1-1-2010

A Cluster-Based Browsing Model For QoS-Aware Web Service Selection

Kian Farsandaj
Ryerson University

A CLUSTER-BASED BROWSING MODEL FOR QOS-AWARE WEB SERVICE SELECTION

by

Kian Farsandaj

B.Sc., Applied Mathematics, Azad University of Tehran

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in

the Program of Computer Science

Toronto, Ontario, Canada, 2010

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis.

This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.
I understand that my thesis may be made electronically available to the public.
I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Kian Farsandaj

Signature …….. Date …….

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Kian Farsandaj

Signature …….. Date …….

CLUSTER-BASED BROWSING MODEL FOR QOS-AWARE WEB SERVICE SELECTION

Kian Farsandaj

Department of Computer Science, 2010
Ryerson University

ABSTRACT

In the last decade, selecting suitable web services based on users' requirements has become one of the major subjects in the web service domain. Many research works have been done – either based on functional requirements, or focusing more on Quality of Service (QoS)-based selection. We believe that searching is not the only way to implement the selection. Selection could also be done by browsing, or by a combination of searching and browsing. In this thesis, we propose a browsing method based on the Scatter/Gather model, which helps users gain a better understanding of the QoS value distribution of the web services and locate their desired services. Because the Scatter/Gather model uses cluster analysis techniques and web service QoS data is best represented as a vector of intervals, or more generically a vector of symbolic data, we apply a symbolic clustering algorithm and implement different variations of the Scatter/Gather model. Through our experiments on both synthetic and real datasets, we identify the most efficient (based on the processing time) and effective implementations.

ACKNOWLEDGEMENTS

It is my pleasure to thank those who made this thesis possible. This thesis would not have been possible without my supervisor's supports. I would like to show my gratitude to Dr. Cherie Ding, who gave me the opportunity to work with her and do my research in a very interesting area with her supports and guidance. She constantly helped me solve various problems I encountered during the course of my graduate study, and she is always patient and shows great concern on my situation.

I would like to express my deepest appreciation to Dr. Alireza Sadeghian, who has always helped me with his wise advices in different circumstances, made available his support in a number of ways, and with his kindly attitudes encouraged me in different parts of my study.

It is an honor for me to thank my committee members Dr. Alireza Sadeghian, Dr. Isaac Woungang, and Dr. Alex Ferworn for their valuable time, and their every appreciated support during my education.

I would like to thank Ryerson University, Department of Computer Science, and all professors and staffs, especially Dr. Santos, Dr. Abhari, Maria Landau, and others for giving me the chance of continuing my education in my very interested area, supporting me in every part of my study, and helping me throughout the whole period of my study here.

I owe my deepest gratitude to my late father, who was always an encouragement and a powerful help angel for me in every part of my life, my kindest mother who provided me a safest environment with her extreme supports, and my patient faithful wife Marjan who tolerated everything with love and kindness, and was a precious motivation in my whole life.

I am indebted to many of my friends – Preethy Sabamoorthy, Shilpi Verma, and Keyvan Tirdad for their supports anytime and anywhere they could. And I am grateful to Dr. Mehrdad Tirandazian as my special friend who always gave me valuable advices, and supported me in every period of my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

SOA	Service Oriented Architecture
UDDI	Universal Description, Definition, and Integration
WSDL	Web Service Description Language
SOAP	Simple Object Access Protocol
QoS	Quality of Service
SLA	Service Level Agreements
SCLUST (ISC)	Symbolic Dynamic Clustering Algorithm
QML	Quality of Service Modeling Language
WSML	Web Service Management Language
WSLA	Web Service Level Agreement Language
WSOL	Web Service Offer Language
NLA	New LAIR2 (H*k*-Means)
S/G	The original Scatter/Gather technique
ISC2	The improved ISC algorithm
EU	Euclidean
CB	City-block
HD	Hausdorff

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, Service Oriented Architecture (SOA) has made an influence which changes the way the enterprise software is developed and deployed, and become an opportunity for enterprises to more easily keep up with the rapidly changing market conditions, and conduct transactions with their business partners. SOA offers some major benefits in scalability, reusability, loose coupling, and platform independence. It allows enterprises to focus more on business processes as well as the application itself instead of the pure software development. Web service architecture is one special implementation of SOA. Web service is described as a self-contained software system which could be loosely coupled with other services to assemble a complex business application [1, 2].

Web services as a new standard technology consist of three main components: **service descriptions** which contain interface definitions, **mechanisms** to access or consume services by invoking their interfaces, and **the implementations of the services** which are the code behind the interfaces. A set of standards and protocols are widely used for web services, which enables them to communicate across different platforms and different languages, namely: Web Service Description Language (WSDL); Universal Description, Definition, and Integration (UDDI); and Simple Object Access Protocol (SOAP) [1].

According to the said protocols, the typical architecture of web services is constructed based on three entities: service provider, service requester, and service registry, and they interact with each other in the "publish, find, and bind" process. In this process, a service provider provides access to a web service by creating and publishing it in a registry such as a UDDI registry, which is responsible for maintaining the functional description of web services [3, 4, 5]. Later, a service requester searches through the registry to find the desired web service which meets the requirements, and then continues to bind with the selected service through the defined interface (Figure 1).

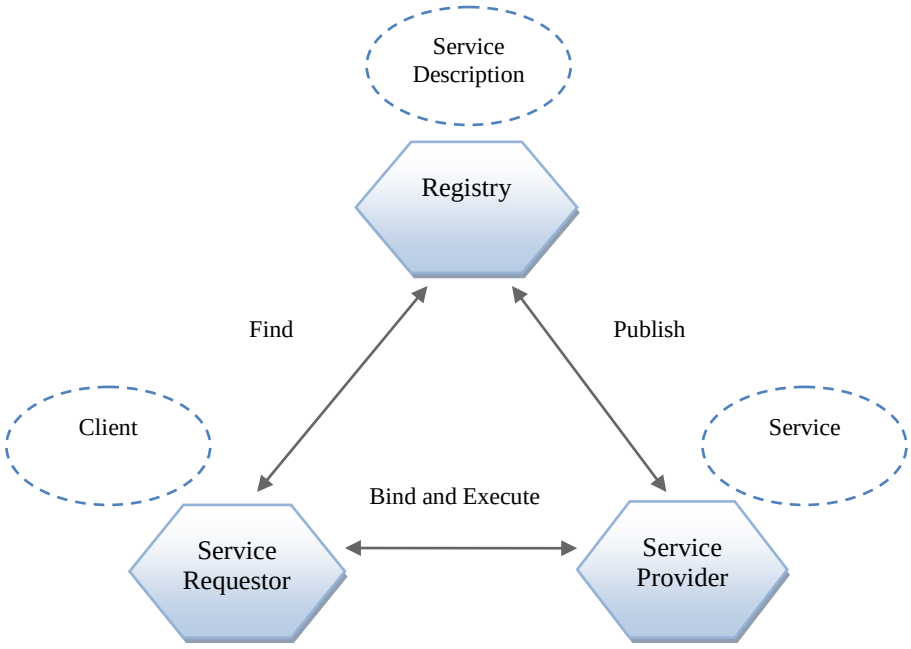


Figure 1- Web Service Architecture [1, 3, 4]

From the above discussion, we could see that the service discovery and selection component in a service registry is a key element of the architecture which leads service requesters to find and integrate the desired services offered by providers [6]. In this thesis, we study how to improve this selection process.

1.2 Motivation and Objectives

Generally speaking, in order to describe or understand a service, there are two key components, referred to as its functional, and its non-functional properties. Usually, the former can be inferred from the name and documentation of the service, the input/output parameters and other behaviour information, whereas the latter mainly includes the quality properties such as performance or security related ones, as well as other properties such as the cost, reputation, etc. In this thesis, to simplify the discussion, we use the term Quality of Service (QoS) to represent all non-functional properties [7, 8].

When more and more services are published online, there could be multiple services implementing the same function, therefore, the non-functional properties will be crucial to differentiate those services. However, the current UDDI registry was not designed to support non-functional attributes of web services, and hence, it limits the service selection to only functional requirements [5].

Considering that the QoS data are collected from the Service Level Agreements (SLA) or real-time monitoring engines, there could be multiple invocation instances and multiple versions of SLAs on the same service, which result in a large amount of QoS data. There could be a big challenge about how to organize, browse, search or analyze the QoS data of web services. There have been many research works on how to expand the current architecture model to support the QoS aware service selection. However, few of them considered the problem of organizing and navigating the QoS data. Therefore, the above-mentioned described the motivation of our research.

There are situations where QoS browsing is very helpful. First, it could give service requesters some idea about the QoS value distribution of the available services. Oftentimes, in order to search a service, requesters need to formulate a QoS query and submit it to a service selection system. If the query is not accurate, the returning results may not be accurate. For instance, a service requester wants to find a service with a high reliability level, and thus the request is stated as "reliability > 99%"; however, none of the services in the registry achieves this level, and the maximum reliability is 97%. In this case, no matching result could be returned although the requester can accept a service with reliability 97% as long as it is the highest available one [7]. The problem here is an inappropriate selection of QoS values in the query. If the requester could browse the QoS data before submitting the query, the problem could be solved.

Secondly, browsing could help service providers understand the actual demand from requesters. For instance, a service provider wants to publish a newly developed service, and in order to attract more users, the guaranteed quality level specified in the service description is very high based on the assumption that the higher quality level is always preferred. Although it does attract some users, the volume of invocation is not high enough to balance the investment on resources for hosting and delivering high quality services. The problem here is that the provider does not understand the real requirement from the users. For this particular service, maybe most of the requesters do not have a demand for a high quality level. Instead, the lower cost is always preferred. If the provider could browse the actual QoS data of services with similar functionality, the decision on service quality levels could be adjusted accordingly.

The purpose of this thesis is to investigate some effective ways of browsing the service QoS data. The knowledge gained from the browsing process could help the search and analysis process.

Scatter/Gather [9] is a well-known and well-studied browsing model on large document collections. The main idea is that the system scatters the given set of documents into a small number of clusters and presents summaries to users; based on their interests, the users choose one or more clusters, then the system gathers documents from the chosen clusters and scatters them into a few clusters again, and the whole process gets iterated, starting from the complete initial collection, and gradually narrowing down to user-desired documents. The process mainly uses the clustering algorithms for its implementation. We believe that this model can also be used to browse the QoS data. It could help users (either providers or requestors) to have a clear view of the QoS distribution of the service set. The number of iterations is decided by the users' granularity requirements.

There are a few main differences between the document collection and the set of services. Firstly, both the document and the service QoS data can be represented as a vector. The size of the document vector is usually the number of terms in the whole vocabulary, whereas the size of the QoS vector is the number of QoS attributes that the system can support. The former value is usually much bigger than the latter value. Secondly, the size of the service set under study is usually much smaller than the size of the document collection because the number of online documents is far more than that of the published services. This situation will probably remain the same even in the foreseen future, due to the higher complexity of developing services and the higher cost of hosting, delivering, or using services. Thirdly, for a QoS vector, each dimension has a different meaning because it represents different QoS attributes, a different data type [10]

(e.g. single numerical, interval, Boolean, categorical, etc.) and a different value range (e.g. reliability is from 0 to 100%, the authentication is either 0 or 1, the response time is from 0 to 2 seconds, etc.), whereas the dimension in a document vector has the same meaning, same data type and same value range which usually represents the term-frequency-decided weight. Fourthly, similarity-based measurements are more commonly used for document clustering and it is also believed to have a better performance [11]. However, for QoS clustering, since it is more appropriate to represent QoS data as symbolic data, the distance-based measurements are more common. Because of all these differences, it is necessary to re-study the Scatter/Gather model in this new context [7].

In this thesis, based on the features of the QoS data, we propose to use the Scatter/Gather as our browsing model, and a symbolic data clustering algorithm as its clustering component. We have four different implementations of the browsing model based on two originals, and two improved algorithms. The original algorithms consist of the original Scatter/Gather and the iterative Symbolic Dynamic Clustering Algorithm (SCLUST) [12]; furthermore, our improved algorithms include the improved SCLUST and the H*k*-means (NLA) algorithm [13], which is implemented based on the improvement on the LAIR2 algorithm [14]. For the improved techniques, we intended to increase the accuracy, as well as the efficiency (based on the processing time) of our clustering results. Therefore, their performances are tested and compared using some synthetic QoS datasets. Based on our experimental results, the Scatter/Gather model is very effective on QoS browsing, and different implementations achieve different levels of performance. To the best of our knowledge, applying the Scatter/Gather model to QoS browsing and searching is a novel idea, and using this model to browse the symbolic data set is also a new idea.

1.3 Thesis Structure

The rest of the thesis is structured as follows:

In Chapter 2, the background information and related works are discussed. Firstly, we describe different QoS properties of web services, including classification and taxonomy of QoS properties in different perspectives. Secondly, we review research works related to QoS-aware web service discovery and selection, with different frameworks. Thirdly, cluster analysis techniques are described by reviewing partitioning and hierarchical data clustering methods and symbolic data clustering. Finally, the Scatter/Gather framework, which is the backbone of our research, is reviewed.

In Chapter 3, all features of the Scatter/Gather model, which were proposed using different algorithms, are described. Subsequently, after discussing different distance functions, which are used to calculate distances between two vectors of intervals (which represent QoS attributes of web services), we describe the concept of the cluster homogeneity, and our proposed framework consisting of four techniques (the original Scatter/Gather, the iterative dynamic symbolic clustering and its improved version, and the improved LAIR2 algorithm).

In Chapter 4, we explain our experimental design, the features of the datasets we used in our experiments. We also present the results of executing our framework on 10 different synthetic datasets and a real dataset. We then continue with results analysis, comparison of different approaches, and further discussion on the implications from these results.

Finally, in Chapter 5, we summarize our work and the conclusions we draw from the experiments. Furthermore, we describe the future works that can be done on QoS-aware web service selection.

CHAPTER 2

LITERATURE REVIEW

The scope of this chapter is to give more detailed information on: i) different QoS properties, ii) different QoS-based service discovery and selection models, iii) overview of data clustering algorithms, different data types such as interval data or symbolic data and symbolic clustering algorithms, iv) the Scatter/Gather model.

2.1 Non-Functional Properties and Quality of Services

As previously mentioned in the Chapter 1, web service requirements are categorized into two types: functional requirements, and non-functional requirements. Functional requirements of services describe the functionality and behaviour of the service which can be described as tasks, activities, users' goals, and in general what the system is expected to do. On the other hand, non-functional requirements contain qualities and characteristics of services, which can affect the users' satisfaction level with a specific web service. We use the term "QoS" to represent all non-functional properties [15, 16].

Generally, QoS attributes can be classified into different groups based on different perspectives. For example, in [17], QoS is categorized into metrics (which describe quantifiable parameters) and policies. Metrics are further divided into performance specifications, security levels, and relative importance levels. Moreover, policies are divided into management policies, and level of service. In [18], QoS is classified into technical qualities and managerial qualities.

There are some general and common attributes (such as price) which are independent of the domain the specific web service belongs to. The domain-independent attributes can also be categorized into different groups. For example, in [8] these attributes have been categorized into four groups as follows:

- Performance: Processing Time/Execution Time, Latency, Throughput, and Response Time;
- Dependability: Availability, Accessibility, Accuracy, Reliability, Capacity, Integrity, Stability/Exception Handling, Robustness/Flexibility, Regulatory/Interoperability, and Scalability;
- Security: Authentication, Authorization, Non-Repudiation, Integrity, Encryption, Traceability/Auditability, and Confidentiality/Privacy;
- Application-specific metrics.

In the above taxonomy, cost is not considered as a metric or a QoS attribute. However, some of papers do include it as a QoS metric. For example, the QoS attributes of web services are classified into the following groups in [5]:

- Runtime Related attributes such as: scalability, capacity, reliability, robustness/flexibility, exception handling, accuracy, and performance (which can be sub-classified into response time, latency, throughput, execution time, and transaction time);
- Transaction Support Related attributes such as: integrity, isolation, durability, atomicity, and consistency;

- Configuration Management and Cost Related attributes such as: regulatory, cost, reputation, completeness, supported standard, stability/change, and guaranteed messaging requirements;
- Security Related attributes such as: authentication, authorization, traceability/audit-ability, confidentiality, accountability, data encryption, and non-repudiation.

Network Related QoS is added to the above classification in [19], which includes network delay, delay variation, and packet loss, to name a few.

The reason of applying different taxonomies and classifications on quality aspects of web services on different researches is to make QoS metrics well organized and have a simpler view for further analysis, so as to be able to use them in different scenarios such as web service selection [8].

2.2 QoS-Based Web Service Selection

Current web service technology based on the UDDI model limits the service discovery and selection to functional requirements only, which causes the problem of selecting services with same functionalities but different qualities [20, 7]. In this regard, QoS-aware methods considering non-functional attributes of web services have been proposed to resolve the weaknesses found in the UDDI keyword-based search [21]. The examples of such solutions include: (1) extending UDDI with the consideration of QoS information embedded into the tModel, (2) defining a QoS repository as a QoS broker to maintain and interact with QoS information [5, 16, 22, 23].

