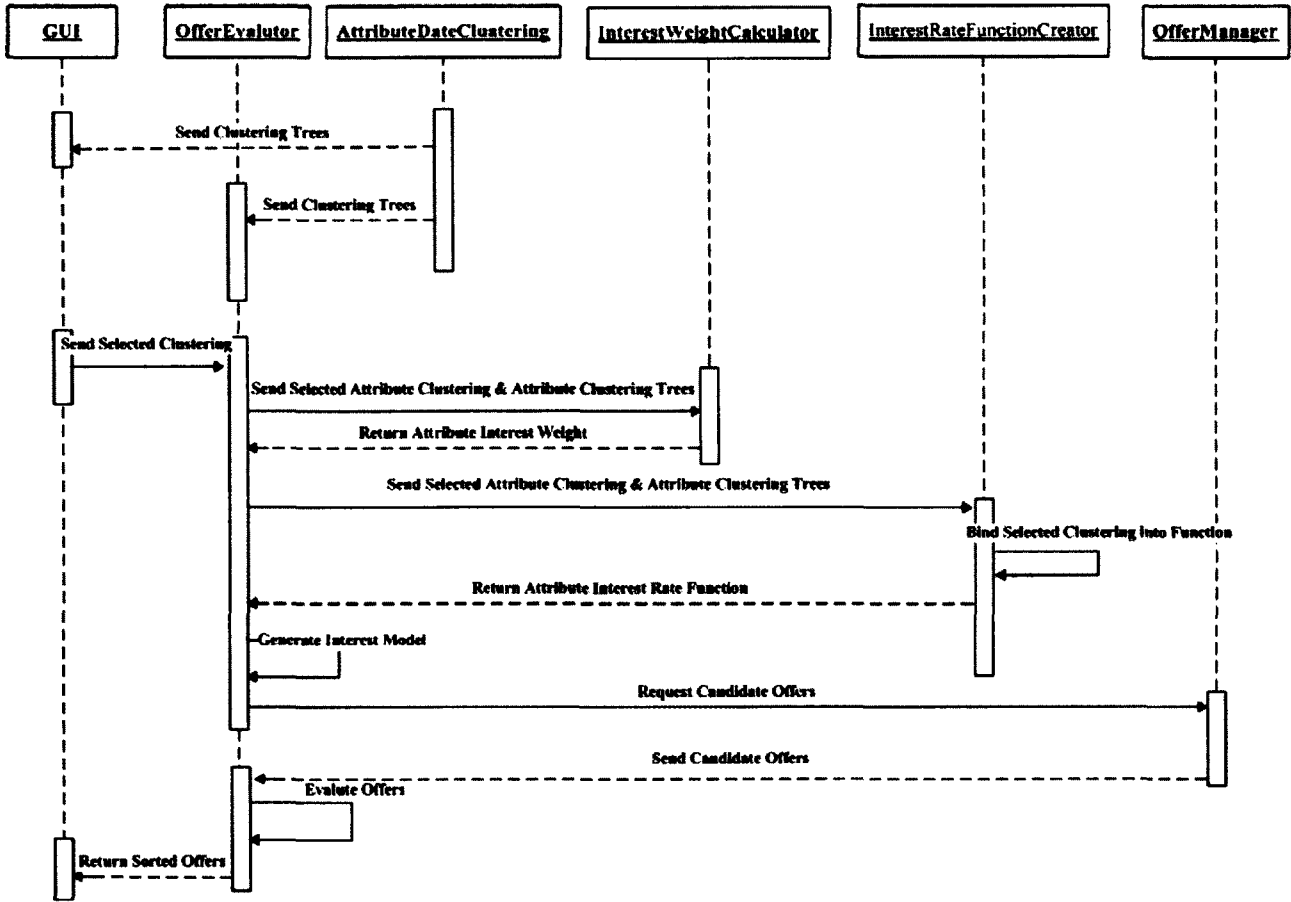


Figure 4.4: Offer Evaluation Sequence Diagram

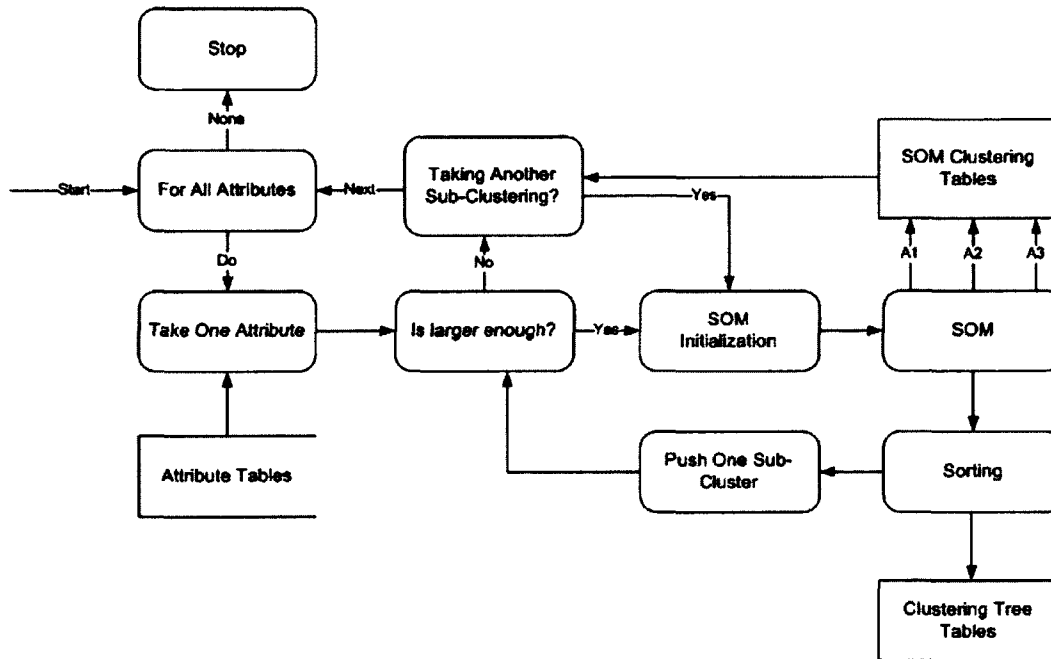


Figure 4.5: *AttributeClustering* Component Processing

For each attribute, if the length of attribute data set is no less than the minimal clustering distance (Line 1, Algorithm 1), we divide this attribute data set into three sub-clustering. *AttributeDataClustering* component creates the attribute tree by initializing the SOM data and call *SOM()* function to cluster the input attribute data set. The *SOM()* function (Algorithm 2) generates three sub-clustering after a competitive learning process. These clustering are stored into the SOM Clustering table.

Sub-clustering  $A_i$  can be arranged by sorting the *LVQ* value (or *LVQ* distance value). The sorting order can be obtained from the matchmaking responding message. The sorted three clustering are children of their parent clustering at

this level. The clustering positions are saved into the Clustering Tree table.

If one of the sub-clustering is still large enough to be cluster, the *SOM()* function is called recursively. The clustering tree is built completely unless no sub-clustering is able to be divided.

### 4.3.2 Interest Weight Calculator

The role of *InterestWeightCalculator* component is to calculate each attribute weight based on the buyer's interest selection on the clustering trees. The formulas are listed previously in Chapter 3 Section 3.5. *InterestWeightCalculator* component needs to obtain the buyer's selection to fulfill the calculation, and store the *IW* in table.

Figure 4.6 shows the whole process inside the *InterestWeightCalculator* component. With the buyer's attribute selection on the tree, *InterestWeightCalculator* component can calculate the *IW\_coe* according to Formula (3.4). From the attribute clustering tree, the length and the level of the total attribute clustering can be attained and donated as variable *DataAttribute* and *TotalLevelTree*. The buyer selection is one sub-clustering of the tree. Thus, *InterestWeightCalculator* can also easily get the rest of the variables: *SelectedClustering* and *SelectedLevelTree*. After getting all attribute *IW\_coe*, *IW* can be calculated by Formula (3.5). Table Interest Weight stores all the calculation results (cf. Figure 4.7).