Furthermore, in order to monitor or extract the QoS data from different sources such as SLAs or users' feedbacks, third parties such as different agents can be involved [24, 25]. For instance, in [26], QoS negotiation and web service selection are implemented with a multi-agent computational paradigm; here, the implemented agents by service provider can negotiate with the agent implemented by the requester for the SLA configuration in the service selection process. In this regard, several languages which use abstract syntaxes or HTML language have been proposed to express QoS information of web services such as Quality of Service Modeling Language (QML) [27], Web Service Management Language (WSML) [28], Web Service Level Agreement Language (WSLA) [29], and Web Service Offer Language (WSOL) [30].

In the following, we study different proposed frameworks or publish and selection models which tried to overcome the above mentioned issues by taking into account the QoS requirements of web services.

In [5] a new regulated model for web service discovery, based on current publish-find-bind model (Figure. 1) is proposed. In order to overcome the inability of supporting the QoS requirements from the UDDI registry, and to improve web service selection based on users' preferences, the authors employed QoS attributes of web services as constraints in the search process. Therefore, the proposed model which can co-exist with the current UDDI registry includes both functional description and non-functional information of web services in the repository. During the publication process, the QoS information is checked and validated by a certifier, thus stored in the certifier's repository after the approval is granted. The UDDI registry needs to check the registered certification with the certifier before it can be stored in the repository.

In [10], a multiple criteria decision making technique – called Analytic Hierarchy Process (AHP) – has been proposed to rank the services based on their QoS values. Furthermore, QoS ontology – WS-QoSOnto, is proposed to semantically describe the QoS information of web services. The technique is based on the following four phases: 1) formulate AHP by putting every QoS attribute and candidate web services together, in order to construct a hierarchy, 2) compute the normalized weight vector of QoS attributes in each QoS group which are used in the next phase, 3) define the relative ranking of each web service by computing the Eigenvector [31] of each attribute, 4) aggregate the web service rankings for all QoS groups by building the ranking matrix.

In [32], another QoS-based web service discovery model which uses ontology to describe QoS information is presented. The approach combines constraint programming with semantic matchmaking method, in order to select web services with different QoS levels. In the proposed framework, the entire process of service discovery and selection consists of three layers: semantic matchmaking layer, constraint programming layer, and QoS selection layer. In the first layer the description logic (DL) reasoning is applied to check if the QoS attributes of the candidate services are semantically matched with the request (i.e. price & cost). The second layer which deals with QoS values converts every requested QoS condition into a set of constraints by adopting Constraint Programming (CP) method. In the third layer, an optimizing algorithm is applied to sort the candidate web services based on the total values computed by quantifying the semantic description of their QoS parameters and multiplication of each value with its related weight.

Liu et al. [33] presented a new dynamic and extensible QoS-driven model for web service discovery and selection based on users' or requesters' feedback, depending on the characteristics

of QoS criteria. The model contains a QoS registry in charge of evaluating the advertized web services based on their QoS information. This evaluation is done by: (a) generating a matrix based on web services and their QoS criteria, (b) normalizing the matrix with the purposes of enabling the uniform measurement of quality attributes independently, and (c) providing uniform indices for QoS criteria, grouping them, and setting a threshold for each group. This proposed framework is an extensible model, and any new QoS criteria (generic or domain specific) can be easily added to the system. It is preference-based, and it has a fair and open QoS computation mechanism.

Skoutas et al. [34] presented a multiple criteria matching algorithm which retrieves the *k* most dominant web services, and then ranked them based on their degree of matching. The model used three ranking criteria to match web service descriptions with the requests, using multiple similarity measurements. Based on these criteria, three algorithms are presented in the paper, including ranking by dominated score, ranking by dominating score, and ranking by dominance score. The concept of top-*k* dominant web service selection problem is formalized in the paper, and the computation of *k* most dominant web services is presented.

In [23], a QoS-aware web service discovery approach which employs matching and ranking algorithms based on user's preferences for both functional and non-functional information of web services is proposed. The model presents a new UDDI tModel with an external file that can be hosted by the service provider or other third party, to store the QoS information of web services. QoS requirements from service consumers are divided into optional and compulsory requirements including different features such as attribute name, attribute type, attribute value, attribute unit, constraints, direction, weight, and relationship, etc. Subsequently, a matching algorithm is applied to locate a set of web services which satisfy the consumers' QoS

requirements. Finally, a ranking algorithm is employed to find the most matched services with all the desired preferences.

In [35], a two-way matchmaking framework is proposed in order to overcome the problem of checking consistency between offers and requests. In order to automate the process of service selection, with consideration of two-way matchmaking context, all requests and offers should be mapped to constraint satisfaction problems. This would be carried out by mapping each parameter to a variable, and mapping every condition to the related constraints. After checking consistency of both sides, the pessimistic conformance – which is based on when all possible values satisfy the requirements – is evaluated. Finally, the best offer is selected by choosing an optimal offer which is the one with the maximum value of the calculated minimum values of all conformant offers.

2.3. Cluster Analysis

Data clustering can be described as an unsupervised classification of pattern or data items into some groups. In other words, grouping similar data objects into the same clusters based on their similarities is referred to as cluster analysis. Cluster analysis is well studied in many different disciplines such as statistics, machine learning, neural networks, data visualization, high performance computing, as well as databases and data warehouses, etc. It is believed that data clustering algorithms can extract interesting patterns from a large amount of data by dividing it into different groups based on certain similarity measures. Therefore, one of the most important subjects in cluster analysis is to understand the spatial relationships between data objects in each cluster, such as dense or sparse regions in a dataset [36].

2.3.1 Data Clustering Techniques

Clustering methods could be organized into different categories such as: partitioning methods, hierarchical methods, density based methods, grid-based methods, model-based methods, methods for high-dimensional data (such as frequent pattern-based methods), and constraint-based clustering [37]. The focus of our work is on partitioning methods and hierarchical clustering methods.

The most well-known partitioning method is the *k*-means algorithm, which groups data into a number of clusters, based on their similarities. It starts with *k* random initial prototypes, keeps assigning data objects to their closest prototypes based on their similarities or distances, and re-calculates cluster's mean which is considered as the prototypes of clusters in any iteration, until the square error criterion function converges. This criterion is based on minimizing the total sum of dissimilarities between each data object and the correspondent cluster's prototype [12, 37, 38]. The main issues with *k*-means algorithm are its sensibility to outliers, and its lack of knowledge of the number of initial clusters (*k*).

Hierarchical methods which are classified into two different types: agglomerative and divisive methods. These methods group data objects into a tree of clusters. For instance, the agglomerative (bottom-up type) algorithm starts with placing every data object in their own cluster, and iterates by joining most similar pair of clusters based on some criteria. This process ends when every data object is placed in one single cluster or when the desired number of clusters is obtained. On the other hand, the divisive or top-down type algorithm does the reverse, it starts with all objects in one single cluster, and splits the clusters into smaller pieces until each object is placed in its own cluster [37, 39].

The splitting and merging of pairs of clusters depend on the linkage methods, which can be categorized as single linkage, complete linkage, and average linkage methods. In single-linkage clustering method, the shortest distance between each cluster's individual and any member of the other cluster, which is defined as the highest similarity between them, is considered. In contrast, in complete-linkage method, the largest distance from any data object in one cluster and any object in the other cluster is considered. Average-linkage clustering method considers the average distance between any member of one cluster and any data object in the other cluster. Each of these methods has its own characteristics, for example complete-linkage algorithm generates the compact clusters, single-linkage method is sensitive to chaining individuals, thus suffers from this effect, but overall, single-linkage method is more adaptable than the other linkage methods [39].

2.3.2. Data Clustering for Interval or Symbolic Data

The traditional data clustering can be extended to deal with symbolic type of data such as set of intervals, lists, structured variables, categories, and so on, which are described as a unified and continuous set of values by means of relationship. Many approaches have been proposed in order to define the similarity between the symbolic data and perform the clustering tasks on the symbolic dataset [11, 40, 41]. Since the interval data is the most common symbolic data type, our review will focus on interval clustering algorithms although most of them can be generalized to more generic symbolic data.

In [42], an adaptive dynamic clustering method for interval data is proposed. This method uses Euclidean metric to calculate the distance between individuals and their correspondent cluster's representatives, and aims to minimize the adequacy criterion that measures the fitting

between the clusters and their representatives. There are two steps involved. In the representation step, first the partition of *k* clusters and the vector of weights are fixed, the clusters' representatives which minimize the criterion function are updated, and then the partition of *k* clusters and their representatives are fixed, and the vectors of weights which minimize the criterion are updated. In the allocation step, both vectors of weights and cluster representatives are fixed, and the clusters which minimize the adequacy criterion are updated.

In [41], two adaptive dynamic data clustering methods for symbolic data which are presented by vector of intervals, based on city-block distance are introduced. The adaptive dynamic clustering algorithm, apart from the initialization step (which is the step that the partitions are chosen by randomly selecting *k* distinct objects as the initial prototypes), has two main steps: the allocation step and the representation step. The allocation step, similar to the standard dynamic clustering algorithm, attempts to assign data objects to the correspondent classes based on their class prototypes. In the representation step, class prototypes are computed based on the individuals' assignments in the previous step. The process is iterated until the converging of the criterion function is achieved.

In any iteration of the above mentioned process, an adaptive distance is defined for each cluster depending on its structure; therefore, the adequacy criterion is locally optimized based on the fitting between the clusters and their prototypes. The adaptive distance function used in this method is based on two different types of distance functions: one-component adaptive city-block distance, and two-component adaptive city-block distance. The difference between these two functions is that in the two-component adaptive function, the lower bound and the upper bound of the intervals are managed independently, whereas in the one-component function, both are considered mutually.

In [43], an iterative dynamical clustering algorithm using the Hausdorff distance measurement is presented. This method again has two steps. In the representation step which is followed by the initialization phase (choosing *k* distinct data objects), each cluster's prototype is computed, which minimizes the criterion function based on the Hausdorff distance metric. Then, in allocation step each individual is assigned to their correspondent class prototype which is the closest cluster's representative to the individual. The algorithm is iterated until the adequacy criterion is converged to the minimum value.

In several other papers, similar distance measurements such as Euclidean, City-Block (Manhattan), and Hausdorff are employed to deal with the clustering of interval data or the vectors of intervals. In [12], two dynamic clustering methods are presented. In first method vectors of intervals are compared in order to minimize the adequacy criterion based on Hausdorff distance metric. The second method employs the weight function, and uses two-component dissimilarity based on Hausdorff distance to compare different vectors of intervals.

In [40], a hierarchical symbolic clustering algorithm using generalized Minkowski measurement for symbolic data is presented. The algorithm which works based on single linkage method, and uses both similarity and dissimilarity values, is applicable to mixed types of symbolic data including quantitative data such as ratio, absolute, and interval values, and qualitative data consisting of nominal, ordinal, and combinational values. In [44], a fuzzy clustering algorithm which uses a non-adaptive Euclidean distance for interval data is presented. The method which is a non-hierarchical clustering method, aims at providing a fuzzy partition of clusters with different dynamic distances assigned to each cluster in order to be compared with their prototypes iteratively.

2.4. Scatter/Gather Model

In 1992, Cutting et al. [9] presented a cluster based approach to browse large document collections. They proposed a browsing model called Scatter/Gather which uses document clustering as the main operation. In any iteration of the approach, the system scatters the dataset into some groups of data and shows their summary to the user. When the user selects a cluster or a number of clusters, system re-clusters the selected data, and again shows their summary of newly clustered data to the user.

The Scatter/Gather model consists of two phases: offline and online phases. In the offline phase, which uses Fractionation technique, in any iteration of clustering process a dataset is divided into a specific number of buckets. Then, by using hierarchical clustering method for each bucket, data objects are agglomerated into a specified number of clusters. Considering each generated cluster in each bucket as new individuals, the algorithm iterates with new number of data objects, until required clusters' centres are obtained. Later, every data object is assigned to each centre to build desired clusters. Finally, by applying the Split/Join refinement method and repeating the process several times, the offline phase will terminate.

The online phase uses the Buckshot technique due to its fast processing time. In this phase, $\sqrt{kn}$ number of data objects is randomly selected to agglomerate, where *k* denotes the number of clusters and *n* is the number of data in a dataset. After achieving the required clusters' centres, every individual is assigned to these centres, and then, the Split/Join refinement algorithm is applied to improve the quality of the clustering result [9].

2.5. Chapter Summary

In this chapter, we reviewed a few aspects of the related works, including QoS taxonomies from different perspectives, various QoS-aware service selection algorithms and frameworks, data clustering techniques and interval clustering algorithms, and Scatter/Gather as a cluster-based browsing model.

In the last section of the chapter, in order to have an introduction to the basis of our research, the Scatter/Gather model was studied. Although the original model is for document clustering, potentially, the model is extensible and can be applied to different areas such as browsing QoS data of web services as presented in this thesis. Due to the different context of our application, it is necessary to make some changes to the original model, such as using symbolic clustering instead of normal clustering algorithms in both online and offline phases.

CHAPTER 3

SCATTER/GATHER MODEL FOR QOS BROWSING

In this chapter we present our framework inspired by the Scatter/Gather browsing model. The proposed framework aims to organize the QoS data and help users browse through it in order to understand the QoS value distribution of available web services, then locate and select the desired services. Its main building block is the clustering component. And the web service QoS data are mainly considered as the interval data, or more generic symbolic data.

3.1. Scatter/Gather: A Cluster-Based Browsing system

As mentioned in Chapter 2, the Scatter/Gather approach was first presented by Cutting et al. [9] in 1992, and was aimed to browse a large number of documents. This method uses data clustering to separate documents into different groups based on their topics, and shows their summary to the user. Each time the user selects one or more clusters based on his/her interest, the system gathers the documents from the chosen clusters, scatters them by re-clustering them into the required number of clusters, and then shows the summary of the newly generated clusters to the user again. This narrowing down process is repeated until the user's satisfaction is met, and the desired data categories are achieved. Figure 2 indicates the process of scattering and gathering documents from the collection of *New York Times* news stories.

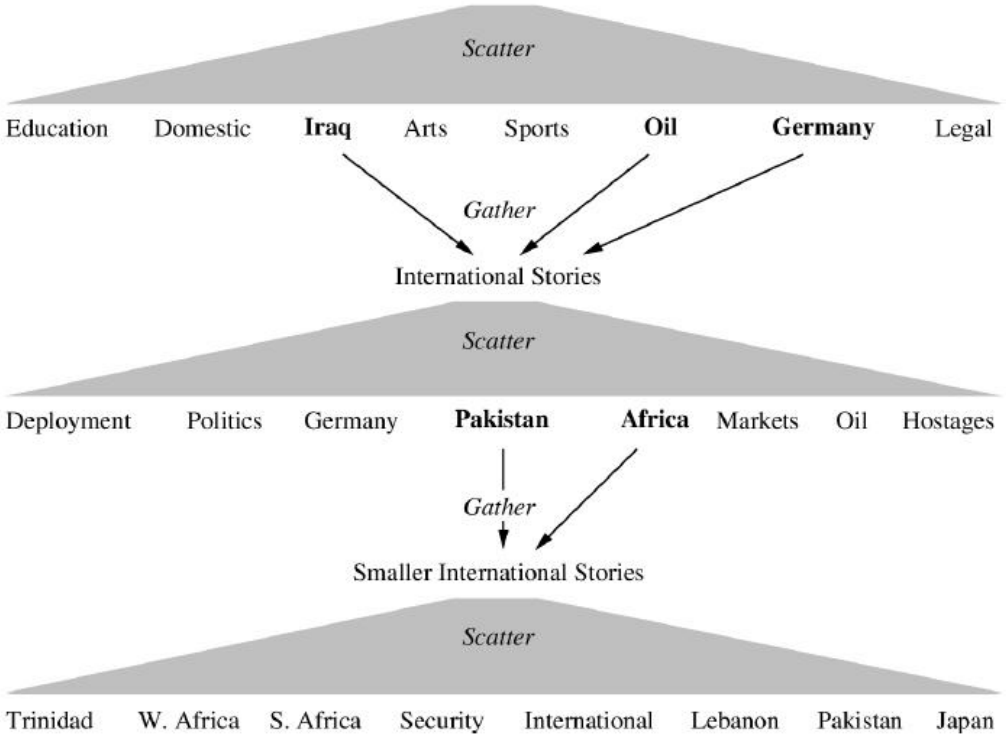


Figure 2- The Example of Scatter/Gather [45, 46]

In any iteration, the user selects one or more clusters based on summaries of clustered documents presented as their topics, and asks for new clustered information based on the latest selected data. As in the above example, in the first iteration three topics "Iraq", "Oil", and "Germany" are selected, and the system scatters the new dataset by clustering data into another eight clusters on the second level. Next, the user chooses two clusters "Pakistan" and "Africa", and demands a clustering of the chosen data. In the next iteration, eight clusters are generated again and are presented to the user [9, 45, 46].

The Scatter/Gather model has two phases: the offline and online phases. In both phases, the agglomerative hierarchical clustering is first employed to cluster data into the desired number of clusters as the initial centres. In the offline phase which has a more accurate but slower mechanism than the online phase, the system uses an algorithm called Fractionation for finding the initial centres. It breaks the dataset into a number of buckets, in order to find *k* centres. The single-linkage similarity measurement is used to merge data objects in each bucket into a certain number of clusters, which are considered as the new data objects for the next iteration of the Fractionation process. After achieving the desired number of centres, the Assign-to-Nearest algorithm is used to assign each data object to the closest cluster's centre, and this step will be repeated three times. Finally, another algorithm referred to as the Split/Join refinement algorithm is applied to improve the accuracy of the result.

In the online phase, which is based on user interactions, the system uses another algorithm called Buckshot to find the initial centres, re-clusters data, and then shows their summary to the user. In this algorithm, the system randomly selects $\sqrt{kn}$ number of data objects from the dataset, and then agglomerates them into the desired number of clusters, where *k* denotes the required number of clusters, and *n* represents the total number of data objects in the dataset. After having *k* cluster centres, the Assign-to-Nearest algorithm is used to assign data objects to their closest cluster's centre, which is repeated three times. Finally, the Split/Join refinement is applied.

The Split/Join refinement is a process of merging and splitting clusters based on their similarities. First, each cluster is divided into two sub-clusters using the Buckshot algorithm (without the refinement part) with $k = 2$. In this way, the data objects with the lowest similarities are placed in different clusters. Second, each pair of clusters with the highest similarity is joined

(agglomerated) together by calculating the distance between two clusters using the single-linkage clustering method, to make one single cluster. This process improves the clustering result, and generates more accurate clusters.

The algorithm of the Scatter/Gather could be improved by making changes in both offline and online phases to make the whole process more efficient [14, 47, 48]. In 2007, Liu, Mostafa, and Ke [48] proposed an improved Scatter/Gather model. This improved model constructs a hierarchy of documents using a hierarchical clustering technique (agglomerative or divisive) during the offline phase and the information of all levels of the hierarchy is maintained in a specific table. The previous knowledge from the first phase is used to find the required clusters in the second phase, instead of gathering and re-clustering the selected documents from scratch. When the required number of clusters is k and the number of clusters selected by the user is k' ($k < k'$), the system scans the hierarchy table from bottom to top (or the hierarchy from top to bottom), until the first cluster pairs which contain all data points selected by the user are found. Then, the cluster pairs are split by removing the entry from the table and adding its two sub-clusters' entries. This process is iterated until k clusters are obtained [47, 48].

In order to further reduce the computational time, and increase the efficiency of the first phase clustering, another algorithm, referred to as LAIR2 [14], has been proposed which uses k-means to split each cluster. In the first phase (offline phase) of this algorithm, a hierarchy using bisecting k-means ($k = 2$) is constructed, instead of the agglomerative hierarchical clustering. This means that in every iteration, each cluster is split into two sub-clusters using the k-means algorithm; instead of using linkage methods (in hierarchical clustering) to divide clusters based on the similarity between pairs of data objects. With this modification, the result that was

obtained is several hundred times faster than the previous versions. The second phase which is the online phase works similarly to the previous version of LAIR2 algorithm as explained above.

3.2. QoS Data Representation

In most of the researches the QoS values are represented as single numeric values such as the following example [7]:

```
<keyedReference tModelKey= "uddi:uddi.org:QoS:Price"
      keyName= "Price Per Transaction" keyValue= "0.01">
<keyedReference tModelKey= "uddi:uddi.org:QoS:ResponseTime"
      keyName= "Average Response Time" keyValue= "0.05">
<keyedReference tModelKey= "uddi:uddi.org:QoS:Availability"
      keyName= "Availability" keyValue= "99.99">
<keyedReference tModelKey= "uddi:uddi.org:QoS:Throughput"
      keyName= "Throughput" keyValue= "500">
```

Figure 3 – Sample of a tModel [7]

As it is indicated in Figure 3, the response time value for the specified web service has been defined as 0.05. However, it may not be the true representation of the actual values and we may have an information loss. Because the response time for a web service might be different in different invocations, depending on the network speed and other factors, it would be more accurate if it could be defined as a value range with an upper bound and a lower bound, which is also more reasonable for providers. Even for this value, an interval such as [0, 0.05] will be more accurate than the single numeric value. It is also similar for other QoS attributes, e.g. availability can be represented as [99.99, 100]. For attributes with single numeric values, they can also be easily converted to interval data, e.g. authentication: (1, 1).

We believe that this observation is true for many different QoS attributes. Therefore, the interval data would be a more appropriate type to represent the QoS values. As pointed out by [10, 32], QoS values could also be Boolean or enumeration or other types. So the symbolic data is the most appropriate type to represent the QoS attribute. In the rest of the thesis, we will mainly focus on the interval data representation of the QoS values. Any discussion on interval data can be expanded to the more generic symbolic data.

3.3. Symbolic Dynamic Clustering Algorithm (SCLUST)

The main component of the Scatter/Gather model is the clustering algorithm. Since the original application of the model is on the document collection, the data object of the clustering algorithm is a document, which is usually represented as a vector of numerical values. As pointed out in the previous section it would be more appropriate to represent QoS data of a service as a vector of symbolic values. Therefore, those popular clustering algorithms which work most effectively for document collections may not work equally well for the symbolic dataset. In order to apply the Scatter/Gather model effectively to the QoS data browsing, it is necessary to choose clustering algorithms which work best for the symbolic data.

Both partitioning and hierarchical algorithms have been used for the symbolic data clustering. Among these algorithms, the most commonly used and well studied one is the Symbolic Dynamic Clustering Algorithm (SCLUST) [12, 43, 49]. The main idea and the steps of the algorithm are similar to those of the *k*-means algorithm; however, it is catered for symbolic data. Below, we will use our QoS dataset as an example to show the steps of the algorithm.

Let $QS = \{Q_1, Q_2, \ldots, Q_n\}$ be a set of n QoS vectors and each QoS vector includes values of p attributes. Each QoS vector Q_i ($i = 1, 2, \ldots, n$) is represented as

$\left([q_{li}^1,\ q_{ui}^1],\ [q_{li}^2,\ q_{ui}^2], \ldots, [q_{li}^p,\ q_{ui}^p]\right)$ where q_{li}^j and q_{ui}^j (j = 1, 2, …, p) represent respectively the lower and upper bounds of interval values for the j th QoS attribute of this vector [7].

The algorithm can be divided into three steps consisting of the initialization step, representation step, and the allocation step. In the initialization step, k distinct vectors $(G_1, G_2, \ldots, G_k)$ which are the initial prototypes of the partition $(C_1, C_2, \ldots, C_k)$, are randomly selected. Then, all remaining QoS vectors are assigned to their clusters according to their proximities to the cluster prototypes, based on a certain distance function which will be defined later, to build the initial partitions. In the representation step, the prototypes $G_i = \left([gq_{li}^1,\ gq_{ui}^1],\ [gq_{li}^2,\ gq_{ui}^2], \ldots, [gq_{li}^p,\ gq_{ui}^p]\right)$ of the generated clusters where gq_{ui}^j is the median of $\{cq_{ui}^j, CQ_i \in C_i\}$ and gq_{li}^j is the median of $\{cq_{li}^j, CQ_i \in C_i\}$, $(j = 1, 2, .., p)$ are computed. In the third step, which is the allocation step, every QoS vector is assigned to the closest prototypes to build the new clusters. Finally, the last two steps (the representation and the allocation steps) are iterated until the criterion function converges, and a satisfactory result is achieved [7, 11, 12]. The adequacy criterion is defined as below,

$$W(P, G) = \sum_{i=1}^{k} \sum_{CQ_i \in C_i} D^2(CQ_i, G_i) \qquad (1)$$

where $D(CQ_i, G_i)$ is a dissimilarity or distance measure between a QoS vector $CQ_i \in C_i$ and the cluster prototype G_i of C_i. There have been many distance functions which have been defined in the past. In this thesis, we mainly use three of them. Their definitions are given in the following paragraphs.

The first one is the Euclidean distance measurement [37, 42, 44] and its formula is given by:

$$D_E(Q_i, Q_k) = \sum_{j=1}^{p} \left[\left(q_{li}^j - q_{lk}^j\right)^2 + \left(q_{ui}^j - q_{uk}^j\right)^2 \right] \quad (2)$$

The second one is Manhattan or city block distance metric [37, 41], obtained as:

$$D_C(Q_i, Q_k) = \sum_{j=1}^{p} \left|q_{li}^j - q_{lk}^j\right| + \left|q_{ui}^j - q_{uk}^j\right| \quad (3)$$

The third one is the Hausdorff distance metric [12, 43, 49]. Basically, the Hausdorff distance between two sets A, B $\in \Re$ is computed as follows:

$$D_H(A, B) = max\left(h(A, B), h(B, A)\right) \quad (4)$$

where:

$$h(A, B) = \sup_{a \in A} \inf_{b \in B} \|b - a\| \quad (5)$$

Therefore, the distance between two QoS attribute interval values can be calculated using the following function, where $Q_1 = [q_{l1},\ q_{u1}]$ and $Q_2 = [q_{l2},\ q_{u2}]$:

$$D_H(Q_1, Q_2) = max(|q_{l1} - q_{l2}|, |q_{u1} - q_{u2}|) \quad (6)$$

By considering Q_i and Q_k as vectors of p intervals $\left[q_{li}^j,\ q_{ui}^j\right]$ and $\left[q_{lk}^j,\ q_{uk}^j\right]$, the Hausdorff distance between these two vectors is calculated as:

$$D_H(Q_i, Q_k) = \sum_{j=1}^{p} max\left(\left|q_{ui}^j - q_{uk}^j\right| + \left|q_{li}^j - q_{lk}^j\right|\right) \quad (7)$$

3.4. A Browsing Model for QoS-Aware Web Service Selection

In this section, we present a framework consisting of different implementations of the Scatter/Gather model, based on the clustering of the QoS attributes of the web services which are represented by the vector of intervals. By using a browsing system, the user can interact with the system, learn about the actual value ranges of the QoS attributes of available services, narrow down to a few attributes or a few value ranges, and eventually locate the desired services, or sometimes be prepared with enough knowledge to switch to a searching process.

In this model, in order to ease the process of browsing, the QoS data are clustered in the offline phase and the summary of the results is shown to the user. Therefore, the user can select one or more clusters depending on their needs, and ask the system to repeat the process based on the user's selected clusters in the online phase of the algorithm. This process gets iterated until satisfactory results based on the user's preferences are achieved.

In the above-mentioned process, each cluster has the following representative information included in its summary: the size of the cluster (i.e. how many services), the range of QoS values (i.e. the minimum value within the cluster and the maximum value), the prototype, the service with QoS values closest to the prototype (prototype is the calculated centre of the cluster, not the real vector), and the homogeneity [12] of the cluster. The homogeneity criterion is used to measure the density and quality of the cluster. Its calculation is discussed later.

The framework is implemented based on the following techniques: (1) the original Scatter/Gather model with normal clustering algorithms being replaced by the interval clustering algorithms, (2) the iterative SCLUST, (3) the improved iterative SCLUST, and (4) the improved LAIR2 algorithm (H*k*-Means).

3.4.1. The Original Scatter/Gather Implementation for QoS Browsing

The model works based on the original Scatter/Gather algorithm, with the difference that the input parameters consist of the QoS attributes of the web services. Each of these attributes is represented as an interval, which is a range of real numbers, and thus, each service is represented as a vector of intervals in the system.

Similar to the original Scatter/Gather model, the algorithm consists of offline phase and online phase. In offline phase, the QoS vectors of the web services are clustered into the specified (k) number of clusters and their summaries are shown to the user. In this phase, the Fractionation algorithm is applied to break the dataset (n data objects) into b buckets of the size $m > k$, in each iteration. The initial value of m is defined as $(k + \frac{n}{n-k} + 1)$ and b is calculated as k / m.

By considering p (here we choose $p = 2$) as the number of desired clusters in each bucket, and the value of reduction in the dataset as k / p, the data objects are agglomerated into p clusters in each bucket separately. The process is iterated by calculating the new values for m, b, and the *reduction* value, and considering each generated cluster as a new individual in the object space, until the required number of clusters is generated. In this step, after defining all generated clusters' centres, the Assign-to-Nearest algorithm is applied to assign every data object into their closest centres. The distance between each data object and the correspondent cluster's centre is calculated using one of the previously mentioned distance functions. This process is repeated three times, and then the Split/Join refinement is applied, to improve the accuracy of the clustering result.

In the online phase, the Buckshot algorithm is used to group the dataset into *k* clusters. In this regard, $\sqrt{kn}$ data objects are randomly selected, and agglomerated into the desired number of clusters. At this time, after calculating the centre for each cluster, the Assign-to-Nearest algorithm is used three times to assign data objects to the closest centre based on their distances. Finally, the Split/Join refinement algorithm is applied to improve the accuracy of the result.

In the split and join refinement part, all data clusters are split into two sub-clusters using the Buckshot algorithm with $k = 2$. In this step, $\sqrt{kn}$ objects are selected, and then agglomerated into the two separate clusters using the hierarchical interval clustering algorithm. With two cluster centres, the system then keeps assigning all individuals to their correspondent clusters based on their distances. In the join part, the clusters with the highest similarity are merged into a single cluster, using one of the mentioned linkage methods.

3.4.2. Iterative SCLUST

The iterative SCLUST algorithm is a variation of the Scatter/Gather model, in which the initial centre finding and the Split/Join refinement parts have been removed, and replaced with the calculation of the adequacy criterion. Because the initial centre finding algorithm such as Fractionation or Buckshot takes extra processing time, the efficiency of the system will be affected. We would like to check whether the accuracy of the system will be largely affected when it is removed.

In this model, the same clustering process is done in both phases. However, in order to increase the processing speed (due to having a large number of data at the beginning), we still separate the process into the non-interaction phase (or offline phase) and the user interaction phase (or online phase). In both phases, the QoS dataset is clustered into a certain number of

clusters, and the summary of each cluster is shown to the user. Then, the user can select one or more clusters and ask the system to repeat the process based on newly selected data.

Each time a dataset is introduced as an input to the system, *k* objects (vectors) as the clusters' representatives are randomly selected from the dataset, and all remaining objects are iteratively assigned to the closest cluster's representative based on their distances. The new clusters' representatives are defined by calculating the centroids of the clusters, which are the mean vectors of all data objects' values in each cluster. Each time after all individuals are assigned to their corresponding clusters, the criterion function is computed, and the result is compared to its previous saved value in the last iteration. If they were identical, it means that the criterion function converges to a certain value and the algorithm stops.

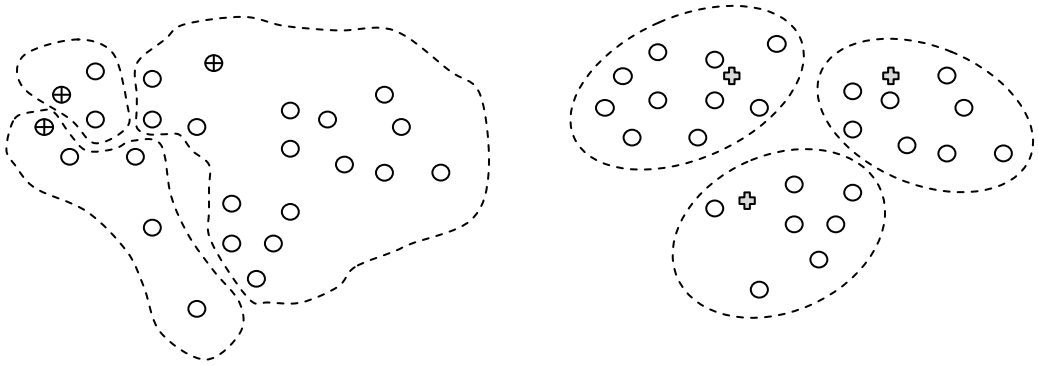
The adequacy criterion is computed based on criterion (8), which is the value of the sum of all square distances (in this formula) between every data object and its corresponding cluster's representative [7, 11, 12].

$$\mathrm{E} = \sum_{i=1}^{k} \sum_{s \in C_i} D^2(q_s, G_i) \qquad (8)$$

3.4.3. The Improved Iterative SCLUST

One of the most important issues which may cause a decrease in the accuracy of the clustering result based on the SCLUST algorithm is that, most of the time, the adequacy criterion function may converge to a local optimum, due to the random selection of the initial clusters' representatives. In this regard, in order to reduce the chance of convergence of the criterion function to a local optimum, we use an initial centre finding algorithm to predict the closest

initial prototypes to the real centres of each cluster, instead of randomly selecting the initial clusters' representative (Figure 4). To avoid the long processing time of the Fractionation algorithm in the original Scatter/Gather model, we use the Buckshot algorithm in both phases.