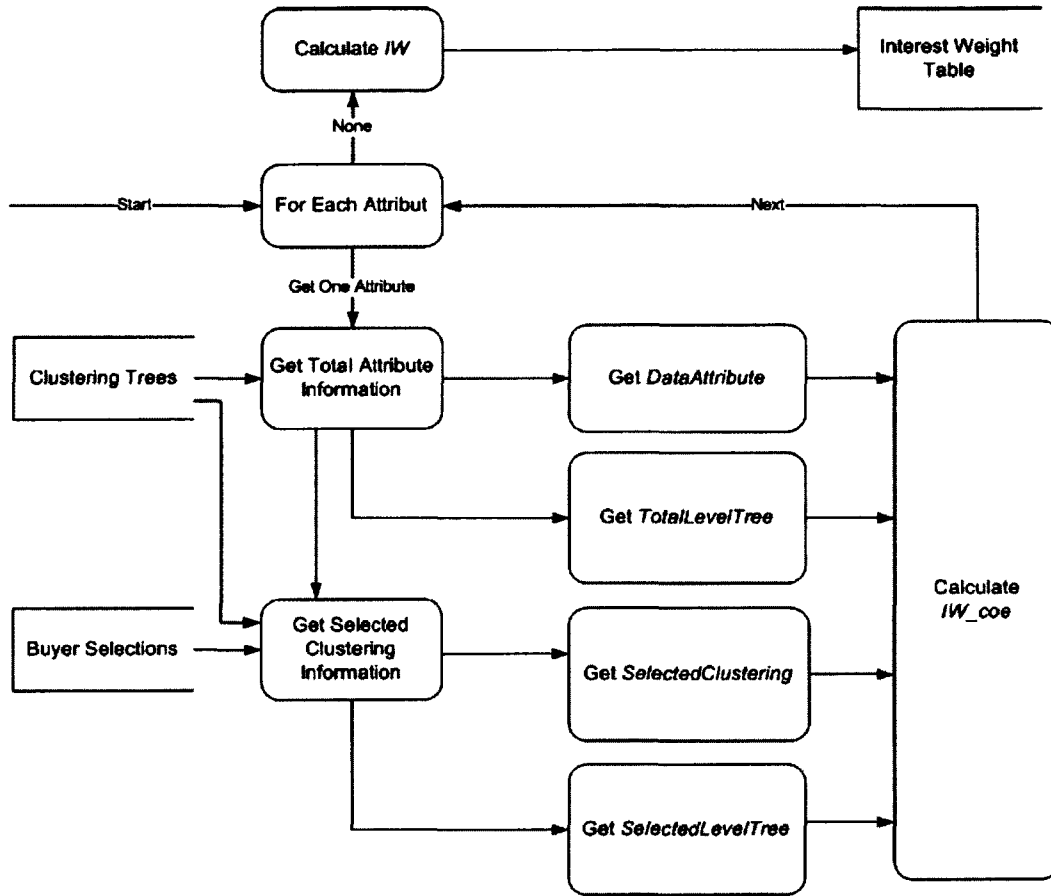


Figure 4.6: *InterestWeightsCalculator* Component Processing

### 4.3.3 Interest Rate Function Generator

*InterestRateFunctionCreator* component is another main component to build the individual interest model. The task of this component is to set up an un-linear interest rate function based on the buyer's selection. Figure 4.8 shows its creating process for one attribute.

After getting attribute clustering tree and selection information, component

Company_ID	Attribute	SelectedCluster...	DataAttribute	SelectedLevel...	TotalLevelFree	IW_COE	IW
1	CPU	0.6	3.35	1	5	1.1167	0.0649
1	RAM	2	14	1	3	2.3333	0.1356
1	Hard Drive	204	1840	2	5	3.6078	0.2096
1	Price	205.91	3484	3	5	10.1521	0.5899
2	CPU	0.28	3.35	1	5	2.3929	0.2349
2	RAM	7	14	1	3	0.6667	0.0555
2	Hard Drive	300	1840	3	5	3.68	0.3513
2	Price	404.48	3484	2	5	3.4454	0.3383

Figure 4.7: Interest Weights Calculation for Two Companies

*InterestRateFunctionCreator* can judge whether the interest rate function has  $S_{Left}$ ,  $S_{Right}$ , or both. The positions of  $L$ ,  $R$ , and  $LVQ$  are used to determine the function structure. If  $\alpha_{Left}$  and/or  $\alpha_{Right}$  exist, then they can be calculated. Attribute feature, *SortingOrder*, can help the component to define the variable *sign*. With these variables, interest rate function can be created based on Formula (3.6). The next step is to classify the attribute data: determining which attribute data belongs to left function or right function. Finally, variables of interest function are stored in Interest Rate Function table of client side.



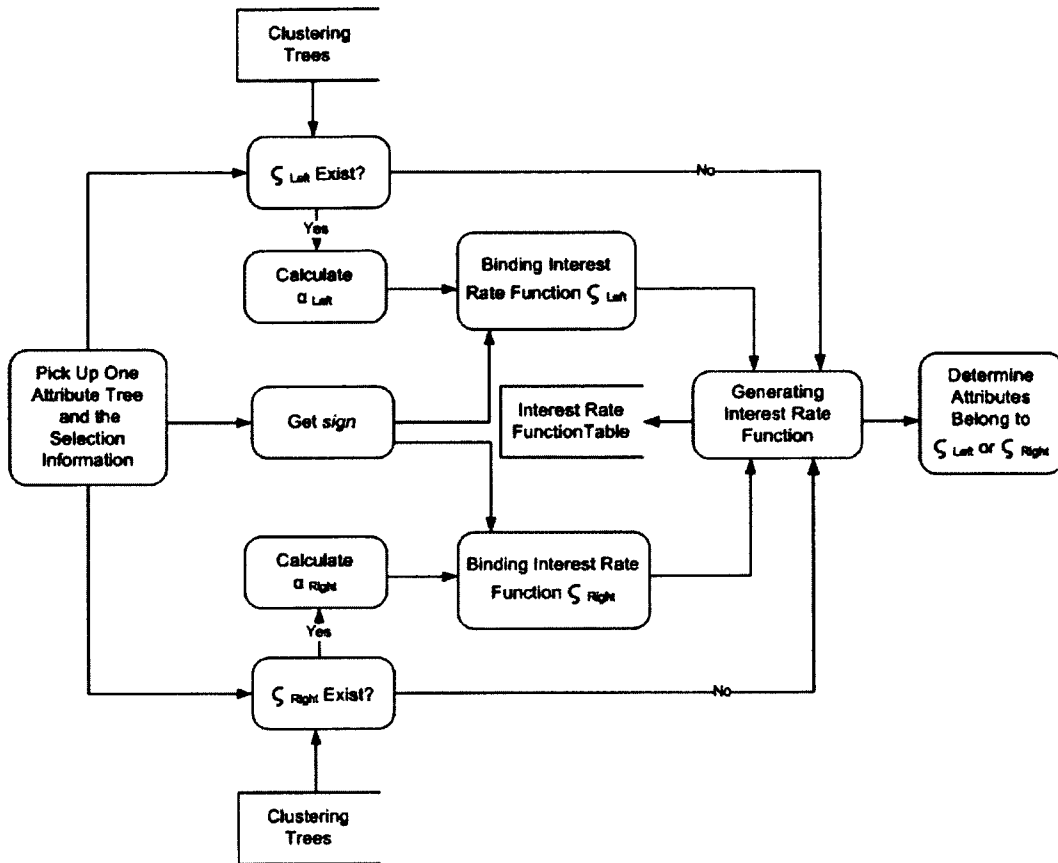


Figure 4.8: *InterestRateFunctionCreator* Component Processing

## 4.4 Implementation

### 4.4.1 Implementation Environment

We implemented our system using the language C# on Windows environment (.NET 3.5 framework). We employed the editor software Visual Studio 2008, and the SQL Server 2008 the database management system to store all the databases for server side and client side. We describe the GUI with XAML files. We also

used the graphic design tool called Expression Blend 3 to help us generate these graphic interfaces.

#### 4.4.2 Implementation and GUI

Figures 4.9 and 4.10 show the class diagrams of the client and server side programs. We created two separate databases source classes, called *serverDB* and *clientDB*, to support server and client side programs. These two classes contain all the necessary functions about database processing and data binding. The classes *ServerWindow* and *ClientWindow* contain multi-thread and network communication functions.

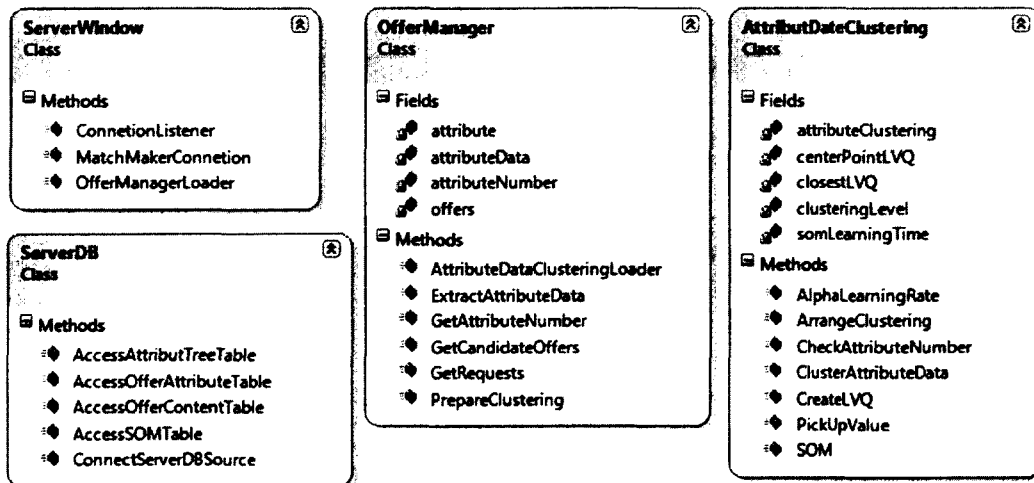


Figure 4.9: Server Side Class Diagram

Each system component is created within a window interface using Blend3, the interface developing tool for Windows Presentation Foundation (WPF) ap-

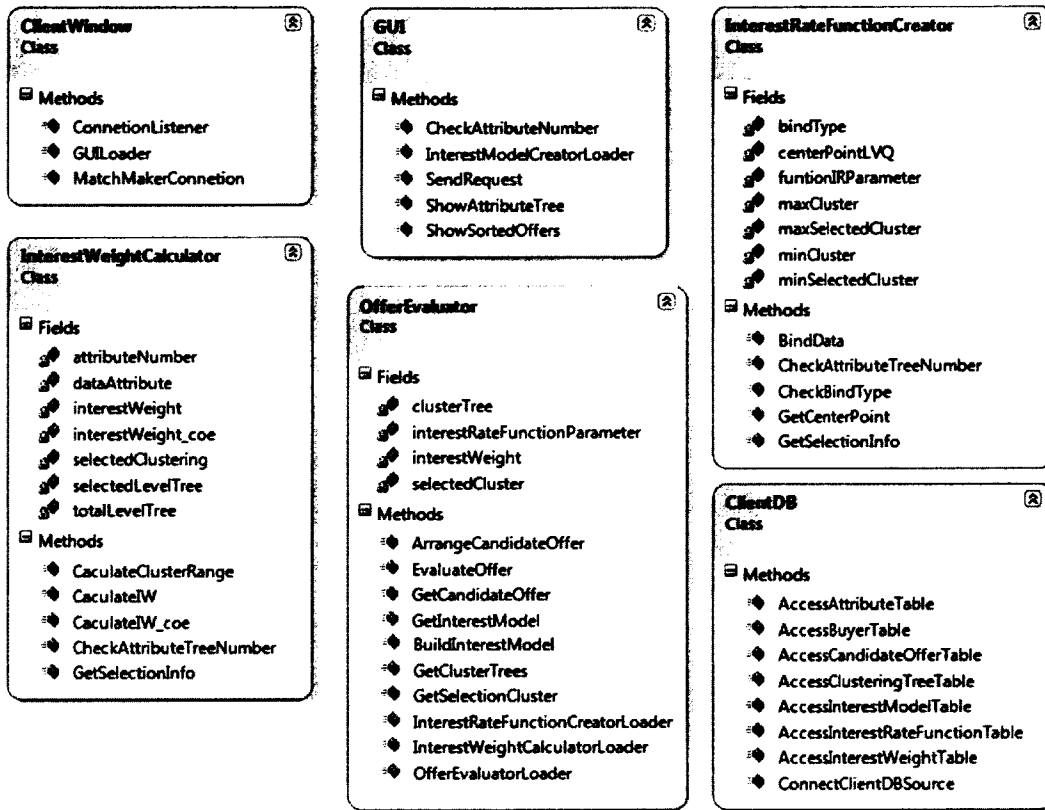


Figure 4.10: Client Side Class Diagram

plication. The benefit of this is we can test each component separately, and make sure they are working properly. The most used component interface for the buyer is the GUI (cf. Figure 4.11).

In GUI, all the attribute clustering trees are shown in the TreeView control in WPF. Buyers can browse the tree and select the interested clustering. The selection can be passed to the *OfferEvaluator* which is launched to build the interest model.

**Client Side**

**CPU**

- 2.2 0, 2.2 0
- 2.33 0, 2.3 0, 2.4 0, 2.4 0, 2.5 0, 2.5 0, 2.5 0
- 2.66 0, 2.6 0
- 1.8 1.8, 1.9 1.9
- 3.0 3.0, 3.2 3.2

---

**RAM**

- 2
- 3
- 4, 4
- 6, 6, 6, 6, 8
- 9, 10, 12, 12, 12, 16

---

**Hard Drive**

- 160
- 320, 500, 620, 640, 750, 750
- 820, 960, 1000, 1024, 1200, 1310, 1500, 2000
- 820
- 820
- 960
- 1000, 1024
- 1200, 1310, 1500, 2000

---

**Price**

- 420 400, 470 370, 600 500, 680 680, 800 700, 999 799, 950 900, 1030 830, 1000 880, 1100 799, 1100 800, 1200 1150
- 420 400, 470 370, 600 500, 680 680, 800 700
- 420 400, 470 370, 600 500
- 420 400
- 600 500
- 470 370
- 680 680, 800 700
- 999 799, 950 900, 1030 830, 1000 880, 1100 799, 1100 800, 1200 1150
- 2500 2200, 2900 2600, 3200 2500

---

**Offers Evaluation**

1	2.2, 0	2	320	420, 400	0.5433
2	2.2, 0	3	640	470, 370	0.5355
3	2.5, 0	4	500	600, 500	0.4510
15	3.2, 3.2	12	2000	3200, 2500	0.4101
13	3.0, 3.0	16	1500	2900, 2600	0.4067
4	2.33, 0	8	1310	1100, 800	0.3409
12	1.9, 1.9	10	1000	800, 700	0.3384

Figure 4.11: GUI for Clustering Selection

# Chapter 5

## EXPERIMENTATION

In this chapter, through a case study we show how our system analyzes different individual interests, and uses the interest features to create different interest models. With these models, the evaluation and sorting of the request-matched offers become more personalized.

### 5.1 Case Study: Computers Purchasing Based on Multiple Attributes

The experimentation consists of computers purchasing based on four attributes: CPU, RAM, Hard Drive and Price. We present both two companies, Company1 and Company2, submit the same purchasing request:

*Request a computer with CPU > 1.5 GHz, RAM ≥ 2.0 GB, HARD DRIVE*

*> 100 GB, Price < \$ 3500 for the first ten purchased computers and Price < \$ 3000 for the next ten.*

Assuming the purchasing request is formatted in a service language, our system sends it to the linked semantic matchmaker and waits for the responding message.

### **5.1.1 Extracting Attribute Data**

The semantic matchmaking system responses to both companies is a XML based messages, like the SOAP message shown in Figure 4.2. It includes two parts: the description of the four attributes, and the description of 15 candidate offers and their attributes. In the attribute description part, we can get all interested attributes such as name, dimension, sorting order, and best attribute value. These parameters help our system generate the interest model. In the offer attributes part, 15 candidate offers are listed one by one with attribute values. Table 5.1 simply lists the values extracted from the responding message and stored in attribute tables.

### **5.1.2 Clustering Attribute Data**

Each extracted attribute data set becomes the input for creating an attribute clustering tree. Figure 4.3 shows the tables containing the generated attribute clustering trees in the database on the server side. The clustering trees for attribute Hard Drive and CPU are given in Figure 5.1 and 5.2. Figures 5.3 and 5.4 give us respectively the clustering trees of the other two attributes: Price and

Table 5.1: Example of Attribute Data

Attributes	Name	CPU(GHz)	RAM(GB)	Hard Drive(GB)	Price(\$)
	Dimension	2	1	1	2
	Sorting Order	ASC	ASC	ASC	DES
	Best Attribute Data	(3.2,3.2)	16	2000	(420,400)
Offers	Offer ID	CPU	RAM	Hard Drive	Price
	1	2.2	2	320	(420,400)
	2	2.2	3	640	(470,370)
	3	2.5	4	500	(600,500)
	4	2.33	8	1310	(1100,800)
	5	2.4	8	750	(999,799)
	6	2.5	6	750	(1030,830)
	7	2.66	12	1024	(2500,2200)
	8	2.3	12	960	(1000,880)
	9	2.5	6	820	(1100,799)
	10	2.4	4	1200	(950,900)
	11	2.8	6	620	(1200,1150)
	12	(1.9,1.9)	10	1000	(800,700)
	13	(3.0,3.0)	16	1500	(2900,2600)
	14	(1.8,1.8)	9	160	(680,600)
	15	(3.2,3.2)	12	2000	(2100,2500)
Tables		attrCPUtbl	attrRAMtbl	attrHardDrivetbl	attrPricetbl

RAM. Attribute Price is two dimensional data here. Its minimal clustering distance is 6.5 and the clustering tree has 5 levels. RAM clustering tree is much more simple, only 3 levels, because of the repeated data, and RAM minimal attribute distance is 1.

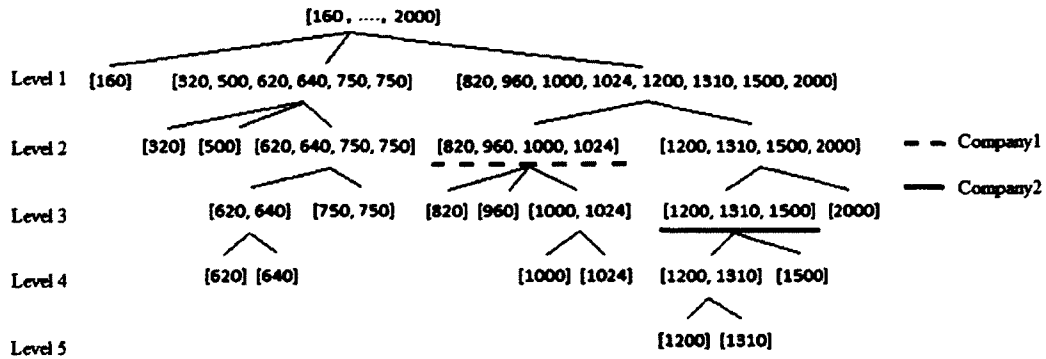


Figure 5.1: Hard Drive Clustering Tree

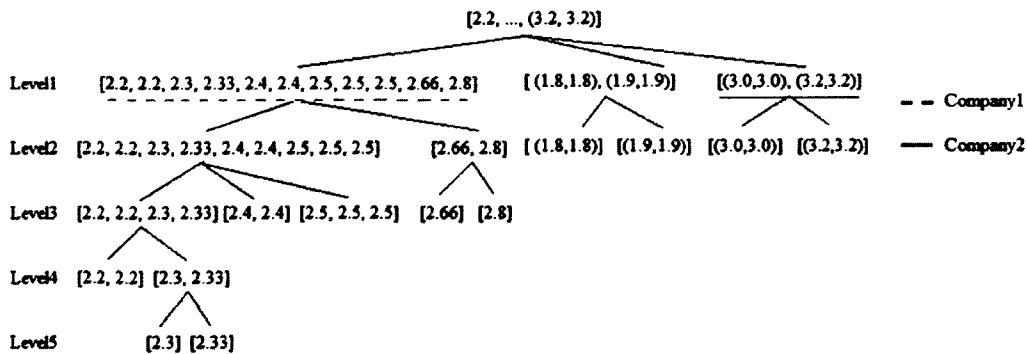


Figure 5.2: CPU Clustering Tree

### 5.1.3 Selecting Attribute Clustering

The buyer selects his interest forces on each attribute clustering tree through the GUI (cf. Figure 4.11). We display both company's selections in Figure 5.1, 5.2, 5.3 and 5.4. Also, we give all the preferences of Company1 and Company2 in Table 5.2. For CPU and RAM, both companies select clustering in level 1 because they have no special interest in these attributes, but they may have interest in