1- Random selection 2- Buckshot selection

Figure 4- Selection of Initial Representatives

The Buckshot algorithm is employed to select $\sqrt{kn}$ number of QoS vectors from the dataset, and agglomerate them into the required number of clusters, using the agglomerative hierarchical interval clustering method, based on the single-linkage similarity measurement. Then, by having the initial centres for *k* clusters, the algorithm does the same as the SCLUST algorithm, and keeps assigning every object to the closest centres, based on one of the introduced distance functions. Finally, after the convergence of the criterion function, the Split/Join process (without Assign-to-Nearest part) is applied to improve the accuracy of the result.

3.4.4. The Improved LAIR2 Algorithm

Because the LAIR2 algorithm has been proved to be much more efficient than the original Scatter/Gather algorithm, and the efficiency is really an important factor for an interactive application, in this thesis, we will also implement LAIR2 for our QoS browsing

system. The online phase of LAIR2 is mainly based on the pre-built hierarchy, and therefore the accuracy of the system may not be as good as the original model. Thus, we would like to investigate the possible ways to improve the accuracy of the LAIR2 algorithm while keeping a similar level of efficiency.

One of the most important subjects in cluster analysis is to understand the spatial relationships between data objects in each cluster, such as dense or sparse regions in a dataset [50]. It becomes a problem, when some clustering algorithms do not obey these relationships and distributions. For example, when a clustering algorithm creates a cluster with uniformly distributed data, it would be advisable to stop splitting that cluster. Or if it is a cluster with some sparse regions, then it should be split into a few sub-clusters. When we calculate their homogeneity or quality values, the latter one has a lower value than the former one. So we may say that the cluster with the lowest quality or homogeneity should be split first. It would be very desirable to use this principle to control the splitting process for our clustering algorithms. In this section, we propose a new approach to using the homogeneity and the quality of partitions to control the clustering process, and consequently to improve the accuracy of the LAIR2 algorithm.

The proposed algorithm is divided into offline and online phases. In the offline phase, the algorithm begins with placing every data object in one single cluster, and then divides the cluster into smaller clusters using the improved SCLUST algorithm with $k = 2$ to construct a hierarchy, with a specific index for each generated cluster. In any iteration, we use the Split/Join refinement to improve the accuracy of the generated clusters. The process iterates until each object forms its own cluster, or satisfies a certain termination condition. The mentioned steps of the algorithm for the offline phase are as follows:

1. Use buckshot algorithm to select $\sqrt{kn}$ number of prototypes;
2. Agglomerate the selected prototypes until 2 cluster centres are achieved;
3. Assign every data objects to the closest initial representatives ;
4. Calculate new cluster's centre for each cluster;
5. Assign all objects to the correspondent clusters;
6. Calculate the criterion function;
7. Repeat last three steps (4, 5, and 6) until the criterion function converges;
8. Use Split/Join refinement to achieve a better result;
9. Give the generated clusters corresponding indices, specify the relationship between parent clusters and their children, and repeat the algorithm for all existing clusters (starting from step 1);
10. Stop the process when all data objects are placed in their own clusters or satisfy a certain termination condition.

The second phase of the algorithm, i.e. the online phase, works based on the user's interaction. It searches the hierarchy which was structured in the offline phase to find the desired number of clusters chosen by the user. Here, the problem appears when the system searches for the desired number of clusters through the hierarchy. For instance, as shown in Figure 5 (1), the desired number of clusters is chosen as three. Hence, in level 2 of the hierarchy, once the first two clusters are found, the system is unable to select which cluster should be split first, and to which sub-cluster it should move (to return the three desired clusters), in order to have the most accurate results [13].

To solve this problem, we calculate the quality or homogeneity of the cluster in the search process, and sort the available clusters in ascending order based on their homogeneities.

As a result, the cluster with the lowest quality is placed on the top of our cluster list. At each iteration, in order to select the sub-clusters related to the parent cluster containing the data chosen by the user, we sort out all available clusters based on their homogeneities, and the cluster with the lowest homogeneity is selected as shown in Figure 5 (2).

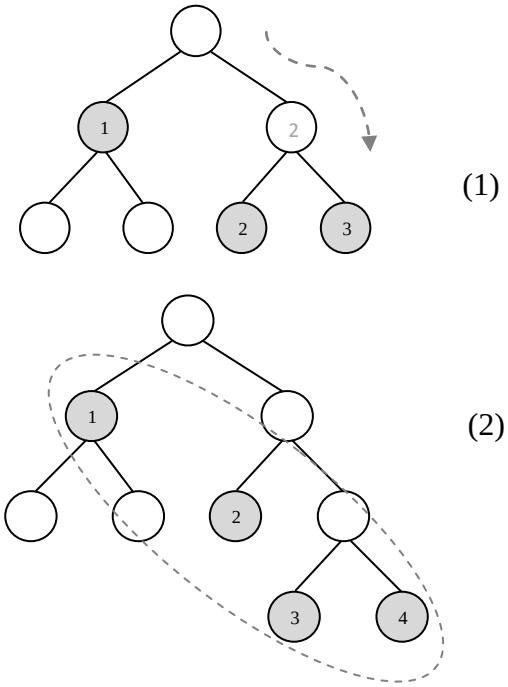


Figure 5- How to Select Clusters along the Hierarchy [13]

The mentioned steps of the algorithm for the second phase, after choosing one or more clusters by user, are as follows:

1. Calculate the selected clusters' homogeneities, and sort them in ascending order;
2. Search through the hierarchy for the first cluster in the list;
3. Replace the cluster with 2 leaf sub-clusters;
4. Repeat the process until the desired number of clusters is achieved.

Hence, in contrast to the LAIR2 algorithm, in which the search moves to the next level of hierarchy after all clusters in previous levels are split, our proposed system is able to choose any sub-cluster in any levels, and moves down in a single branch as deep as necessary to find the cluster with the lowest homogeneity [13].

Another problem with the LAIR2 algorithm is that, every time, when the user selects a cluster to be scattered, the search process restarts from the first level of the hierarchy, and this drastically slows down the processing time especially for a large dataset. However, in our proposed approach, the search always begins from the minimum index of the clusters selected by the user (which was specified for each cluster in the previous phase) in the hierarchy. Thus, the duration of the search process would be almost the same at any iteration, even for a dataset with a large amount of data [13].

In order to calculate the homogeneity of clusters, we use a generalized criterion proposed in [51], which decomposes the total inertia into between-cluster and within-cluster inertia. The adequacy between a partition P and a vector L of k prototypes is measured by (9), which is defined as the sum on k clusters and on every object $s \in C_i$ of dissimilarities $D(q_s, G_i)$ [12].

$$\Delta(P, L) = \sum_{i=1}^{k} \sum_{s \in C_i} D(q_s, G_i) \qquad (9)$$

If we use the Hausdorff distance measurement as our distance metric, we have the following adequacy criteria.

$$f_{C_i}(G_i) = \sum_{i=1}^{k} \sum_{s \in C_i} D(q_s, G_i) \qquad (10)$$

$$f_C(G) = \sum_{s \in C} D(q_s, G) \qquad (11)$$

Therefore, the homogeneity or the quality of each cluster is calculated as:

$$Q(C_i) = 1 - \frac{f_{C_i}(G_i)}{f_C(G)} \qquad (12)$$

where G_i denotes the prototype of the cluster C_i ; G is the mean of *n* vectors of QoS attributes.

3.5. Chapter Summary

In this chapter, we described: (1) the original document-based Scatter/Gather model, (2) the LAIR2 model which is based on an improvement on the Scatter/Gather algorithm, (3) different distance functions based on interval data used in our methods and the SCLUST algorithm, and (4) our proposed framework consisting of four different implementations of the Scatter/Gather model. The first method only changes the original Scatter/Gather on its clustering component, with all clustering algorithms changed for symbolic data. In the second method, the SCLUST algorithm is adopted to cluster QoS data in the offline phase, and does the same, in order to deal with user interaction in the online phase. The third model improves the second model by adding Buckshot as the initial centre finding algorithm. And finally, in the forth model which is an improved version of the LAIR2 algorithm, a hierarchy is constructed using the bisecting SCLUST algorithm, and in the online phase the homogeneity of the cluster is used to sort the selected clusters based on their homogeneity in ascending order, so as to improve the accuracy of the clustering results.

CHAPTER 4

EXPERIMENTS AND RESULTS

In this Chapter, different types of datasets are used to check the flexibility of our techniques, and compare them to each other based on some of the clustering requirements which were described in [37] such as: – Scalability, – Ability to deal with different types of attributes, – Discovery of clusters with arbitrary shape, – Minimal requirements for domain to determine input parameters, – Ability to deal with noisy data, – insensitivity to the order of input records, – High dimensionality,– Constraint-based clustering, and – Interpretability and usability. Furthermore, the results of the experiments for all algorithms of our framework are illustrated and discussed.

4.1. Framework and Testing Environment

The framework has been implemented as a windows-based application, using C# language, in Microsoft Visual Studio 2008 environment with .Net framework 3.5, to help users select their desired web services, based on the combination of the preferred QoS parameters.

Furthermore, the experiments were done with a machine configured as an Intel dual Core CPU 6300 with speed of 1.86 GHz, 1 GB RAM, with Microsoft Windows XP Professional 2002 as the platform.

4.2. Experiment Design

Generally, all the experiments which are described in the next sections are based on the following test scenarios:

1) Test the algorithms to find out which distance metric, and the linkage method (for hierarchical interval clustering) is more suitable to be used for our QoS dataset;
2) Compare four algorithms based on the following testing conditions, with different datasets:
    1.1) Synthetic datasets (distinct or overlapped) with similar distribution but different sizes (when the number of data is increased);
    1.2) Synthetic datasets with similar distribution but different numbers of attributes (when the number of attributes is increased);
    1.3) Similar datasets but different number of required clusters (when the number of clusters is increased);
3) Conduct the experiment on a real QoS dataset.

4.3. Evaluation of the Results

The following describes the three measurements we used to evaluate our algorithms and their clustering results: (1) Runtime Duration shows the processing time from when the algorithm's function is called, to when it is finished and back to the next line of the calling function. (2) Cluster's Homogeneity or quality which was described in 3.4.4 is to measure the density of a cluster, with the value range between 0 and 1 (the higher the value, the better the quality). (3) Rand Index, which is discussed in the next section, is a concept for defining the accuracy of the clustering result, in compared to a predefined clustered dataset.

4.3.1. Rand Index Calculation

Rand index is one of the most common ways to measure the accuracy of the clustering result, based on the calculation of different possibilities (or decisions), and the assignment errors which happened during the process, such as assigning a data object which does not belong to a specific cluster. These possibilities, which are based on testing each pair of data objects in all existing clusters, are divided into four action types: – true positive (TP) is when two similar data objects are assigned to the same cluster, – true negative (TN) is a situation in which two dissimilar data objects are assigned to different clusters, – false positive (FP) happens when two dissimilar objects are assigned to the same cluster, and finally – false negative (FN) is when two similar objects are assigned to different clusters. Based on this, Rand Index (RI) is defined as follows.

Given a set $S = \{O_1, \dots, O_n\}$ of *n* elements, $U = \{u_1, \dots, u_r\}$ and $V = \{v_1, \dots, v_s\}$ are two partitions of *S*. Rand Index is calculated by the following formula,

$$RandIndex = \frac{a + d}{a + b + c + d} \;\; ; \;\; 0 \leq RI \leq 1 \qquad (13)$$

where *a* denotes the number of pairs of objects in *S*, that are placed in the same cluster in U and V, *b* is the number of pair of objects which are placed in the same cluster in U, but in different cluster in V, *c* is the number of pairs of objects which are placed in different cluster in U but the same cluster in V, *d* is the number of pairs of objects which are placed in the different cluster in U and V [46, 52].

4.4. Data Generation

As previously mentioned, in order to illustrate the differences between the four algorithms, we used different datasets with different sizes and distributions, generated by MATLAB. In this regard, 9 datasets with their data points which follow the multivariate normal distribution pattern using different mean vectors μ and specified covariance values σ, were employed. As an example, we have chosen three synthetic datasets to demonstrate their overall data distributions based on specific mean vectors and covariance values. The rest of the datasets and their input parameters could be found in Appendix A.

The first simulated dataset including randomly generated 3,000 data follows a multivariate normal distribution, in which the data has been considered as a vector of three intervals, consisting of Cost ($), Response time (ms), and Reliability (%) respectively (Table 1). Nine main data vectors have been chosen as the initial mean vectors for data generations to create nine distinct clusters using their own covariance values. At the end, based on what was mentioned, three distinct clusters (each consisting of 1,000 data vectors), in which each cluster contains three sub-clusters containing 330, 330, 340 data vectors respectively, were generated.

Table 1 – Input Parameters for Dataset-2 Containing Distinct Clusters and Distinct Sub-Clusters

	Cost ($)		Response Time (ms)		Reliability (%)	
	μ	σ	μ	σ	μ	σ
Cluster 1 (1000) (330,330,340)	55	2.0	100	2.0	10	1.0
	80	2.0	140	1.8	13	1.2
	125	2.0	170	3.0	17	0.9
Cluster 2 (1000) (330,330,340)	310	3.0	400	3.0	42	0.8
	360	3.6	450	2.4	45	1.3
	420	4.0	510	4.0	47	0.6
Cluster 3 (1000) (330,330,340)	660	2.5	760	3.6	82	1.2
	700	3.9	800	3.0	85	1.0
	740	2.0	830	2.0	88	0.9

In the second dataset, nine initial mean vectors containing 3,000 randomly generated data vectors based on multivariate normal distribution, have been chosen to create three distinct clusters, which include three indistinct or overlapped sub-clusters, all having closer mean vectors and larger covariance values (Table 2). Therefore, each predefined distinct cluster contains 1,000 data objects which are subdivided into three overlapped sub-clusters consisting of 330, 330, 340 data vectors respectively.

Table 2 – Input Parameters for Dataset-5 Containing Distinct Clusters with Indistinct Sub-Clusters

	Cost ($)		Response Time (ms)		Reliability (%)	
	μ		σ		μ	
Cluster 1 (1000) (330,330,340)	150	10	100	4	50	2
	170	9	110	5	55	3
	200	12	120	6	57	3
Cluster 2 (1000) (330,330,340)	410	10	250	4	66	2
	430	20	260	6	69	2.5
	460	15	275	7	73	3
Cluster 3 (1000) (330,330,340)	700	12	370	8	80	3
	730	11	385	10	85	2
	750	8	400	10	88	3

In the third dataset, generated with the same characteristics of the previous examples, there are 3,000 data objects, represented as the vectors of six interval data which have the six QoS attributes including: Cost ($), Response time (ms), Reliability (%), Availability (%), Accessibility (%), and Security (%) respectively (Table 3). In this regard, three initial mean vectors of six interval values were defined to generate three predefined distinct clusters, in which each cluster contains three sub-clusters and some of the elements of data vectors in one cluster are overlapped with some elements of data vectors in other two clusters. It means that, we may have different data vectors with the same similarity between their attributes, placed in different

clusters. The idea of using this type of dataset is to check the accuracy of the vector clustering when there exists redundant data placed in different clusters.

Table 3 – Input Parameters for Dataset-8 Containing Distinct Clusters with Redundancy

	Cost ($)		Response Time (ms)		Reliability (%)	
	μ		σ		μ	
Cluster 1 (1000) (330,330,340)	150	2	100	3	45	2
	170	2	100	3	50	1.5
	200	2.5	120	2	55	1
Cluster 2 (1000) (330,330,340)	310	2	250	3	64	1.2
	325	3	275	2	69	1
	350	3.5	275	2.5	76	1.2
Cluster 3 (1000) (330,330,340)	420	2	100	4	83	1
	435	3	160	3	89	0.5
	460	=3	275	3	95	1

	Availability (%)		Accessibility (%)		Security (%)	
	μ		σ		μ	
Cluster 1 (1000) (330,330,340)	82	1.5	100	0	48	2
	89	1.8	92	3	54	3
	95	1	95	1	60	3
Cluster 2 (1000) (330,330,340)	62	1.8	58	2	72	2
	77	1	65	2	77	1.8
	70	1.5	50	1.8	83	2
Cluster 3 (1000) (330,330,340)	52	1.2	75	1.2	88	1.1
	45	1.2	80	0.8	96	1
	40	1.5	85	1.2	54	3

4.5. Experiment-1: Choosing Appropriate Distance Functions and Linkage Methods

To shorten the names of the algorithms, we use the following abbreviations for the presented techniques: S/G for the original Scatter/Gather technique, ISC for the iterative

SCLUST algorithm, ISC2 for the improved ISC algorithm, and NLA (New LAIR2) for the improved LAIR2 algorithm based on interval data. This notation will be used in the rest of the thesis.

The application is able to execute each method based on three different distance metrics for the interval data including: Euclidean (EU), City-block (CB), and Hausdorff (HD). Furthermore, it can be run based on three different linkage methods for hierarchical interval clustering algorithm used in the original Scatter/Gather, including single linkage, average linkage, and complete linkage methods. Therefore, in order to choose the most appropriate methods, we tested our algorithms based on different distance metrics and linkage methods separately (Table 4). We have used a few datasets with different distribution patterns to conduct this experiment. Below is an example dataset (Dataset 4), containing three distinct clusters with three overlapped sub-clusters which is illustrated as below (Figure 6).