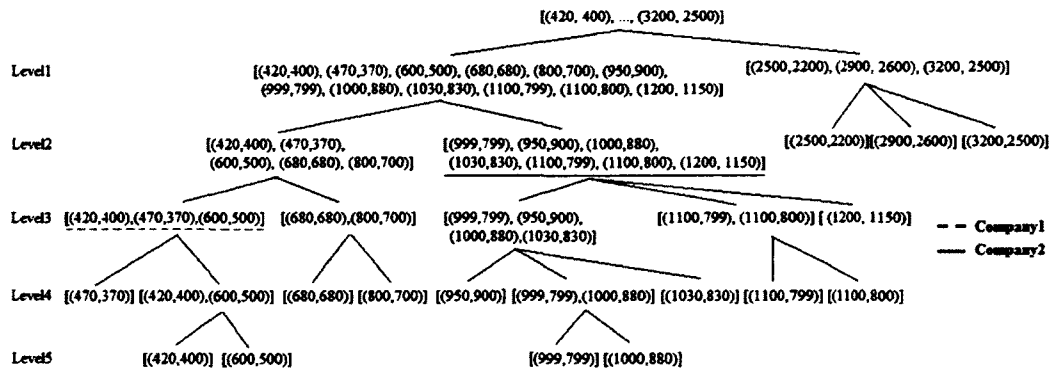


Figure 5.3: Price Data Clustering Tree

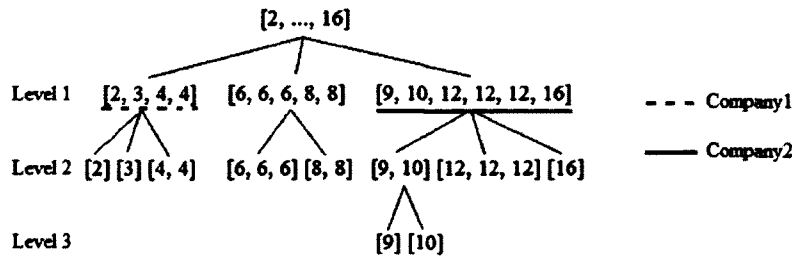


Figure 5.4: RAM Data Clustering Tree

different attribute value. Company1 may have an interest in lower CPU than Company 2 because of Company 1's selection. With the same reason, Company 2 may have more interest in higher RAM and Hard Drive. Company 1 focuses on the lower price computer and then higher performance. Company 2 is the opposite, which wants a higher performance with an acceptable price.

Table 5.2: Sections for both Companies

		Company1	Company2
<b>CPU</b>	Clustering	[2.2, ..., 2.8]	[(3,3), (3.2,3.2)]
	Level	1	1
<b>RAM</b>	Clustering	[2, ..., 4]	[9, ..., 16]
	Level	1	1
<b>Hard Drive</b>	Clustering	[820, ..., 1024]	[1200, ..., 1500]
	Level	2	3
<b>Price</b>	Clustering	[(420,400), (600,500)]	[(999,799), ..., (1200,1150)]
	Level	3	2

#### 5.1.4 Calculating Interest Weights

According to these attribute selections, interest weight coefficients for both companies are calculated with Formula 3.4. Using selections of Company1 as an example, the CPU interest weight coefficient is 1.1167, Price with a coefficient of 10.1521, RAM with 2.3333, and Hard Drive with 3.6078. According to Formula 3.5, we can get four attribute interest weights for Company1 as follows (as shown in Figure 4.7, “*Company\_ID = 1*”):

$$IW_{Company1\_HardDrive} = \frac{3.6078}{1.1167 + 10.1521 + 2.3333 + 3.6078} \approx 0.2096$$

$$IW_{Company1\_RAM} = \frac{2.3333}{1.1167 + 10.1521 + 2.3333 + 3.6078} \approx 0.1356$$

$$IW_{Company1\_CPU} = \frac{1.1167}{1.1167 + 10.1521 + 2.3333 + 3.6078} \approx 0.0649$$

$$IW_{Company1\_Price} = \frac{10.1521}{1.1167 + 10.1521 + 2.3333 + 3.6078} \approx 0.5899$$

Based on the interest weight calculation, we can see that Price is much more important than other offer attributes for Company1. Such interest features will lead our evaluation system to find Company1's best supplier.

### 5.1.5 Generating Interest Rate Functions

#### Hard Drive Interest Rates

Company1's interest rate function for Hard Dive is the same as the function in Example 6 because both clustering tree and selection are the same. In Table 5.3, we summarize the Hard Drive interest rate function for Company1, called  $S_{HardDrive\_Company1}()$ . Since LVQ of 1042.646 (calculated with  $SOM()$ ) is right outside of the selected clustering [820, 1024], we only use the left function.

Table 5.3:  $\varsigma_{HardDrive}()$  Generation for Company1

[AL, AR]	[L,R]	LVQ	Sign	$\alpha_{Left}$	$\varsigma_{HardDrive\_Left}()$
[160, 2000]	[820, 1024]	1042.6457	+1	0.0089	$\frac{1}{1+exp(0.0089 \cdot  x-1042.645 )}$
				$\alpha_{Right}$	$\varsigma_{HardDrive\_Right}()$
				N/A	N/A

Figure 5.5 shows Company1’s interest for Hard Drive increases with the rise of Hard Drive values. Based on this function, we can see that the Hard Drive data in the selected clustering, or called interest focus, doesn’t get a higher interest rate. This can be explained as: there exists a much higher attribute value (Offer15) than the interest focus data in the Hard Drive data set, and because of this data, the interests for the interest focus data are limited.

In Table 5.4, we build the Hard Drive interest function for Company2. Since LVQ of 1295.336 is inside [1200, 1500], we decompose  $\varsigma_{HardDrive\_Company2}()$  into two functions:  $\varsigma_{HardDrive\_Left}()$  and  $\varsigma_{HardDrive\_Right}()$ . In Figure 5.6, the function shows the offers that are less than 1200 GB are not interested by Company 2 at all. The clustering [1200, 1500] represents Company2’s most interested area thus the interest rates rise sharply in that area. Comparing with this “interest area”, Hard Drive of 2000 GB seems far away from this area. Thus, interest rate increases slowly between values 1500 GB to 2000 GB.

The interests of Hard Drive for both companies are obviously different. Company1 focus on some lower or middle range of Hard Drive data, while Company2

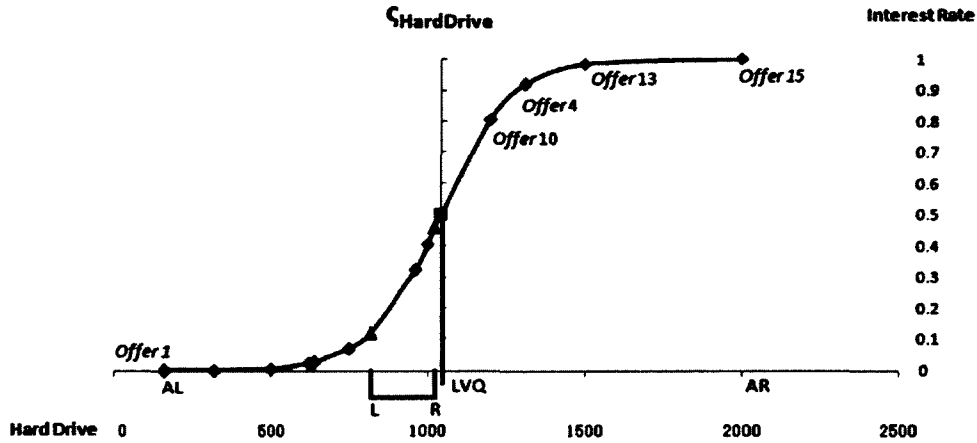


Figure 5.5: Company1 Interest Rate Function for Hard Drive Data

Table 5.4:  $\varsigma_{HardDrive}()$  Generation for Company2

[AL, AR]	[L,R]	LVQ	Sign	$\alpha_{Left}$	$\varsigma_{HardDrive\_Left}()$
[160, 2000]	[1200, 1500]	1295.336	+1	0.0021	$\frac{1}{1+exp(0.0021 \cdot  x-1295.336 )}$
				$\alpha_{Right}$	$\varsigma_{HardDrive\_Right}()$
				-0.0098	$\frac{1}{1+exp(-0.0098 \cdot  x-1295.336 )}$

is only interested in higher data values. Using Hard Drive 1310 GB in Offer4 as an example, for Company1 it is pretty good with a higher interest rate of 0.9798, but for Company2 it is an average choice with an interest rate of 0.5358.