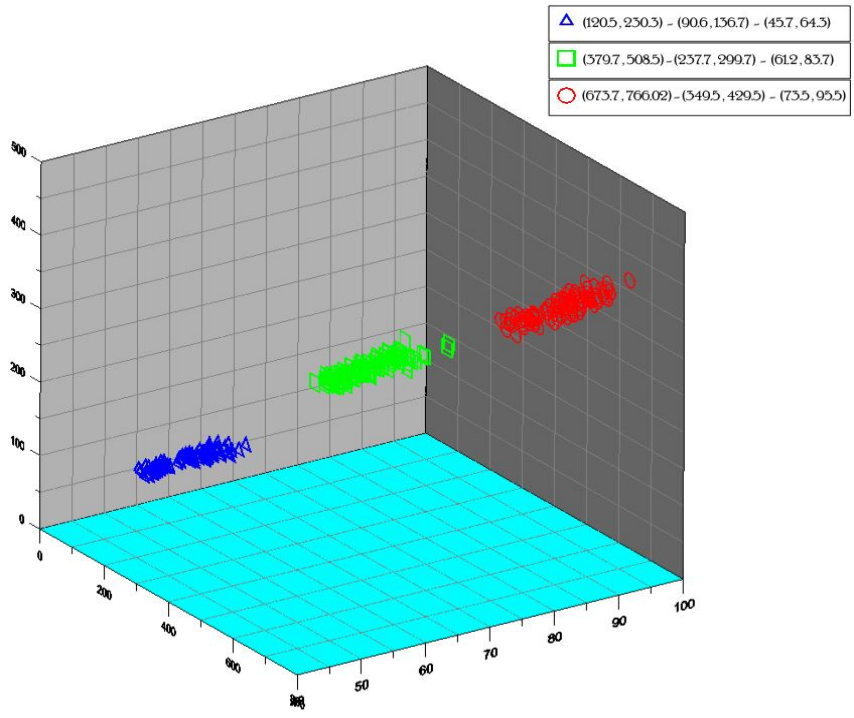


Figure 6 – The Distribution of Data in Dataset 4

Table 4 – Results for Dataset-4 Using All Distance Functions and Linkage Methods

	S/G								
	Single			Average			Complete		
	EU	CB	HD	EU	CB	HD	EU	CB	HD
Duration	0.2.752	0.2.755	0.2.738	0.3.881	0.3.856	0.3.867	0.2.797	0.2.805	0.2.803
Homogeneity	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989
Rand Index	1	1	1	1	1	1	1	1	1
	ISC2			ISC			NLA		
	EU	CB	HD	EU	CB	HD	EU	CB	HD
Duration	0.0.277	0.0.280	0.0.277	0.0.228	0.0.209	0.0.184	0.7.897	0.7.886	0.7.866
Homogeneity	0.989	0.963	0.989	0.94	0.941	0.989	0.989	0.989	0.989
Rand Index	1	0.963	1	0.934	0.934	1	1	1	1

When we compare different distance functions, the lowest duration is consistently from the Hausdorff distance measurement for these four algorithms. Accuracy-wise, there is no obvious winner. Hausdorff usually performs well regarding its rand index values. Based on our literature review, Hausdorff was advocated as the most commonly used distance function among the three. Therefore, we will use it in our experiment later. Comparing different linkage methods for Scatter/Gather algorithm, single linkage method consistently performs the best. According to these results, we are going to use the following setting in the rest of the experiments:

- Distance metric: Hausdorff.
- Linkage method for Hierarchical clustering: Single-linkage method, as it was briefly mentioned in section 2.3.1, about the more adaptability of the single-linkage in comparison to other linkage methods [39].

- Number of runs: 10 for the datasets with 300 data objects, and 5 for the datasets with more than 1000 data points, and 2 for the datasets with 30,000 data objects.

4.6. Experiment-2: Different Datasets, Attributes, and Number of Clusters

Altogether there are 9 datasets. With the different datasets, we could test the performance change in different scenarios. We have defined three groups of testing scenarios: changing the size of the dataset, changing the number of QoS attributes, and changing the number of clusters (e.g. *k* value).

In this section, the results from different testing scenarios are presented and discussed. In most cases, we did our experiments for two levels of browsing (2 iterations) – initial level and one level after user selection. In the first level or initial level, we repeated each clustering algorithm for a number of times, and calculate the average values of the clustering results. In second level, by choosing the clustering result from the previous level, we execute the application for a few times and get the average value for results.

Test1: Similar Distributed Datasets with Different Sizes

The following datasets contain 300, 3000, and 30000 QoS vectors respectively, and each vector includes three QoS attributes. There are three distinct clusters on the first level, and each cluster consists of three distinct sub-clusters on the second level. Below Table 5-7 show the values of three measurements – duration, homogeneity and rand index in the first and second iteration, when identifying the first level and second level sub-clusters.

Table 5 – Results for Dataset-1 with 300 Data

Run: 10	Dataset-1					
Data	300			100		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	0.2.667	0.98	1	0.0.261	0.999	1
ISC	0.0.164	0.917	0.898	0.0.061	0.997	1
ISC2	0.0.242	0.98	1	0.0.091	0.999	1
NLA	0.6.519	0.955	0.984	0.0.016	0.999	1

Table 6 – Results for on Dataset-2 with 3,000 Data

Run: 5	Dataset-2					
Data	3000			1000		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	0.45.605	0.998	1	0.2.717	0.996	1
ISC	0.1.891	0.998	1	0.0.775	0.916	0.89
ISC2	0.2.641	0.998	1	0.0.875	0.996	1
NLA	1.34.520	0.998	1	0.0.121	0.996	1

Table 7 – Results for Dataset-3 with 30,000 Data

Run: 2	Dataset-3					
Data	30000			10000		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	84.48.501	0.811	0.891	2.0.921	0.999	1
ISC	0.22.078	0.77	0.864	0.9.159	0.975	0.898
ISC2	0.32.547	0.98	1	0.9.937	0.999	1
NLA	28.57.766	0.917	0.962	0.0.875	0.992	0.996

From these tables, it can be observed that both S/G and NLA take much longer processing time on the first level than ISC and ISC2, whereas the time difference on the second level is much smaller and within an acceptable range. The processing time for NLA is higher than that for S/G in the smaller datasets with 300 and 3000 vectors, however by increasing the size of the dataset, the duration time for S/G drastically increases to almost two times higher than NLA for the dataset with 30000 vectors due to more iterations in the refinement part, which uses the hierarchical clustering method.

The clustering time on the second level from NLA is always the smallest among all four methods. Since for the interactive browsing, the offline processing time is not very important, whereas the online clustering time is really crucial to attract users to use the browsing system. From this perspective, NLA – our improved version of LAIR2 algorithm, is the most efficient implementation of Scatter/Gather model for the QoS interval dataset. It reconfirms the conclusion from [13, 14].

When we check the homogeneity and rand index, ISC usually gets one of the worst results due to its weakness (the convergence of the criterion function to a local optimum). Since all the other three methods use certain ways to find initial centres which are closer to the real centres, whereas ISC just randomly chooses the initial centres, it indicates the effectiveness of adding the initial centre finder component in the clustering algorithm. ISC sometimes suffers from getting stuck in a local optimum.

Duration of ISC2 on both levels is always longer than that of ISC, which is mainly due to the additional time used for identifying the initial centres. But they get very close to each other when the size of the dataset is increased. The main reason is that in ISC by choosing imperfect

initial prototypes (e.g. choosing *k* similar objects which belong to the same cluster as the initial cluster representatives), the process iterates several times more to find the optimum centroids, whereas in ISC2, by choosing the most optimum initial cluster representatives at the beginning of the process, much faster iterations can be achieved to reach the final clustering result. Both homogeneity and rand index of ISC2 are better than those of ISC, which indicates that although the processing time of ISC2 is longer, its accuracy is also higher. When the system requires a high clustering accuracy, ISC2 would be preferred than ISC.

The best homogeneity and rand index results on both levels are from ISC2. S/G also performs very well in the first two datasets, and it is slightly worse on the first level results for the third dataset. NLA could always achieve a reasonable quality result which is usually not too much worse than the best result. Due to its bisecting splitting behaviour (based on $k = 2$ instead of 3), it may have some small errors in the clustering result which are usually negligible. ISC performs consistently the worst on the two accuracy metrics.

In conclusion, for these three datasets, if we want to choose an implementation which is best efficiency-wise, it is NLA, and if we want to choose one with the best accuracy, it is ISC2. Overall speaking, NLA is most preferred for online interactive browsing because of its shortest online clustering time and a high level of accuracy (although not the best).

Figure 7 below shows the comparison of four methods when the size of the dataset increases. It is based on dataset 1, 2 and 3.

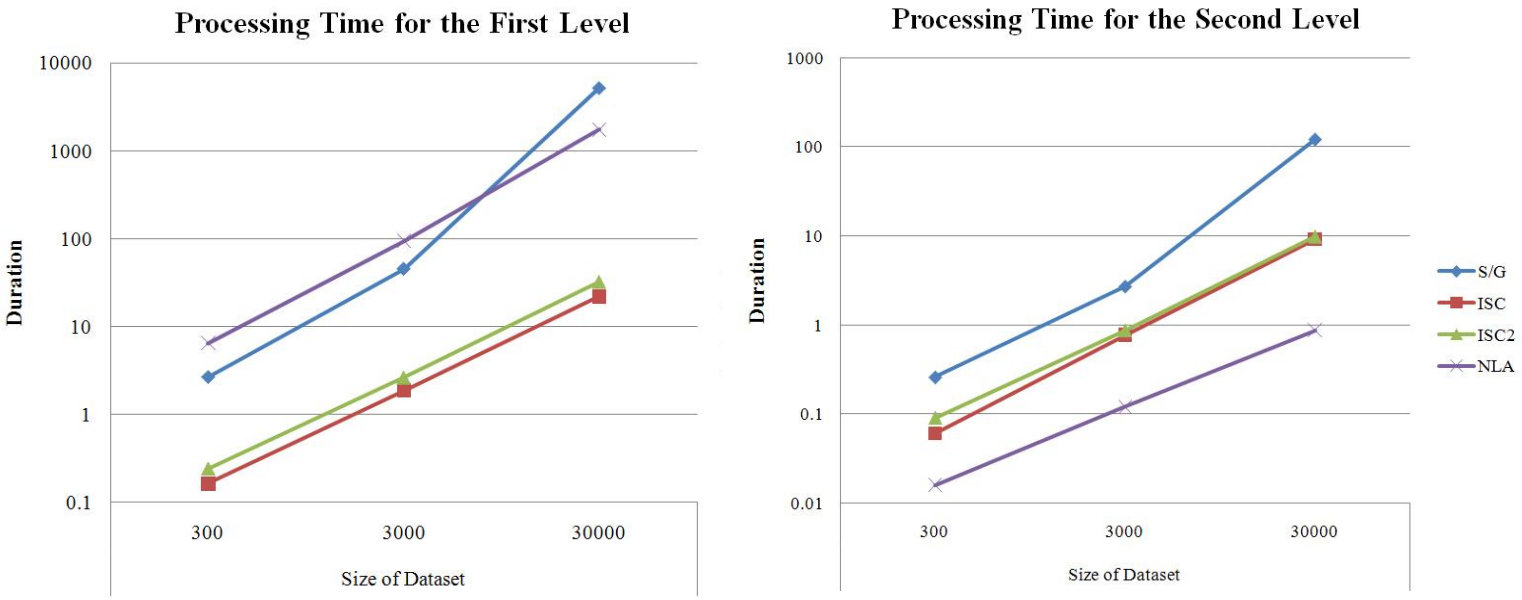


Figure 7 – Comparing Clustering Time on Two Levels for Dataset 1, 2, and 3

From this figure, it is observed` that if we look at the clustering time for level 1, which basically measures the efficiency of the offline clustering phase, both ISC and ISC2 are better than the other two algorithms. However, if we look at the time for level 2, which measures the online phase efficiency, NLA is the best whereas S/G is the worst. When the size of the dataset increases, the time increases almost linearly for all algorithms.

If we compare the homogeneity and rand index results on datasets with increasing sizes based on Table 5-7, this kind of linear relationship may not hold. The lower value of homogeneity on dataset 3 is mainly due to the existence of three sub-clusters within each cluster. And the value of rand index is usually controlled by the data distribution pattern. When there is more overlapping between clusters, the rand index value is generally lower.

Tables 8-10 show the experimental results of similar tests. The difference is that in these datasets, although the first level clusters are distinct, the second level sub-clusters are overlapped or indistinct. Our aim was to test our techniques for different distribution patterns.

Table 8 – Results for Dataset-4 with 300 Data

Run: 10	Dataset-4					
Data	300			100		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	0.2.666	0.989	1	0.0.261	0.998	0.775
ISC	0.0.188	0.94	0.934	0.0.102	0.998	0.836
ISC2	0.0.261	0.989	1	0.0.116	0.998	0.917
NLA	0.6.616	0.989	1	0.0.016	0.998	0.954

Table 9 – Results for Dataset-5 with 3,000 Data

Run: 5	Dataset-5					
Data	3000			1000		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	0.45.262	0.989	1	0.2.749	0.998	0.693
ISC	0.1.479	0.989	1	0.1.072	0.998	0.784
ISC2	0.2.038	0.989	1	0.1.182	0.999	0.813
NLA	1.34.776	0.989	1	0.0.141	0.999	0.81

Table 10 – Results for Dataset-6 with 30,000 Data

Run: 2	Dataset-6					
Data	30000			10000		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	79.40.562	0.989	1	1.53.531	0.978	0.899
ISC	0.48.984	0.741	0.898	0.14.958	0.983	0.914
ISC2	0.59.343	0.989	1	0.13.781	0.998	0.968
NLA	29.26.219	0.989	1	0.0.903	0.998	0.942

We could almost get the same conclusion as the previous set of experiments. Efficiency-wise, ISC achieves the best performance on the first level and NLA achieves the best performance on the second level. Accuracy-wise, ISC2 again has the highest rand index and homogeneity values on both levels, NLA is the second best, and S/G also has a pretty good performance. There is an obvious drop on the rand index value for all the datasets on the second level from all four methods, which is mainly due to the higher level of overlapping between those sub-clusters. Generally speaking, ISC2 and NLA could achieve a better rand index value on the second level than S/G and ISC. The homogeneity values are still high despite the overlapping between the sub-clusters. Figure 8 below shows the comparison of four methods when the size of the dataset increases for dataset 4, 5 and 6.

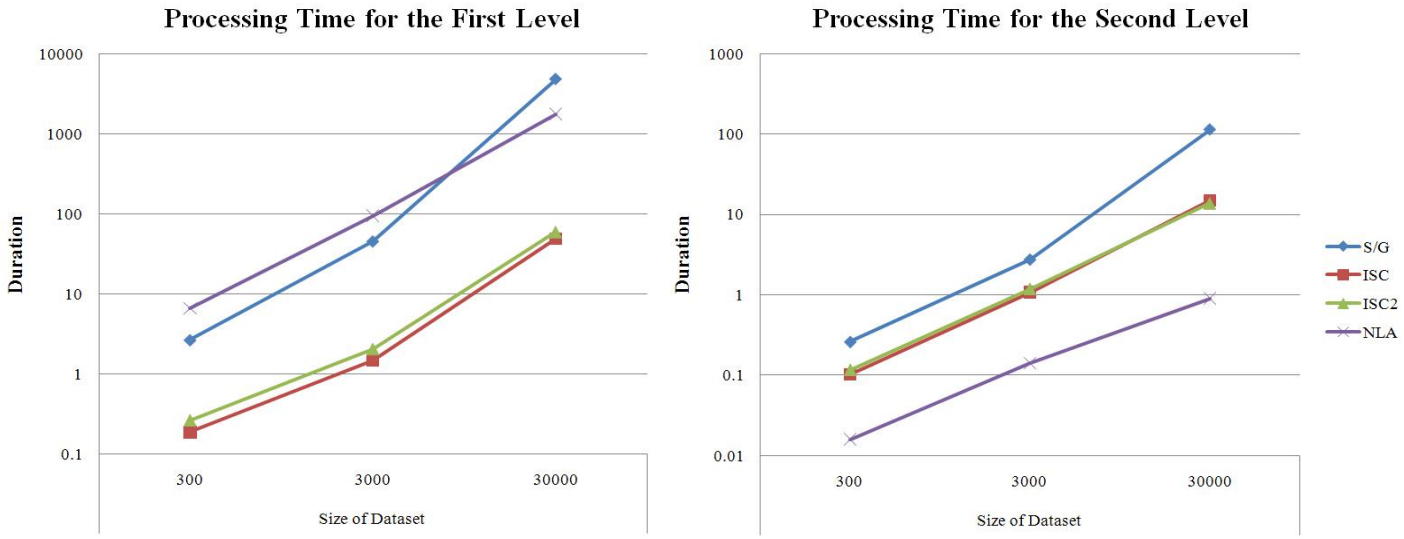


Figure 8 – Comparing Clustering Time on Two Levels for Dataset 4, 5, and 6

Again, there is a close-to-linear relationship between the duration and the size of the dataset. On the first level, the lowest duration belongs to ISC and the duration from ISC2 is very close to it. Also, the processing time for the original Scatter/Gather starts with a value which is

lower than NLA, but in 30,000 dataset it increases more than NLA. As for the second level, NLA consistently performs the best and S/G the worst.