### CPU Interest Rates

Company1's CPU interest rate function is the same function shown in Example 7, because of the same clustering tree and selection. It only has function  $\varsigma_{CPU\_Left}()$

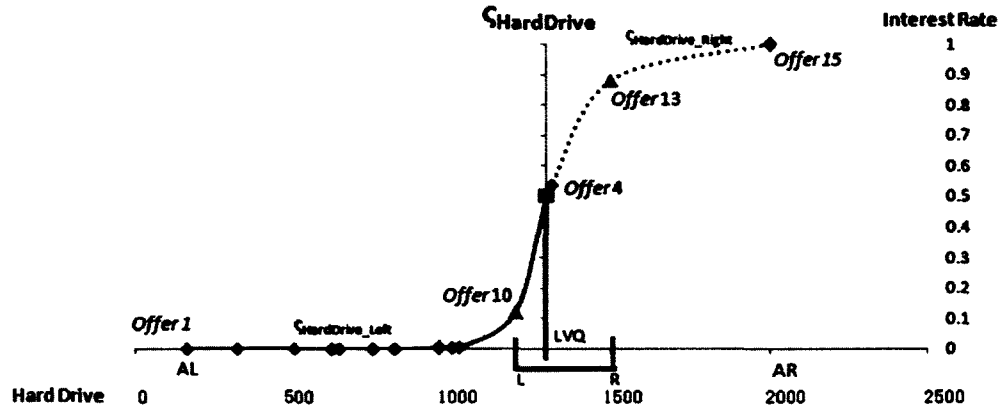


Figure 5.6: Company2 Interest Rate Function for Hard Drive Data

displayed in Figure 5.7.

In Figure 5.7, We can see there are four offers on the right side of LVQ, and their interest values are around 1: Offer15 (3.2, 3.2) with an interest rate of 0.9999, Offer13 (3, 3) with 0.9999, Offer12 (1.9, 1.9) with 0.9997 and Offer 14 (1.8, 1.8) with 0.9991. On the left side of LVQ, most offers are sorted in the range from 0.11 to 0.24. This is caused by the distribution of CPU data and the selected clustering. Many lower attribute values are close together, and stay at the left side of LVQ. On the other side, the larger data are far from them. This gap between the lower value and the higher value pushes these selected CPU values getting less interest rates.

According to Company2's selection, SOM() shows LVQ for the selected clustering [(3.0, 3.0), (3.2, 3.2)] is (2.8818, 2.9115). The selected clustering includes the best CPU attribute data (3.2, 3.2). Distances for LVQ, L and R to the best

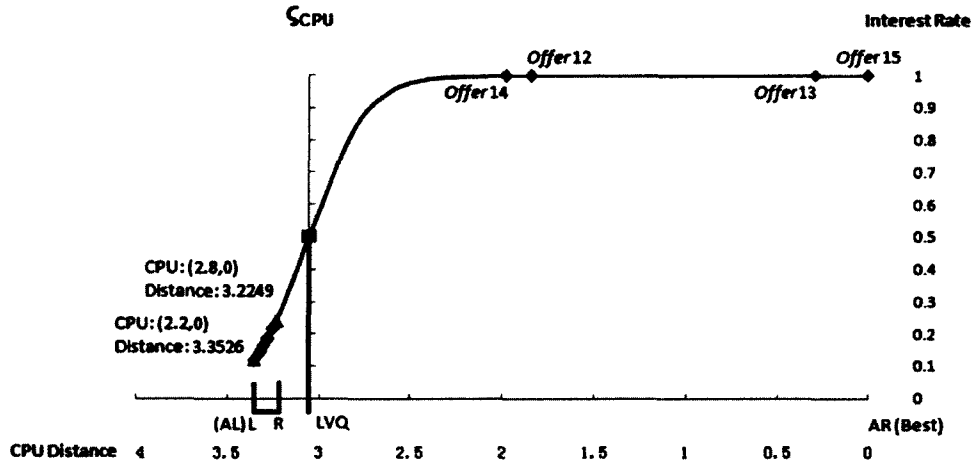


Figure 5.7: Company1 Interest Rate Function for CPU Data

attribute data are 0.4295, 0.2828 and 0. LVQ is left outside of the selected clustering distance range. Thus, the following generated CPU interest rate function (cf. Table 5.5) for Company2 only has function  $\zeta_{CPU\_Right}()$  (cf. Figure 5.8).

Table 5.5:  $\zeta_{CPU}()$  Generation for Company2

[AL, AR]	[L,R]	Distance(LVQ,BestAttributeData)	Sign	$\alpha_{Left}$	$\alpha_{Right}$
[2.2, (3.2,3.2)]	[(3,3), (3.2,3.2)]	0.4295	+1	N/A	-4.6564

$$\zeta_{CPU}(x) = \begin{cases} \zeta_{Left}(x) \text{ does not exist} \\ \zeta_{Right}(x) = \frac{1}{1 + \exp(\alpha_{Left} \cdot \text{Sign} \cdot |\text{Distance}(x, \text{BestAttributeData}) - \text{Distance}(LVQ, \text{BestAttributeData})|)} \end{cases}$$

$$= \frac{1}{1 + \exp^{-4.6564 \cdot |\text{Distance}[x, (3.2, 3.2)] - 0.4295|}} \quad x \in [2.2, \dots, (3.2, 3.2)]$$

$$\alpha_{Right} = \frac{-2}{|Distance(R, BestAttributeData) - Distance(LVQ, BestAttributeData)|}$$

$$= \frac{-2}{|0 - 0.4295|} = -4.6564$$

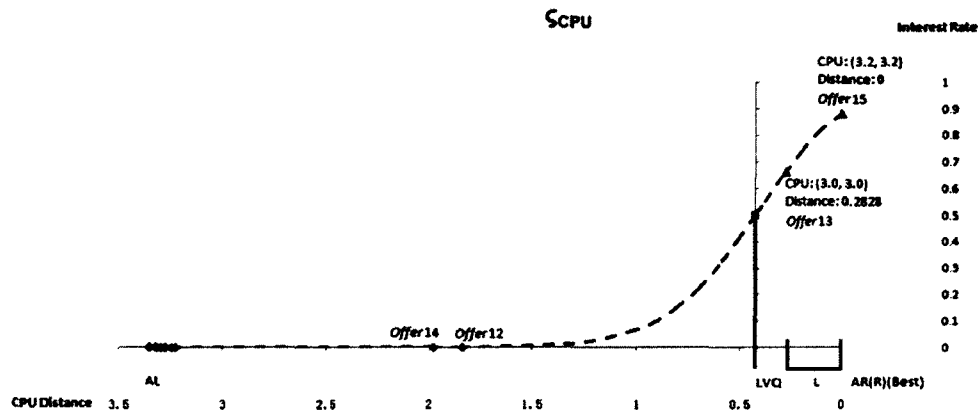


Figure 5.8: Company2 Interest Rate Function for CPU Data

In Figure 5.8, only two offers get higher interest rates: Offer15 (3.2, 3.2) with 0.8808 and Offer13 (3, 3) with 0.6644. The rest of offers have lower interest rates (less than 0.1), because they are far from the selected interest cluster.

Comparing these two interest rate functions, we find that most offers are acceptable for Company1; Offer12 and Offer14 can be considered as good offers. However, for Company2, only Offer15 and Offer13 are acceptable, and the rest of the offers including Offer12 and Offer14 are not.



## Price Interest Rates

The Sign of Price attribute is -1, which means the interest rate decreases with the increase of Price value. Also, Price (420, 400) is the best attribute data because it is the smallest value in this attribute. Therefore, based on Company1's selection, we can get the attribute distance values (cf. Table 5.6) to generate the Price interest rate function in Figure 5.9.