Test2: Same Datasets with Different Number of Attributes

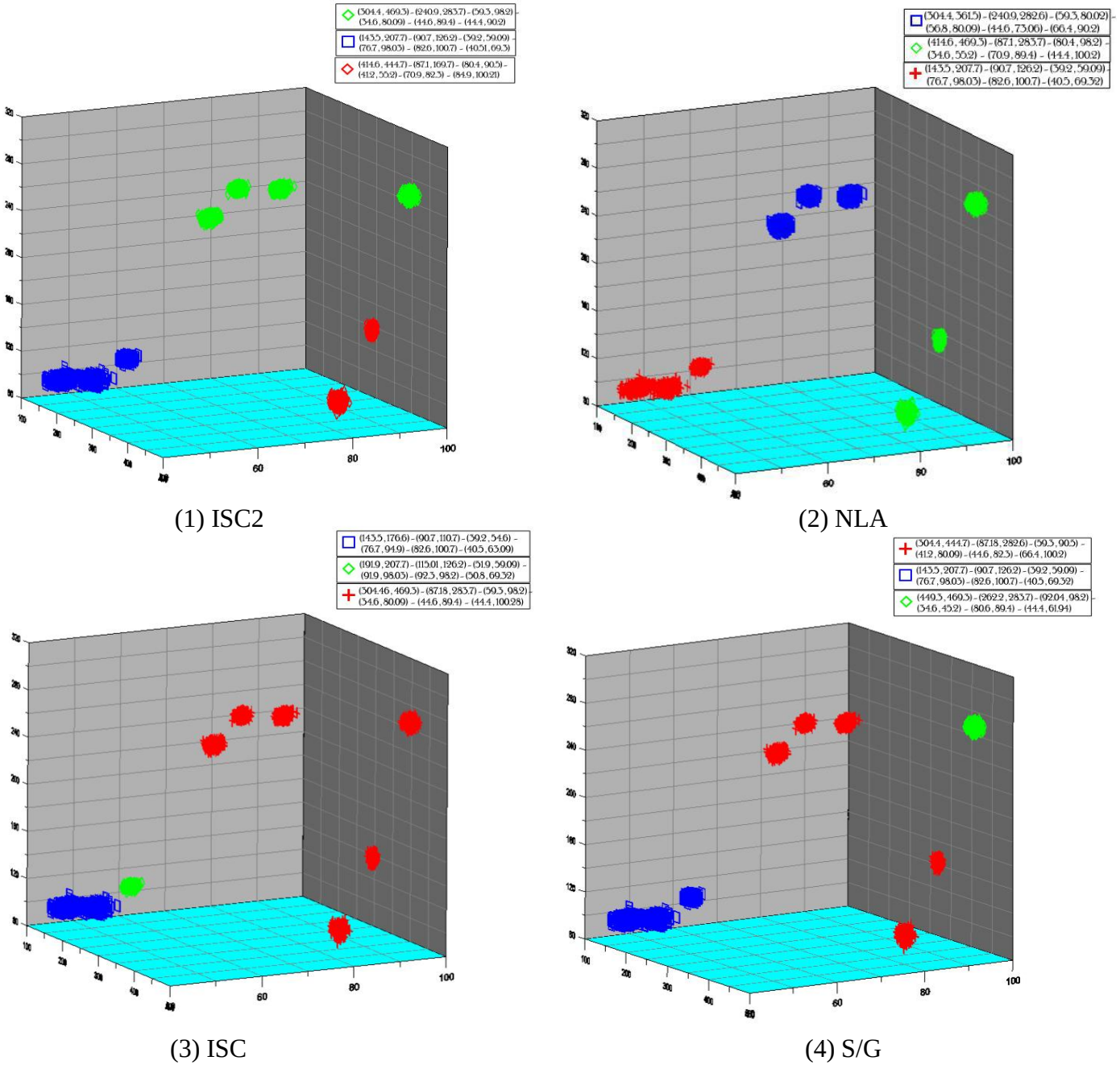
Table 11 shows the results of the experiment for a dataset with 3000 vectors of six QoS attributes (Dataset 8). As previously mentioned, the purpose of this test is to check how the performance is changed when the number of attributes is increased. The dataset has three distinct predefined clusters and three distinct sub-clusters for each.

Table 11 – Results for Dataset-8 with 3,000 Data

Run: 5	Dataset-8					
Data	3000			1000		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	1.18.646	0.812	0.819	0.4.078	0.999	1
ISC	0.2.016	0.838	0.889	0.0.617	0.999	1
ISC2	0.3.076	0.875	0.908	0.0.971	0.999	1
NLA	2.1.052	0.893	1	0.0.161	0.999	1

In this test, the most accurate clustering results in both iterations belong to NLA with the rand index of 1. The next most accurate clustering algorithm is ISC2. The lowest processing time is from ISC in the first iteration and NLA in the second iteration. The change does not have a big effect on homogeneity and rand index values on both levels. Regarding the duration, it is higher in the first iteration for all four methods, whereas in the second iteration, S/G and NLA get a longer duration time, but ISC and ISC2 are not largely affected by the change.

In the following figure (Figure 9), the clustering results of all methods are illustrated by 3D figures. We could see that every method would generate different clustering result. When comparing the original data distribution with the result, intuitively, ISC2 and NLA give the best result, and results from both ISC and S/G are not so good.



(1) ISC2 (2) NLA

(3) ISC (4) S/G

Figure 9 – Results for experiments on dataset-8 (Level 1)

The next experiment is based on a dataset (Dataset 9) with 3000 vectors of nine QoS attributes, consisting of three distinct clusters and three distinct sub-clusters for each cluster. The aim of this experiment is again to check how the performance is changed when the number of attributes is increased. The results are shown in Table 12.

Table 12 – Results for Dataset-9 with 3,000 Data

Run: 5	Dataset-9					
Data	3000			1000		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	1.48.615	0.959	1	0.5.083	0.998	0.878
ISC	0.2.026	0.956	1	0.0.708	0.998	0.808
ISC2	0.3.589	0.956	1	0.1.125	0.999	1
NLA	2.19.792	0.956	1	0.0.208	0.999	1

Above table clearly indicates that in first level, the efficiency of ISC is higher than the rest of the algorithms in the order of ISC2, S/G, and NLA. However, in the second level, NLA is the fastest one in comparison to other techniques, followed by ISC and ISC2 which are not very different in their processing time. The original Scatter/Gather in this level is the slowest one mainly due to its three times of repeating the Assign-to-Nearest process. Regarding to their accuracy levels, both ISC2 and NLA are equally well. Overall speaking, NLA is the best one considering its fastest online processing time and high accuracy of the clustering results.

Figure 10 below shows the relationship between the number of attributes and the duration. We could see that in the first level of browsing, the most efficient algorithm is ISC followed by ISC2, S/G, and NLA and their durations increase almost linearly with the increasing of the number of attributes. However, in the second level the most efficient one is NLA, and the

next efficient algorithms respectively are listed as: ISC, ISC2, and S/G. In the second level, for ISC and ISC2, the changes of the duration from dataset-8 with 6 attributes to dataset-9 with 9 attributes are small compared to those from 3 attributes to 6 attributes. For S/G and NLA, the change is almost linear.

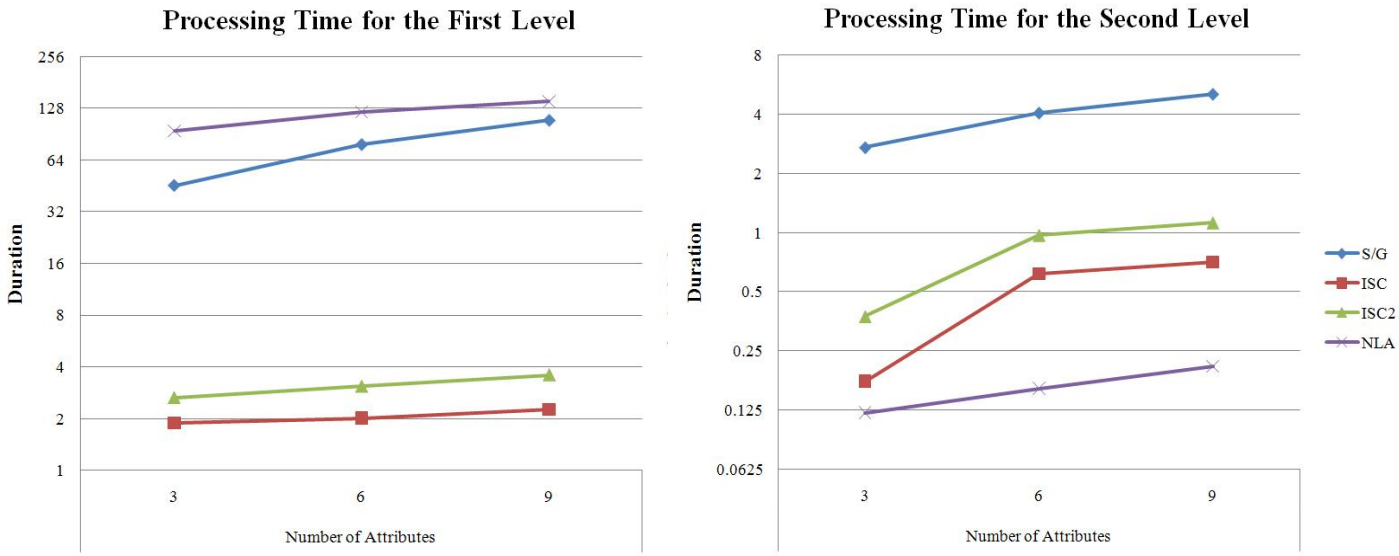


Figure 10 – Comparing Clustering Time on Two Levels for Dataset 2, 8, and 9

Again, if we compare the homogeneity and rand index values for dataset 2, 8 and 9 (Table 6, 11, 12), the change is not linear, and it is mainly controlled by the data distribution patterns.

Test3: Same Size Datasets with Different Numbers of Clusters

The aim of the next experiment is to test how each one of our proposed algorithms deals with the different number of clusters (*k* values). We use Dataset 1 for the test. In our previous result (Table 5), three clusters are generated in the first level, and three sub-clusters are generated

for each cluster in the second level. Now we change the *k* value to 9 for both levels to compare their performances. Table 13 shows the results.

Table 13 – Results for Dataset-1 with 9 Clusters

Run: 5	Dataset-1					
Data	300			300		
Iteration	1			2		
	Duration	Homogeneity	Rand Index	Duration	Homogeneity	Rand Index
S/G	0.3.057	0.997	0.891	0.0.276	0.998	0.952
ISC	0.0.266	0.988	0.885	0.0.078	0.997	0.927
ISC2	0.0.365	0.999	1	0.0.132	0.999	0.982
NLA	0.7.703	0.999	0.912	0.0.031	0.998	0.961

Again, we could get similar conclusions as before. NLA is the most efficient one for online clustering and ISC2 is the most accurate one for both levels. Figure 11 below shows the comparison of the processing time for dataset-1 when the number of clusters is 3 and 9.

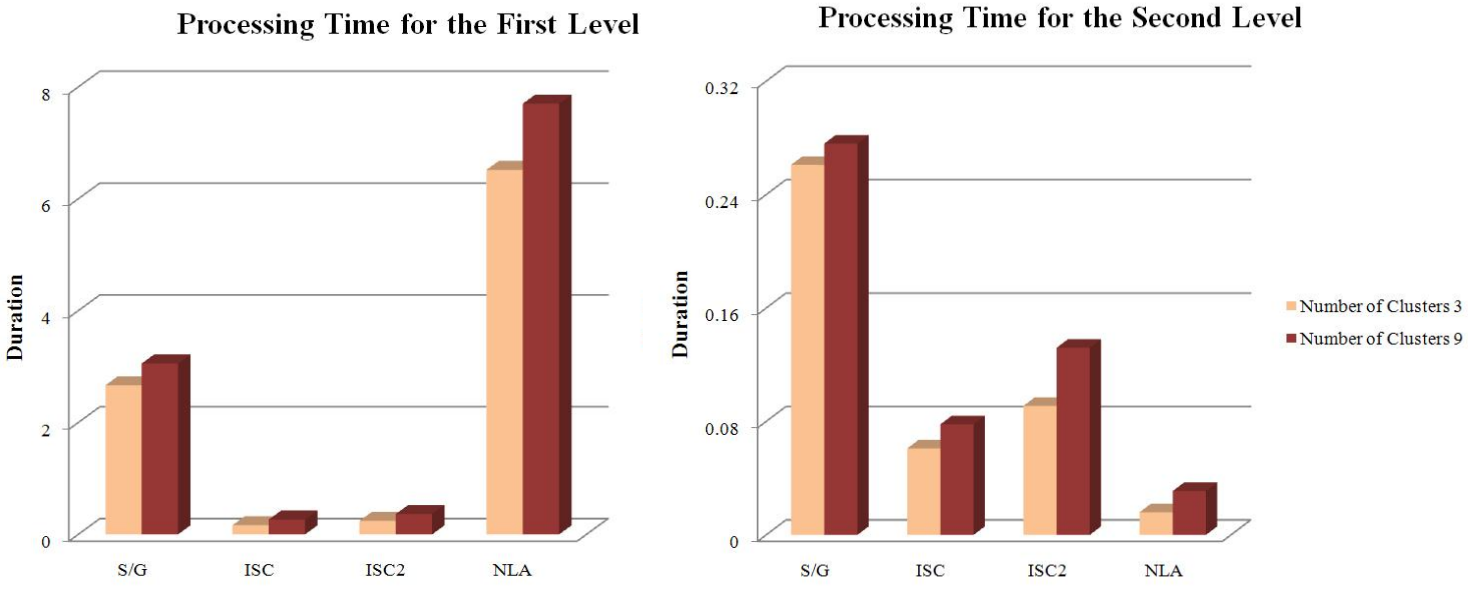


Figure 11 – Comparing Clustering Time on Two Levels for Dataset 1 with 3 Clusters and 9 Clusters

We could see that when the number of clusters is changed from 3 to 9, its processing time is also increased for both levels. The degree of increase in the first level is very obvious, and we could see a considerable jump (but not that big) in all techniques. The increase in the second level is not that obvious.

Test4: Same Size Datasets with Different Distributions

Our next experiment is based on a dataset (dataset 7) including three predefined indistinct clusters. The dataset contains 3000 data vectors with four QoS attributes. Only one level clustering is done on this dataset. The purpose of this test is to check how the accuracy is affected when the overlapping degree is higher, and we also want to compare the accuracy of each technique for this kind of distribution pattern. The results are shown in Table 14.

Table 14 – Results for Dataset-7 with 3,000 Data

Run: 5	Dataset-7		
Data	3000		
Iteration	1		
	Duration	Homogeneity	Rand Index
S/G	0.56.854	0.383	0.547
ISC	0.3.375	0.578	0.741
ISC2	0.4.594	0.589	0.772
NLA	1.54.786	0.581	0.774

Above table (Table 14) shows how these algorithms deal with a dataset with a high overlapping degree between clusters. Compared with the results from Dataset 2 and 5 (as shown in Table 6 and 9), which also contain 3000 QoS vectors, the homogeneity and rand index values are much lower, and the duration for ISC and ISC2 is also slightly higher. We believe that the high overlapping is the main contributing factor to the lower accuracy level, and the longer

duration time is due to the larger number of elements of the QoS vectors (four attributes for this dataset). Regarding the comparison between four algorithms, as it is clearly indicated in the table, the techniques with the most accurate clustering results are sorted as follows: NLA, ISC2, ISC, and S/G. However, the lowest duration belongs to the ISC and the highest is for the NLA.

Figure 12 depicts the comparison of the accuracy of dataset 2 and 7, based on their total clusters homogeneity and rand index for the first level. We could see that when the degree of overlapping between clusters is increased, both homogeneity and rand index values are lower.

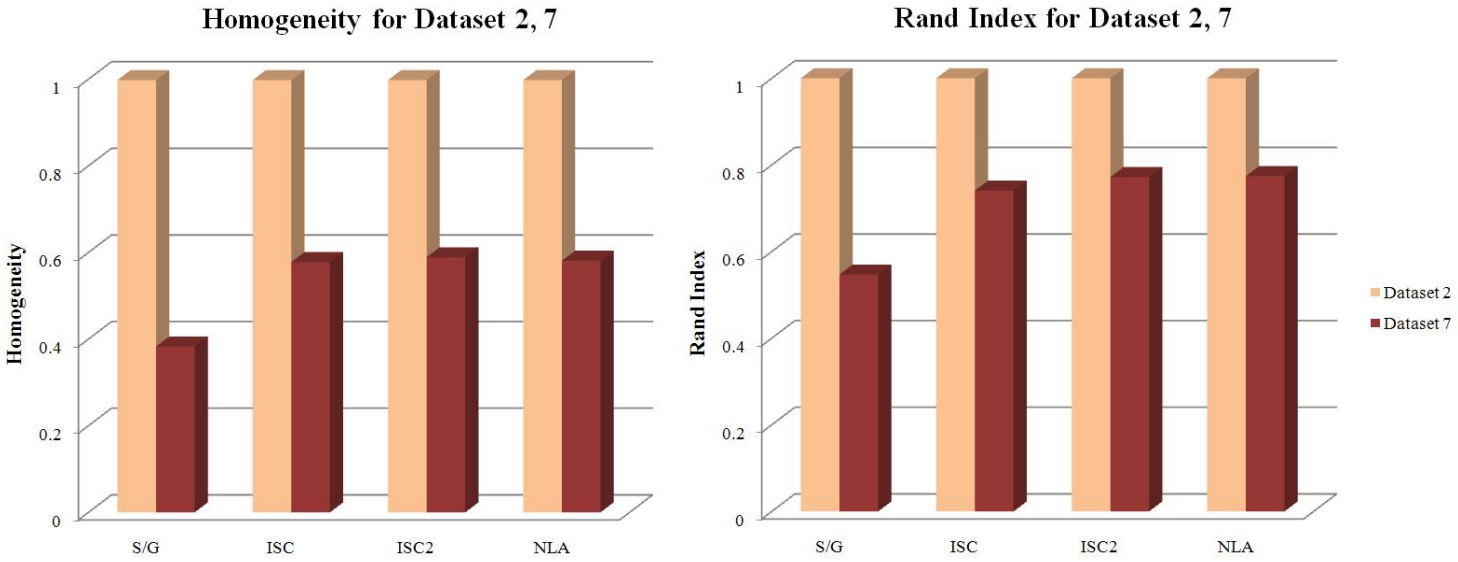


Figure 12 – Comparing the Accuracy for Dataset 2 and 7

4.7. Experiment-3: A Real QoS Dataset

Our last experiment is based on a real dataset (QWS dataset) which consists of 2,507 web services and their QoS measurements. The data was obtained in 2008 [53, 54, 55]. Since our

browsing system is mainly for services with similar functionalities, we manually assign keywords to all the services, and sort the services based on their functionalities.

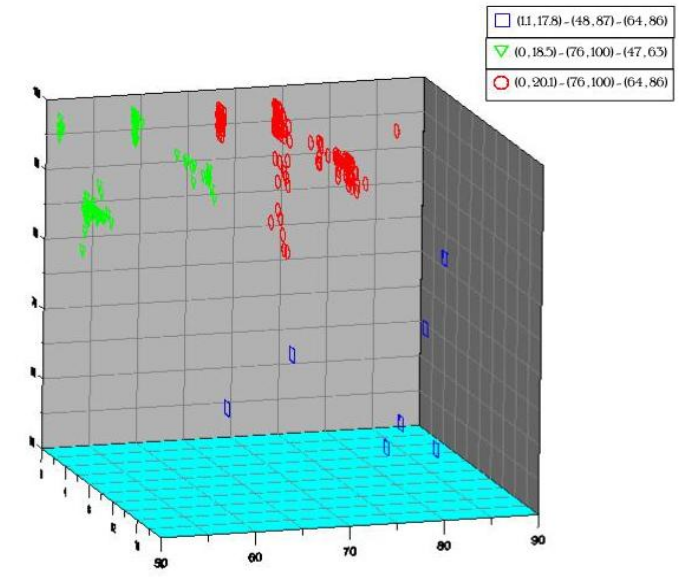
After checking the functional keywords which contain a good number of services, we chose the topic “bioinformatics” with its 281 services for our testing. We chose three QoS attributes, namely: Reliability, Successability, and Throughput.

Table 15 shows the results of the experiments for four methods to be compared based on their processing time and their total clusters homogeneities. Due to the lack of any predefined clusters related to this dataset, we couldn’t calculate their Rand Index values.

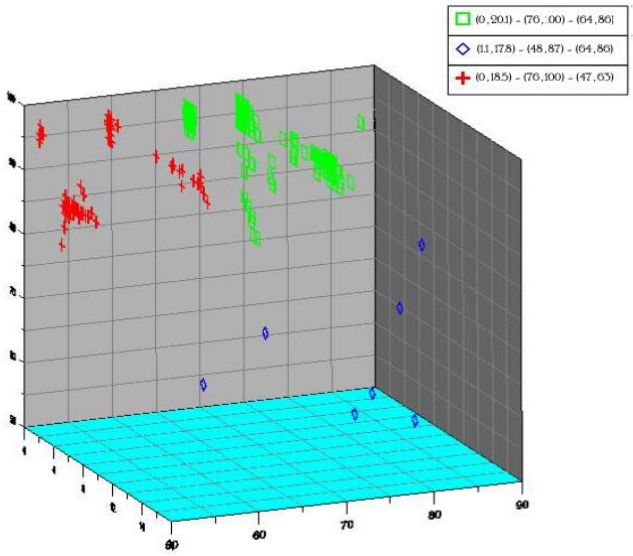
Table 15 – Results for Dataset-10 (Real World Dataset) with 281 Data

Run: 10	Dataset-10			
Data	281		60	
Iteration	1		2	
	Duration	Homogeneity	Duration	Homogeneity
S/G	0.1.512	0.599	0.0.350	0.9972
ISC	0.0.156	0.623	0.0.053	0.9984
ISC2	0.0.298	0.673	0.0.156	0.9987
NLA	0.5.811	0.673	0.0.016	0.9989

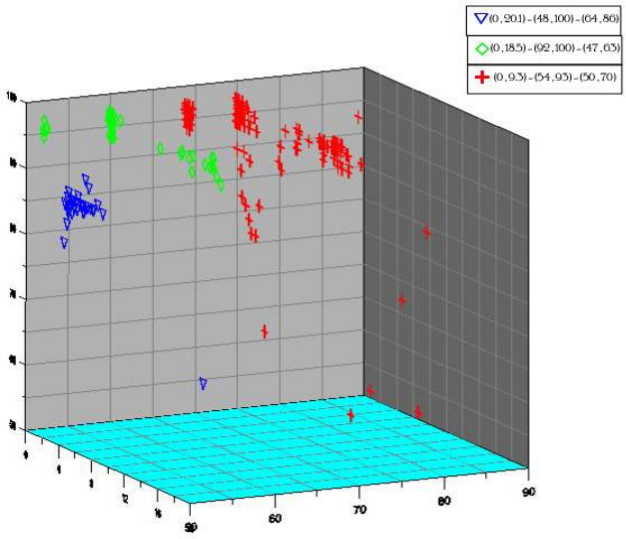
Above table (Table 15) indicates that ISC2 and NLA generated the clusters with the highest homogeneities in compare to other two algorithms, and similar to the previous experiments ISC has the lowest duration in level 1 and NLA has the lowest duration in level 2. The clustering results are displayed in the following 3D figures (Figure 13).



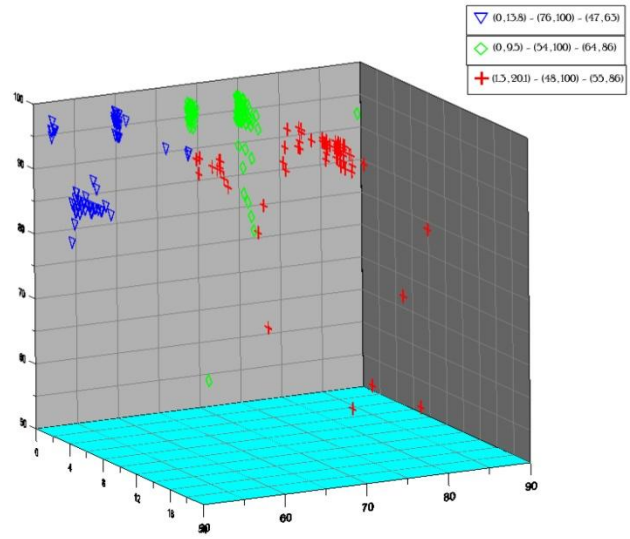
(1) ISC2



(2) NLA



(3) ISC



(4) S/G

Figure 13 – Results for Experiments on Dataset-10

From Figure 13, it is observed that ISC2 and NLA get the similar results. However, their results are quite different from those from the other two algorithms. By looking at the actual data

distribution as shown in the 3D figure, the result from ISC2 and NLA is more accurate to reflect the spatial relationship between those data objects.

4.8. Implementation of the Cluster-Based Browsing Model

In this section, we describe the implemented browsing system, and discuss about its capabilities by showing some snapshots from the user interface.

As mentioned in the previous chapter, the application which is a cluster-based browsing system has the potential to help users select their preferred web services based on the combination of their QoS attributes. The system groups the available services to the desired number of clusters, based on the user selected algorithm, the distance metric, as well as the linkage method (if the related algorithm is selected as S/G). Afterwards, the summary of each cluster is shown to the user. User can select one or more clusters which are considered as the new input data to be clustered in the next iteration. This process is continued until the user satisfaction is achieved. Additionally, user can change the number of clusters in each level of browsing, or get back to the previous level by clicking on the related buttons.

The user interface consists of eight sections: Set Data, Algorithms, Distance Metrics, Linkage Methods, Summaries, Parent clusters, Generated Clusters, and Cluster Details. Set Data section contains the following three parts: Browse, Number of Clusters, and Order of Input Data. When start, user can select the specific file containing the QoS vectors of the available web services, by pressing the browsing button (Figure 14).

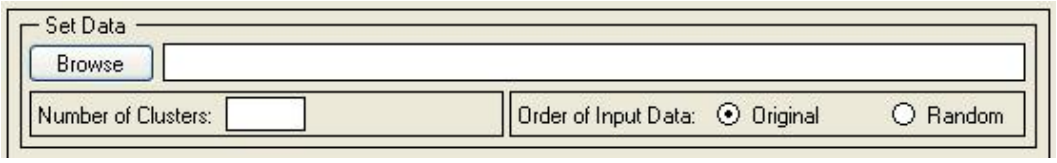


Figure 14 – Set Data Section

When saving the data, the interval's lower bound and upper bound are separated by commas ",", intervals of different QoS attributes are separated with semicolons ";", and each vector is placed on a new line (Figure 15).

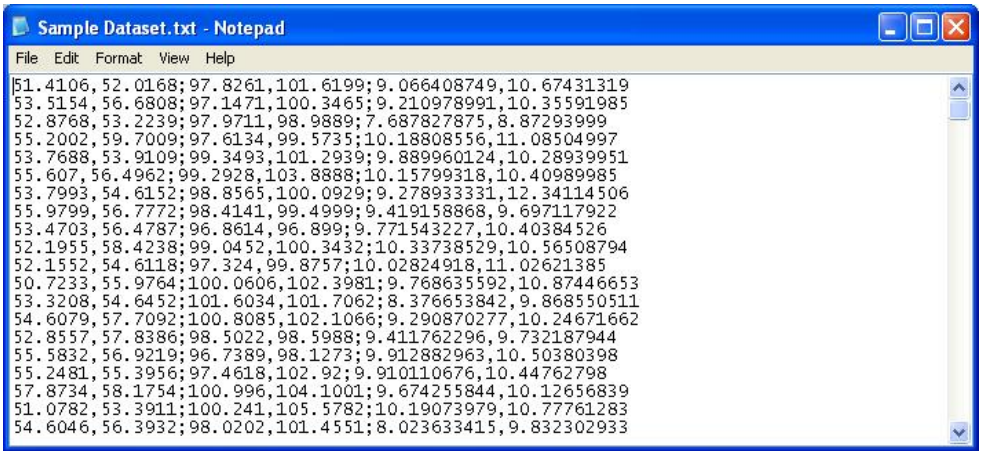


Figure 15 – Sample Dataset

By choosing the specific dataset, the users can define their required number of clusters by typing it in the related text box. Furthermore, the order of data can be changed to random if the algorithm is sensitive to the order of input data.

In the next section, the preferred algorithm is chosen from the list, as well as the specific distance metric which is needed to be used in the clustering process (Figure 16).

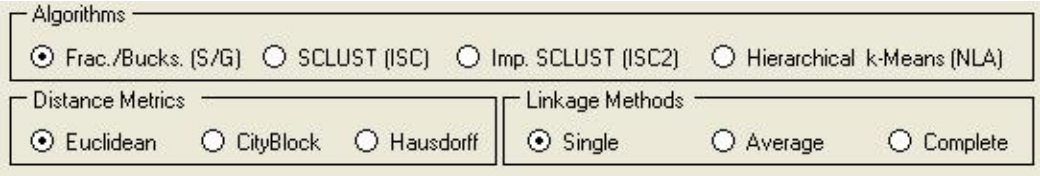


Figure 16 – Algorithms, Distance Metrics, and the Linkage Methods Sections

Depending on the selected technique (whether S/G or the others), the linkage methods used in the original Scatter/Gather technique is chosen from the related panel, which are disabled when other techniques are selected.

The other part is the execution button, containing Scatter/Gather command button, Back button, Run and Export, Reset, Rand Index, and Export Buttons (Figure 17).

Figure 17 – Command Button Part

After defining the address of the specific dataset, the required number of clusters, distance metric, and linkage method, the input data is scattered into different clusters, and their summaries are shown to the user in the related sections, by pressing the Scatter/Gather button, as illustrated in Figure 18. Later, we can clear the results from the user interface and the application's memory by clicking the Reset button.

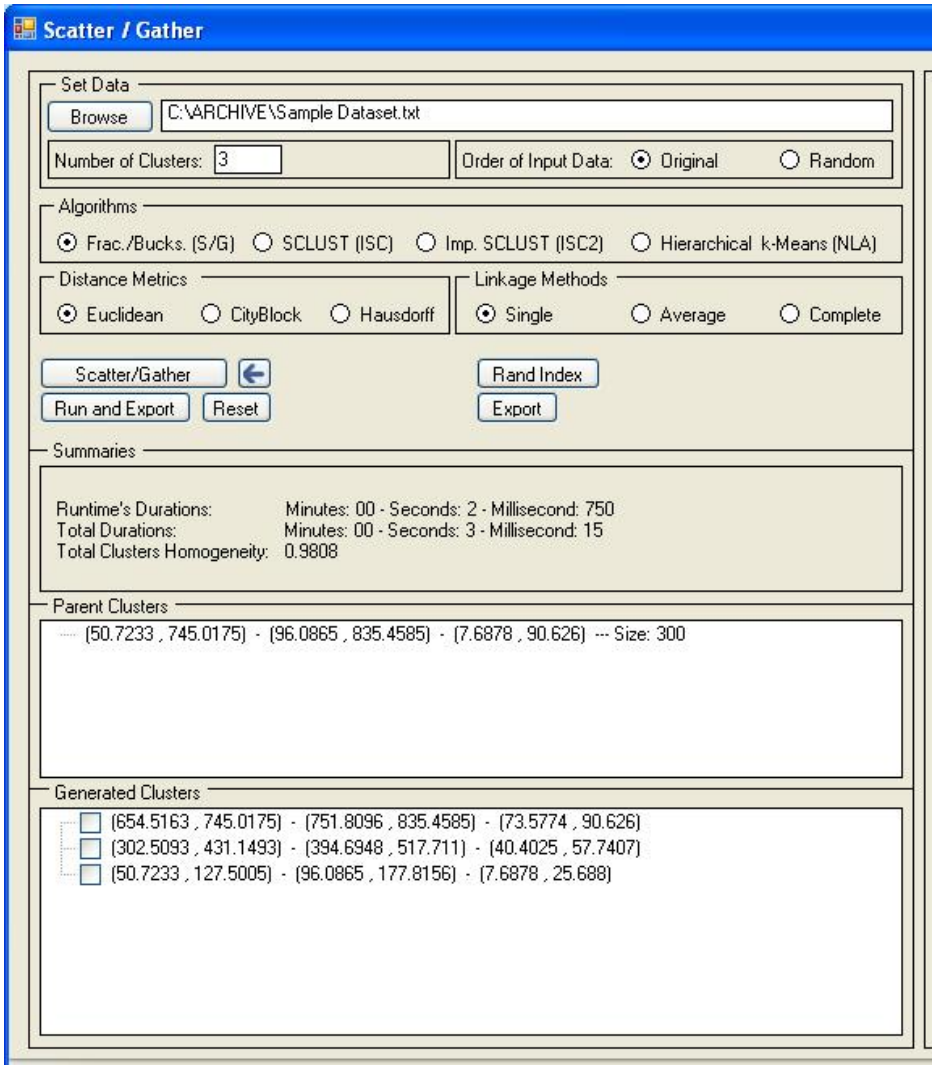


Figure 18 – The Clustering Result of the First Level of Browsing

By pressing the Rand Index button, the system could calculate the rand index value so that it could help us measure the accuracy of the clustering result. When pressing this button another window is opened to get the address of the predefined clustered dataset from user to be compared with the result of the generated clusters by the application (Figure 19). The result which is a real number between 0 and 1 is shown in the summaries section.