Table 5.6: Distances of Price Data for Company1

	$x$	$BestAttributeData$	$Distance(x, BestAttributeData)$
L	(420, 400)	(420, 400)	0
R	(600, 500)	(420, 400)	205.9126
LVQ	(539.9864, 586.4896)	(420, 400)	221.7546

$$\varsigma_{Price}(x) = \begin{cases} \varsigma_{Left}(x) = \frac{1}{1 + \exp(\alpha_{Left} \cdot Sign \cdot |Distance(x, BestAttributeData) - Distance(LVQ, BestAttributeData)|)} \\ \varsigma_{Right}(x) \text{ does not exist} \end{cases}$$

$$= \frac{1}{1 + \exp^{0.0090 \cdot (-1) \cdot |Distance[x, (420, 400)] - 221.7546|}} \quad x \in [(420, 400), \dots, (3200, 2500)]$$

$$\alpha_{Left} = \frac{2}{|Distance(L, BestAttributeData) - Distance(LVQ, BestAttributeData)|}$$

$$= \frac{2}{|0 - 221.7546|} = 0.0090$$

Company2 selects clustering [(999,799), ..., (1200,1150)] as its interest focus. According to the function  $SOM()$ , we can get the LVQ value (1163.661,

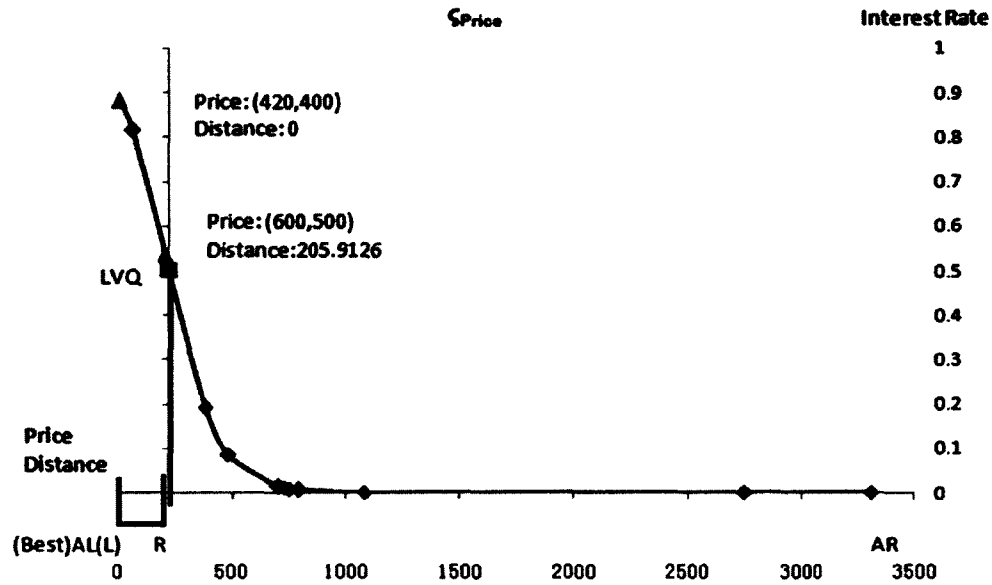


Figure 5.9: Company1 Interest Rate Function for Price Data

1008.061). Similarly, we can get Company2's interest rate function with the following distance calculation in Table 5.7.

Table 5.7: Distances of Price Data for Company2

	$x$	$BestAttributeData$	$Distance(x, BestAttributeData)$
L	(999, 799)	(420, 400)	703.1657
R	(1200, 1150)	(420, 400)	1082.081
LVQ	(1163.661, 1008.061)	(420, 400)	960.6087

$$\begin{aligned}
s_{Price}(x) &= \begin{cases} s_{Left}(x) = \frac{1}{1+\exp(\alpha_{Left} \cdot \text{Sign} \cdot |Distance(x, BestAttributeData) - Distance(LVQ, BestAttributeData)|)} \\ s_{Right}(x) = \frac{1}{1+\exp(\alpha_{Right} \cdot \text{Sign} \cdot |Distance(x, BestAttributeData) - Distance(LVQ, BestAttributeData)|)} \end{cases} \\
&= \begin{cases} s_{Left}(x) = \frac{1}{1+\exp^{0.0078 \cdot (-1) \cdot |Distance[x, (420, 400)] - 960.6087|}} \\ x \in [(420, 400), \dots, (1163.661, 1008.061)] \\ s_{Right}(x) = \frac{1}{1+\exp^{-0.0165 \cdot (-1) \cdot |Distance[x, (420, 400)] - 960.6087|}} \\ x \in [(1163.661, 1008.061), \dots, (3200, 2500)] \end{cases}
\end{aligned}$$

$$\begin{aligned}
\alpha_{Left} &= \frac{2}{|Distance(L, BestAttributeData) - Distance(LVQ, BestAttributeData)|} \\
&= \frac{2}{|703.1657 - 960.6087|} = 0.0078 \\
\alpha_{Right} &= \frac{-2}{|Distance(R, BestAttributeData) - Distance(LVQ, BestAttributeData)|} \\
&= \frac{-2}{|1082.081 - 960.6087|} = -0.0165
\end{aligned}$$

Comparing these two interest rate functions (cf. Figure 5.9 and 5.10), we can find the function for Company1 drops from the beginning of the function, while Company2's drops at the middle. It shows Company1's focus is on the very lower price and any cost increase can make Company1 lose some interest rates. However, Company2 can still accept a little higher price offer.

### RAM Interest Rates

We use Table 5.8 and 5.9 to show the interest rates of Company1 and Company2 on attribute RAM.

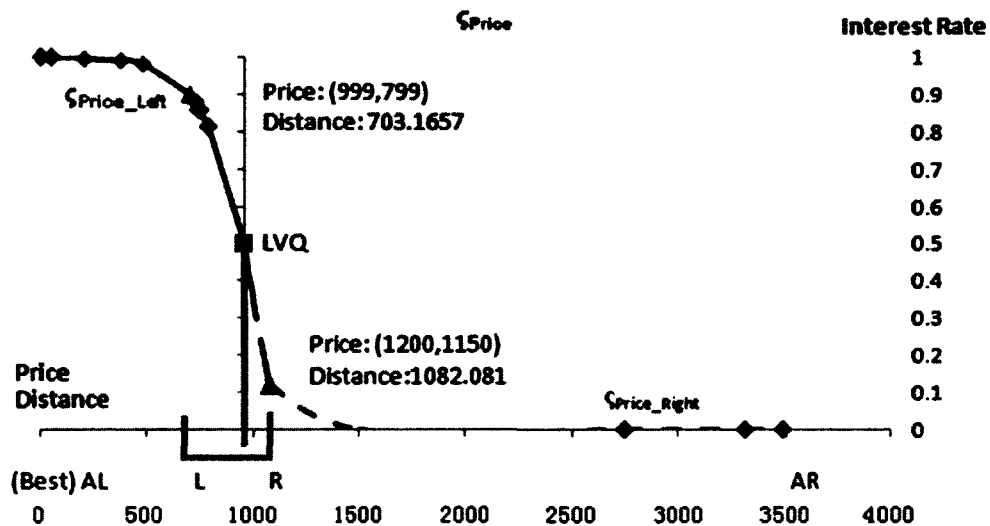


Figure 5.10: Company2 Interest Rate Function for Price Data

Table 5.8:  $\varsigma_{RAM}()$  Generation for Company1

[AL, AR]	[L,R]	LVQ	Sign	$\alpha_{Left}$	$\varsigma_{RAM\_Left}()$
[2, 16]	[2, 4]	3.7726	+1	1.1283	$\frac{1}{1+\exp(1.1283 \cdot  x-3.7726 )}$
				$\alpha_{Right}$	$\varsigma_{RAM\_Right}()$
				-8.7951	$\frac{1}{1+\exp(-8.7951 \cdot  x-3.7726 )}$

Table 5.9:  $\varsigma_{RAM}()$  Generation for Company2

[AL, AR]	[L,R]	LVQ	Sign	$\alpha_{Left}$	$\varsigma_{RAM\_Left}()$
[9,16]	[2, 16]	9.8894	+1	2.2487	$\frac{1}{1+\exp(2.2487 \cdot  x-9.8894 )}$
				$\alpha_{Right}$	$\varsigma_{RAM\_Right}()$
				-0.3273	$\frac{1}{1+\exp(-0.3273 \cdot  x-9.8894 )}$

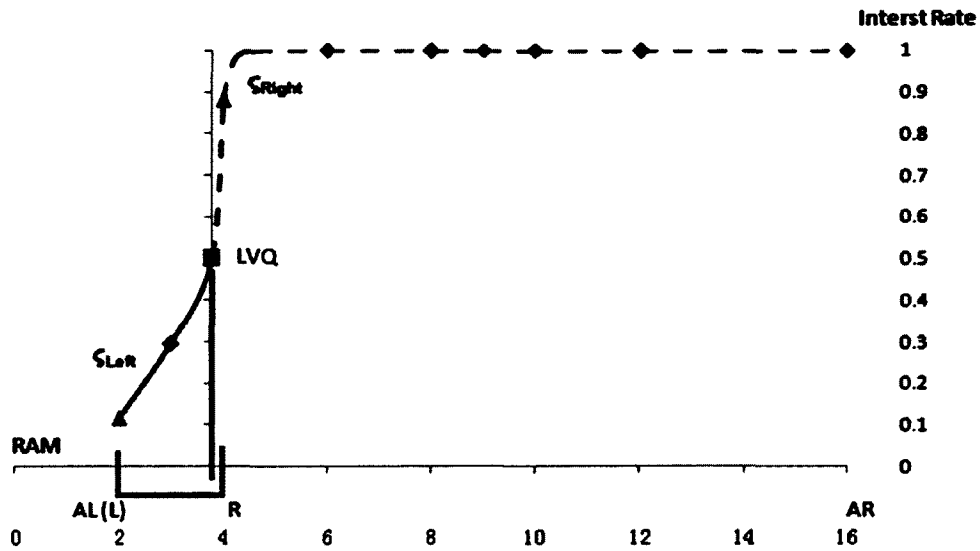


Figure 5.11: Company1 Interest Rate Function for RAM Data

Comparing these two RAM interest rate functions in Figure 5.11 and 5.12, we can see the Company2 has a strong interest of the highest GB RAM. Company2's interest rate drops quickly for the best three RAM attribute values, and decreases sharply for the rest attributes. Company1 prefers the lower RAM attributes.

### 5.1.6 Generating the Interest Models

After our system gets the interest weights and interest rate functions for the four attributes, it generates the interest model (IM) for the companies as follows.

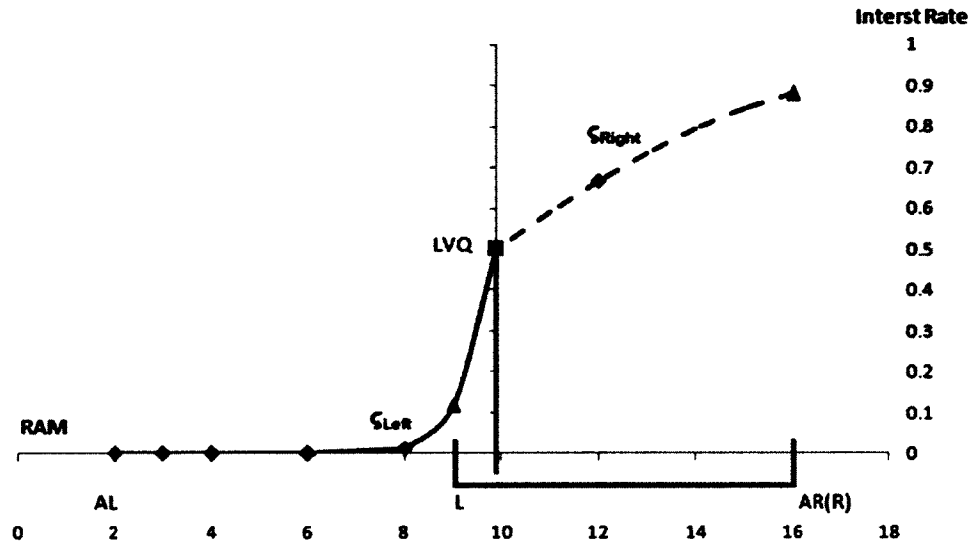


Figure 5.12: Company2 Interest Rate Function for RAM Data

$$\begin{aligned}
 IM_{Company1}(Offer) = & 0.0649 \cdot \varsigma_{CPU}(CPU) + 0.1356 \cdot \varsigma_{RAM}(RAM) \\
 & + 0.2096 \cdot \varsigma_{HardDrive}(HardDrive) + 0.5899 \cdot \varsigma_{Price}(Price)
 \end{aligned}$$

$$\begin{aligned}
 IM_{Company2}(Offer) = & 0.2349 \cdot \varsigma_{CPU}(CPU) + 0.0655 \cdot \varsigma_{RAM}(RAM) \\
 & + 0.3613 \cdot \varsigma_{HardDrive}(HardDrive) + 0.3383 \cdot \varsigma_{Price}(Price)
 \end{aligned}$$

For example, we show below how the interest model calculates the two companies' interest rates for Offer1:

$$\begin{aligned}
IM_{Company1}(Offer1) &= 0.0649 \cdot \varsigma_{CPU}(CPU) + 0.1356 \cdot \varsigma_{RAM}(RAM) \\
&\quad + 0.2096 \cdot \varsigma_{HardDrive}(HardDrive) + 0.5899 \cdot \varsigma_{Price}(Price) \\
&= 0.0649 \cdot 0.119202 + 0.1356 \cdot 0.119207 + 0.2096 \cdot 0.0015 \\
&\quad + 0.5899 \cdot 0.8808 \\
&= 0.5438
\end{aligned}$$

$$\begin{aligned}
IM_{Company2}(Offer1) &= 0.2349 \cdot \varsigma_{CPU}(CPU) + 0.0655 \cdot \varsigma_{RAM}(RAM) \\
&\quad + 0.3613 \cdot \varsigma_{HardDrive}(HardDrive) + 0.3383 \cdot \varsigma_{Price}(Price) \\
&= 0.2349 \cdot 3.93E - 07 + 0.0655 \cdot 1.97E - 08 + 0.3613 \cdot 1.3E - 09 \\
&\quad + 0.3383 \cdot 0.9997 \\
&= 0.3382
\end{aligned}$$

So, we have an interest rate of 0.5438 for Company1 and 0.3382 for Company2. Based on these values, we can conclude that Company1 has a higher interest than Company2 to co-operate with the supplier having Offer1. With our interest model, we can evaluate all the candidate offers of Table 5.1. Table 5.10 shows that Offer1 is the best offer for Company1, while Company2 may purchase Offer15 in Table 5.11.

From the evaluation results, we can see that Company1's results followed the order as the lower price first (Offer1, Offer2, Offer3) and then the higher performance (Offer15, Offer13, Offer4, etc). Company2's result is on the opposite

Table 5.10: Sorted Offers for Company1

Offer	ID	CPU	RAM	Hard Drive	Price	Interest Rate
*	1	2.2	2	320	420,400	0.5438
	2	2.2	3	640	470,370	0.5332
	3	2.5	4	500	600,500	0.4489
	15	(3.2,3.2)	12	2000	2100,2500	0.4101
	13	(3.0,3.0)	16	1500	2900,2600	0.4067
	4	2.33	8	1310	1100,800	0.3408
	12	(1.9,1.9)	10	1000	800,700	0.3361
	14	(1.8,1.8)	9	160	680,600	0.3130
	10	2.4	4	1200	950,900	0.3046
	7	2.66	12	1024	2500,2200	0.2456
	8	2.3	12	960	1000,880	0.2172
	9	2.5	6	820	1100,799	0.1760
	5	2.4	8	750	999,799	0.1678
	6	2.5	6	750	1030,830	0.1668
	11	2.8	6	620	1200,1150	0.1560

\*: The Best Offer

order. These results are consistent with our interests setting for Company1 and Company2 in section 5.1.3.



Table 5.11: Sorted Offers for Company2

Offer	ID	CPU	RAM	Hard Drive	Price	Interest Rate
*	15	(3.2,3.2)	12	2000	2100,2500	0.6553
	13	(3.0,3.0)	16	1500	2900,2600	0.5852
	4	2.33	8	1310	1100,800	0.5292
	12	(1.9,1.9)	10	1000	800,700	0.4488
	14	(1.8,1.8)	9	160	680,600	0.4302
	2	2.2	3	640	470,370	0.4254
	1	2.2	2	320	420,400	0.4250
	3	2.5	4	500	600,500	0.4248
	10	2.4	4	1200	950,900	0.4219
	8	2.3	12	960	1000,880	0.4183
	5	2.4	8	750	999,799	0.3924
	6	2.5	6	750	1030,830	0.3804
	9	2.5	6	820	1100,799	0.3672
	7	2.66	12	1024	2500,2200	0.1796
	11	2.8	6	620	1200,1150	0.1775

\*: The Best Offer

## Chapter 6

# CONCLUSION AND FUTURE WORK

### 6.1 Thesis Contributions

In this thesis, we successfully created an interest based evaluation system to make up the limitation of current matchmaking systems. Indeed, matchmakers do not recognize the difference between buyers' interests and tastes. Unlike any other existing offer evaluation in the matchmaking systems, an individual best offer can be found thanks to our system. Our system can provide a personalized best offer which is better than returning the same believed best offer to several buyers who submit the same query.

We also showed the benefits of sorting the query-matched offers according to the buyer's interests and needs. First, the ranking criteria can be defined by the buyers not by the system. The buyer interacts with our system to build an

interest model which is an individual ranking criterion. It is better to take into account the buyer's interest changes on multiple offer attributes rather than the system own ranking criteria. Thus, our system recommends a best offer which is the closet to the buyer's specific needs. Furthermore, our evaluation method avoids the linear matching problems. Since our interest model is a non-linear function, it can provide more accurate results than other linear functions. Our system can evaluate more complex attribute data while most other evaluation functions only process simple data.

## 6.2 Future Work

There are several possible directions of our work. The first one is to assess the feasibility of our system on a more relevant data set. Currently, we are testing our model with a database of 210 users' selections of several types of transportation. Our goal is to compare the accuracy of our interest model with the MNL model.

The second direction is to include the interest learning [28] in our offer evaluation system. The main purpose of this learning is to update the interest model to fit the buyer's interests instantly. A learned interest model will be able to determine the best offer in these two following situations: the buyer shifts his interests, or new matched offers are added in our system database.

The third future direction is to extend the interest model to rank web services. Web service offers are more complex than value-based offer attributes. We can still use a similar evaluation method to rank the semantic atoms based on

individual interests.

Last but not least, we are also interested in developing a personalized attribute clustering trees specifically for each buyer. Buyers may have different knowledge, experience, attitude, and other factors which make them have different clustering trees than any other buyers. Our system will be able to better understand each buyer's interests with these features.

# Bibliography

- [1] D. Kuokka, and L. Harada. Matchmaking for information agents Readins in Agents. ed. Michael N. Huhns and Munindar P. Singh, Morgan Kaufmann, pp. 672-678, 1995.
- [2] S. H. Ha, and S. C. Park. Matching buyers and suppliers: an intelligent dynamic exchange model. Intelligent Systems, IEEE, vol. 16, no. 4, pp. 28-40, Jul-Aug 2001.
- [3] T. Kawamura, T. Hasegawa, A. Ohsuga, M. Paolucci, and K. Sycara. Web services lookup: a matchmaker experiment. IT Professional, IEEE, vol. 7, no. 2, pp. 36-41, Mar-Apr 2005.
- [4] T. Qiu, and P. F. Li. Web Service Discovery Based on Semantic Matchmaking with UDDI. 9th International Conference for Young Computer Scientists, pp. 1229-1234, Nov. 2008.
- [5] B. Dong-wei, F. Ai-guo, C. Shan-fa. Semantic Matchmaking of Web Services Constraint Conditions. 5th Int. Conference on Wireless Communications, Networking and Mobile Computing, IEEE, pp. 1-5, Sep. 2009.
- [6] T.W. Cheng, W.L. Wang, and A. P. Chen. E-marketplace using artificial immune system as matchmaker. International Conference on e-Commerce Technology, IEEE, pp. 358-361, July 2004.

- [7] W. Guo. Ontology Design for Supporting Matchmaking in E-commerce. International Symposium on Information Science and Engineering, pp. 410-413, Dec. 2008.
- [8] B. Dong-wei, L. Chuan-Chang, P. Yong, C. Jun-liang. Web Services Matchmaking with Incremental Semantic Precision. International Conference on Wireless Communications, Networking and Mobile Computing, IEEE, pp. 1-4, Sept. 2006.
- [9] H. Wang, and Z. Z. Li. A Semantic Matchmaking Method of Web Services Based on SHOIN  $\hat{+}$  (D)\*. Asia-Pacific Services Computing Conference, IEEE, pp. 26-33, Dec. 2006.
- [10] C. Hahn, S. Nesbigall, S. Warwas, I. Zinnikus, M. Klusch, and K. Fischer. Model-Driven Approach to the Integration of Multi-Agent Systems and Semantic Web Services. 12th Enterprise Distributed Object Computing Conference Workshops, IEEE, pp. 314-324, Sep. 2008.
- [11] R. Huang, Y. W. Zhuang, J. L. Zhou, and Q. Y. Cao. Semantic Web-based Context-aware Service Selection in Task-computing. International Workshop on Modelling, Simulation and Optimization, pp. 97-101, Dec. 2008.
- [12] U. Bellur, and H. Vadodaria. Web Service Ranking Using Semantic Profile Information. International Conference on Web Services, IEEE, pp. 872-879, July 2009.

- [13] D. Essex. Matchmaker, matchmaker. *Communications of the ACM*, ACM, vol. 52, no. 5, pp. 16-17, May 2009.
- [14] S. Sadaoui, and W. Jiang. An Offer Evaluation System based on Buyer's Interests. In *Proceedings of 26th Annual ACM Symposium on Applied Computing*, Mar. 2011.
- [15] D. McFadden. Economic Choices. *American Economic Review*, American Economic Association, vol. 91, no. 3, pp. 351-378, Jun. 2001.
- [16] D. McFadden. Conditional Logit Analysis of Qualitative Choice Behavior. *Frontiers in Econometrics*. ed. P. Zarembka, Academic Press: New York, pp. 105-142, 1974.
- [17] C.C. Xu, W. Wang, Z.B. Li, C. Yang. Comparative study on drivers' route choice response to travel information at different departure time, 2nd International Asia Conference on Informatics in Control, Automation and Robotics, IEEE, vol. 3, pp. 97-100, Mar. 2010.
- [18] T. Kohonen. *The Self-Organizing Map*, 3rd Edition, Springer -Verlag New York Inc., 2001.
- [19] Y. Y. Chen, and K. Y. Young. Applying SOM as a Search Mechanism for Dynamic System. 44th IEEE Conference on Decision and Control and European Control Conference, IEEE, pp. 4111- 4116, Dec. 2005.
- [20] X. Shen, X. Jin, R. F. Bie, and Y. C. Sun. MSC: A Semantic Ranking for Hitting Results of Matchmaking of Services. 30th Annual International

- Computer Software and Applications Conference, IEEE, pp. 291-296, Dec. 2006.
- [21] H. Q. Yu, and S. Reiff-Marganiec. Non-Functional Property based Service Selection: A Survey and Classification of Approaches. Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop, ECOWS 2008, IEEE, Nov. 2008.
- [22] Y. Liu, A. H. Ngu, and L. Z. Zeng. QoS computation and policing in dynamic web service selection. In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters. ACM, pp. 66-73, May 2004.
- [23] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A qos-aware selection model for semantic web services. 4th International Conference on Service-Oriented Computing, pp. 390-401, 2006.
- [24] J.W. Yang, D. Sun, Y.P. Du, Y.Y. Zhao. The effect of sampling of alternatives on MNL models: An empirical analysis in the context of shopping-destination choice models. 2nd IEEE International Conference on Computer Science and Information Technology, pp. 75-80, Aug. 2009.
- [25] Y. Chabeb, S. Tata, A. Ozanne. YASA-M: A Semantic Web Service Matchmaker, International Conference on Advanced Information Networking and Applications, IEEE, pp. 966-973, April 2010.



- [26] M. Klusch, P. Kapahnke, F. Kaufer. Evaluation of WSML Service Retrieval with WSMO-MX, International Conference on Web Services, IEEE, pp. 401-408, Sep. 2008.
  
- [27] C. W. Lin, J. S. Wang. A digital circuit design of hyperbolic tangent sigmoid function for neural networks, International Symposium on Circuits and Systems, ISCAS, IEEE, pp.856-859, May 2008.
  
- [28] Y.Z. Wei, L. Moreau, N.R. Jennings. Learning users' interests by quality classification in market-based recommender systems. Transactions on Knowledge and Data Engineering, IEEE, vol. 17, no. 12, pp. 1678- 1688, Dec. 2005.

## Appendix A: SOM Implementation

```
/* Related class member variables */
private double[] A1LVQ;           // LVQ1
private double[] A2LVQ;           // LVQ2
private double[] A3LVQ;           // LVQ2
private ArrayList A1_offer_id = new ArrayList(); // A1
private ArrayList A2_offer_id = new ArrayList(); // A2
private ArrayList A3_offer_id = new ArrayList(); // A3
private void SOM( DataTable attribute_table , int dimension , DataTable
    root_attribute_table , int level )
{ /* Initialize Data Values */
    double LVQ1_distance =0;
    double LVQ2_distance =0;
    double LVQ3_distance =0;
    int learn_time =1;
    /* Create Random LVQ */
    this.CreateLVQ(dimension , attribute_table);
    /* Learning Loop */
    while (this.GetLearningRate(learn_time) > 0.0001)
    //Check if it need learning again
    { // Pick up value x
        for (int i=0; i<attribute_table.Rows.Count; i++){
            for (int j=1; j<=dimension; j++){
                // Calculating distance between x to LVQ1 through the loop
                LVQ1_distance += Math.Pow(
                    (System.Convert.ToDouble(attribute_table.Rows[i][j])
                    - this.A1LVQ[j-1] ), 2.0 );
                // Calculating distance between x to LVQ2 through the loop
                LVQ2_distance += Math.Pow(
                    (System.Convert.ToDouble(attribute_table.Rows[i][j])
                    -this.A2LVQ[j-1] ), 2.0 );
                // Calculating distance between x to LVQ3 through the loop
                LVQ3_distance += Math.Pow(
                    (System.Convert.ToDouble(attribute_table.Rows[i][j])
                    - this.A3LVQ[j-1] ), 2.0 );
            }
        }
    }
}
```

```

    }
    // Get each LVQ distance to x
    LVQ1.distance = Math.Round( Math.Sqrt(LVQ1.distance), 2 );
    LVQ2.distance = Math.Round( Math.Sqrt(LVQ2.distance), 2 );
    LVQ3.distance = Math.Round( Math.Sqrt(LVQ3.distance), 2 );
    // Find the closest LVQ to x
    this.FindWinnerLVQ( LVQ1.distance, LVQ2.distance, LVQ3.distance,
        System.Convert.ToInt32( attribute_table.Rows[i][0] ) // ID of x
    );
    // Update LVQi
    this.updateLVQ( i, dimension, attribute_table, LVQ1.distance,
        LVQ2.distance, LVQ3.distance,
        this.GetLearningRate(learn_time),
        root_attribute_table );
    // Clear LVQ distance for next input x
    LVQ1.distance = 0;
    LVQ2.distance = 0;
    LVQ3.distance = 0;
}
learn_time++; // Learning time update
// Prepare for next learning
if (this.GetLearningRate(learn_time) > 0.01){
    A1_offer_id = new ArrayList();
    A2_offer_id = new ArrayList();
    A3_offer_id = new ArrayList();
}
}

/*
 * SOM sub-function
 * Find the closest LVQc to x
 * Put x into LVQc related cluster
 */
private void FindWinnerLVQ( double distance1, double distance2,

```

```

                                double distance3, int current_row )
{
    int winner = 0;
    /* Find the closest LVQ based on distance */
    if (distance1 <= distance2)
    {
        if (distance1 <= distance3) { winner = 1; }
        else { winner = 3; }
    }
    else
    {
        if (distance2 <= distance3) { winner = 2; }
        else { winner = 3; }
    }
    /* Put x in Ac
    * The Ac could be temporary.
    * It depends on when the learning loop stop.
    * The last time generated Ac is the final result
    */
    if (winner == 1) { this.A1_offer_id.Add(current_row); }
    if (winner == 2) { this.A2_offer_id.Add(current_row); }
    if (winner == 3) { this.A3_offer_id.Add(current_row); }
}

```