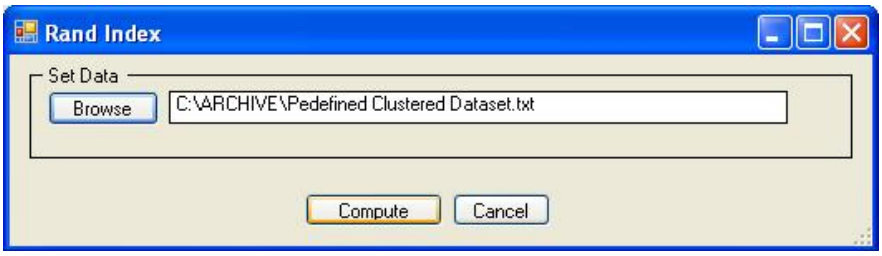


Figure 19 – Rand Index Calculation

The other two buttons (Run and Export, Export) are responsible for exporting the generated clusters to external text files, containing all clustering information such as cluster ranges, cluster representatives, three closest individuals to each centre, cluster homogeneity, processing time, size, and some other information such as name of the technique, distance metric, etc.

The difference between Run and Export, and the Export button is that the former one runs the algorithm a number of times, then exports the average values for all clustering results as well as different text files containing each cluster's data, whereas the latter one only exports those files without executing the program.

By clicking each cluster's summary in the Generated Clusters section, the detailed information is shown in the Clusters Details section. Furthermore, we can see the data of each cluster by double clicking on each of those clusters in the Generated Clusters list (Figure 20).

In order to continue the browsing of the web service information, we can simply select the specific cluster by clicking on the checkboxes beside each cluster's information in the Generated Clusters lists, and press the Scatter/Gather button. Then, the application returns new clustering result based on the latest selection. Each time the user can return to the previous level by clicking on Back button, and repeat the browsing process.

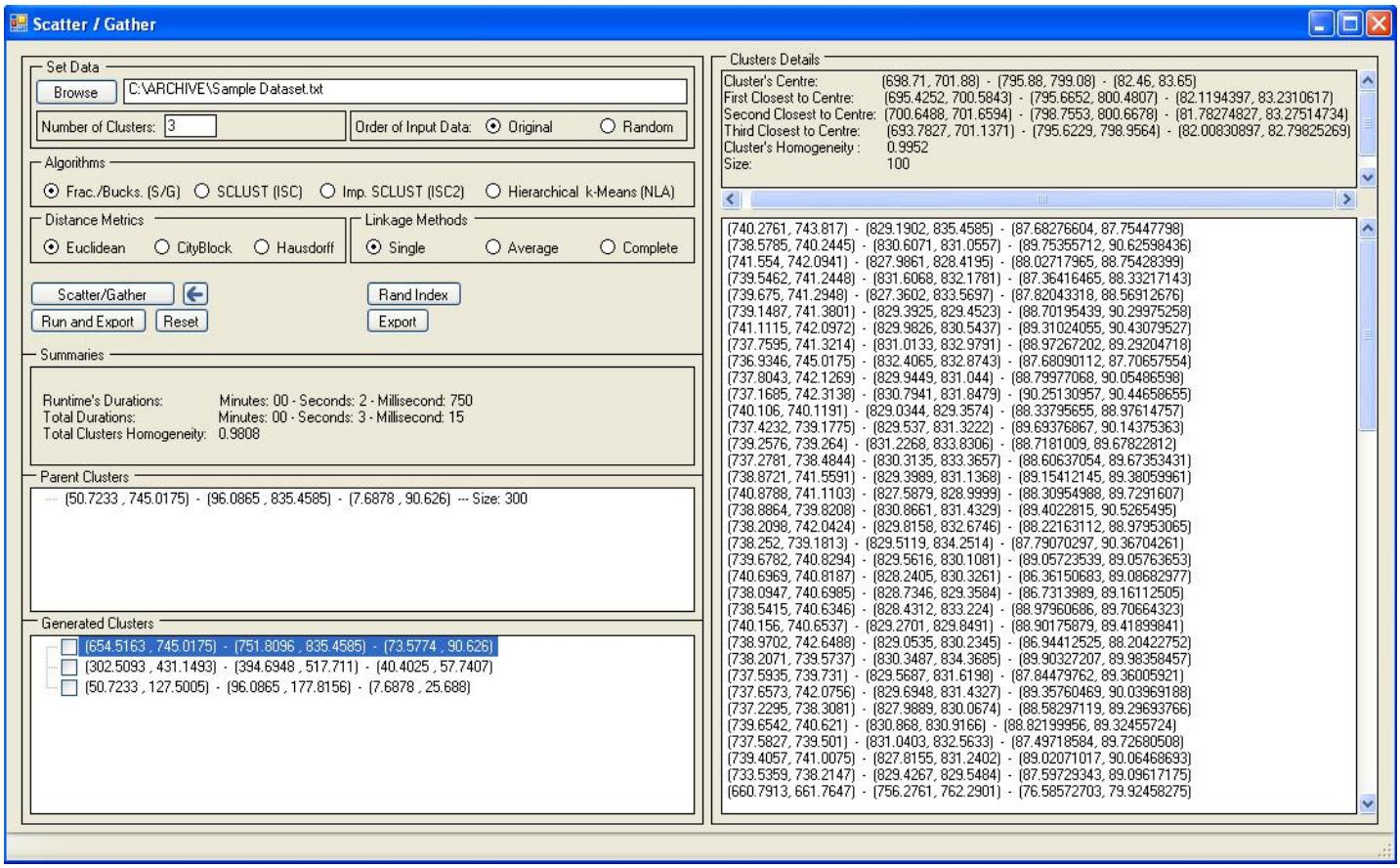


Figure 20 – Showing Cluster Information

Finally, the user can click on the export button, after finding the most preferred web services which are close to the user's need. The specific files are stored in the location which its address is defined in the Set Data section.

4.9. Chapter Summary

In this chapter, we tested our algorithms based on 9 synthetic datasets with different distribution of data, and one real dataset containing the QoS information of the real web services, in two levels of browsing. The datasets were based on randomly generated vectors of interval data which followed a multivariate normal distribution, and contained three predefined clusters

in which each cluster included three sub-clusters. The difference between datasets were in their cluster distances (whether distinct or indistinct), their sizes, complexities, and the number of elements of each data vectors.

We tested our algorithms based on the above mentioned datasets, measured the accuracy of each technique by calculating the cluster homogeneities and their rand indices, computed the processing time and the duration for each algorithm, and compared them to each other. We showed the results on separated tables and plotted some charts trying to find the relationship between the processing time and a few different changes on the dataset. Furthermore, the application is explained by showing the flow of the user interfaces.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

In this thesis, we presented a cluster-based browsing framework for the QoS-aware web service selection consisting of the following algorithms: (1) Original Scatter/Gather (S/G), (2) Iterative SCLUST (ISC), (3) Improved Iterative SCLUST (ISC2), and (4) Improved LAIR2 (NLA). The proposed framework, which is based on the Scatter/Gather browsing model, uses the above mentioned techniques to cluster the vectors of interval data, in which each interval represents a QoS attribute of a web service. In order to compare the proposed methods to one another, we used various datasets which were generated based on different scenarios but all following a multivariate normal distribution. Depending on the scenario, each dataset contains three distinct or indistinct clusters, in which each cluster consists of three other distinct or indistinct sub-clusters. Furthermore, by executing the application a number of times on each dataset, and further calculating the averages of the generated values of the clustering results, such as process duration, cluster homogeneity, and rand index for each technique, certain results were achieved and shown in different tables for comparison.

We compared the accuracy and the efficiency of the clustering results for each technique when dealing with different types and sizes of datasets. The results of the testing based on the original Scatter/Gather showed that the algorithm (S/G) is more suitable for datasets with some distinct clusters, rather than the overlapped datasets. Moreover, by increasing the size of datasets

the processing duration of the algorithm is drastically increased, especially in the first level of browsing or offline phase (which uses the Fractionation method). ISC is the fastest algorithm among our four techniques in the offline phase, but it suffers from getting stuck in a local optimum, which is caused by the convergence of the adequacy criterion; therefore, it generates poor clusters in some of the executions of the technique (or the iterations of the clustering). Hence, in order to increase the accuracy of the ISC algorithm, we applied some modifications on the technique, which include choosing more than *k* initial prototypes and agglomerating them to *k* centroids, and adding a refinement part at the end of the clustering process. The improved ISC algorithm (ISC2) is more accurate and stable than the other techniques (ISC and the original Scatter/Gather), but its processing time is approximately one and a half times more than the ISC. The problem with all those techniques appears in large datasets and in the online phase, when the number of data in the selected clusters is not small. By comparing the improved LAIR2 algorithm (NLA) which used the modified version of ISC to build the hierarchy, to other algorithms in the first level of browsing (offline phase), it was observed that the results are almost similar to the improved ISC, but the duration is higher than that both in the improved and the original ISC, and lower than that of the original Scatter/Gather. In contrast, this algorithm (NLA) is more efficient (e.g. with the processing time of less than a second for 10,000 data) and more accurate than other techniques in the second phase.

5.2 Future Works

For future work, we intend to continue our research in the following three directions:

(1) Employ Fuzzy C-Means Clustering (FCM), as an extension of the ISC algorithm, to discover soft clusters, especially in the first level of browsing (offline phase). By choosing fuzzy

clustering for the dataset containing vectors of intervals, we can overcome the issue with the datasets consisting of the overlapped clusters, due to their symbolic (interval) nature of data. Having similar performance to the popular SCLUST algorithm, we can obtain fuzzy output which can be more sensible for the user to deal with.

(2) Predict the number of initial prototypes which presents the number of clusters in different datasets, depending on the distribution of data. Determining the number of clusters is one of the main issues in different clustering algorithms. In this regard, many techniques have been proposed to overcome this issue, such as grid-based and density-based clustering methods, or even by the calculation of the within-cluster qualities or homogeneities. Therefore, in order to make the application simpler for the user, we can let the program decide how many clusters are more suitable for the specific type of dataset.

(3) Increase the ability of the system to support various value types of QoS properties of the web services, by moving beyond the interval data to more generic symbolic data. As a matter of fact, web services contain different QoS attributes with various value types such as: fuzzy value type, multiple value type (consisting of list, set, range, and vector), and single value type (enumeration, string, numeric, ordinal, nominal, and Boolean). Therefore, in order to make our framework more flexible for dealing with different value types of QoS attributes, in the future, we will provide new techniques to support different types of data.

APPENDIX A: PARAMETERS FOR DATA GENERATION

Table 16 – Dataset-1, 300 three attribute QoS vectors, containing three distinct clusters with three distinct sub-clusters

	Cost ($)		Response Time (ms)		Reliability (%)	
	μ	σ	μ	σ	μ	σ
Cluster 1 (100) (33,33,34)	55	2.0	100	2.0	10	1.0
	80	2.0	140	1.8	17	1.2
	125	2.0	170	3.0	23	0.9
Cluster 2 (100) (33,33,34)	310	3.0	400	3.0	42	0.8
	360	3.6	450	2.4	49	1.3
	420	4.0	510	4.0	56	0.6
Cluster 3 (100) (33,33,34)	660	2.5	760	3.6	77	1.2
	700	3.9	800	3.0	83	1.0
	740	2.0	830	2.0	89	0.9

Table 17 – Dataset-3, 30,000 three attribute QoS vectors, containing three distinct clusters with three distinct sub-clusters

	Cost ($)		Response Time (ms)		Reliability (%)	
	μ	σ	μ	σ	μ	σ
Cluster 1 (10000) (3300,3300,3400)	55	2.0	100	2.0	10	1.0
	80	4	140	1.8	17	1.2
	125	4	170	3.0	23	0.9
Cluster 2 (10000) (3300,3300,3400)	310	3.0	400	3.0	42	0.8
	360	3.6	450	2.4	49	1.3
	420	4.0	510	4.0	56	0.6
Cluster 3 (10000) (3300,3300,3400)	660	2.5	760	3.6	77	1.2
	700	3.8	800	3.0	83	1.0
	740	2.0	830	2.0	89	0.9

Table 18 – Dataset-4, 300 three attribute QoS vectors, containing three distinct clusters with three indistinct sub-clusters

	Cost ($)		Response Time (ms)		Reliability (%)	
	μ	σ	μ	σ	μ	σ
Cluster 1 (100) (33,33,34)	55	2.0	100	2.0	10	1.0
	80	2.0	140	1.8	17	1.2
	125	2.0	170	3.0	23	0.9
Cluster 2 (100) (33,33,34)	310	3.0	400	3.0	42	0.8
	360	3.6	450	2.4	49	1.3
	420	4.0	510	4.0	56	0.6
Cluster 3 (100) (33,33,34)	660	2.5	760	3.6	77	1.2
	700	3.9	800	3.0	83	1.0
	740	2.0	830	2.0	89	0.9

Table 19 – Dataset-6, 30,000 three attribute QoS vectors, containing three distinct clusters with three indistinct sub-clusters

	Cost ($)		Response Time (ms)		Reliability (%)	
	μ	σ	μ	σ	μ	σ
Cluster 1 (10000) (3300,3300,3400)	150	10	100	4	50	2
	170	9	110	5	55	3
	200	12	120	6	57	3
Cluster 2 (10000) (3300,3300,3400)	410	10	250	4	66	2
	430	15	260	4	69	2.5
	460	20	275	7	73	3
Cluster 3 (10000) (3300,3300,3400)	700	12	370	8	80	3
	730	11	385	10	85	2
	750	8	400	10	88	3

Table 20 – Dataset-7, 3000 four attribute QoS vectors, containing three overlapped clusters

	Cost ($)		Response Time (ms)		Reliability (%)		Availability (%)	
	μ	σ	μ	σ	μ	σ	μ	σ
Cluster 1 (3000) (1000,1000, 1000)	50	20	120	30	78	10	80	5
	100	35	150	20	85	7	85	5
	75	28	180	20	90	3	88	3

Table 21 – Dataset-9, 3000 nine attribute QoS vectors, containing three distinct clusters with three distinct sub-clusters

	Cost ($)		Response Time (ms)		Reliability (%)	
	μ	σ	μ	σ	μ	σ
Cluster 1 (10000) (3300,3300,3400)	70	1.8	15	1.8	45	2
	95	1.5	30	1.5	50	1.5
	115	1.5	50	2	55	1
Cluster 2 (10000) (3300,3300,3400)	250	2	250	2	64	1.2
	270	1.5	275	1.8	9	1
	285	1.8	288	2	76	1.2
Cluster 3 (10000) (3300,3300,3400)	340	1.8	100	2	83	1
	365	2	120	1.5	89	0.5
	380	2	140	2	95	1

	Availability (%)		Accessibility (%)		Security (%)	
	μ	σ	μ	σ	μ	σ
Cluster 4 (10000) (3300,3300,3400)	45	1.2	46	1.8	45	2
	52	1	51	1.5	50	1.5
	57	1	57	1	55	1
Cluster 5 (10000) (3300,3300,3400)	66	1.3	67	1.5	64	1.2
	71	1	74	1.2	69	1
	77	1.2	79	1	76	1.2
Cluster 6 (10000) (3300,3300,3400)	84	1.5	87	1.2	83	1
	90	1	92	1	89	0.5
	96	0.8	97	1	95	1

	Compliance (%)		Latency (ms)		Documentation (%)	
	μ	σ	μ	σ	μ	σ
Cluster 7 (10000) (3300,3300,3400)	82	1.2	4	0.5	40	1.5
	89	1.2	10	1.5	46	1.5
	95	1	18	2	52	1.5
Cluster 8 (10000) (3300,3300,3400)	60	1.4	40	1.5	62	1
	66	1	46	1	69	1
	71	1.5	52	1.2	75	1.2
Cluster 9 (10000) (3300,3300,3400)	52	1.2	75	1.2	84	1.1
	45	1	80	0.8	90	1.3
	40	1.3	85	1.2	97	1

References

[1] J. McGovern, O. Sims, A. Jain, M. Little, “Enterprise Service Oriented Architectures: Concepts, Challenges, Recommendations”, Berlin/Heidelberg, DEU: Springer-Verlag, 2006.

[2] S. Pastore, “The service discovery methods issue: A web services UDDI specification framework integrated in a grid environment”, National Institute of Astrophysics (INAF), Vicolo Osservatorio 5, 35142, Padova, Italy, pp. 93 – 107, 2006.

[3] IBM Developer Works, “Web Services Architecture Overview”, 2000. <http://www-106.ibm.com/developerworks/webservices/library/w-ovr/>. (Last retrieved at August 1, 2010)

[4] B.S. Chandhoke, B. Heetland, H. Turner, E. Waitkaitis, “Will UDDI Succeed as the Web Service Description and Discovery Standard”. 2002. <https://drachma.colorado.edu/dspace/handle/123456789/254>. (Last retrieved at August. 1, 2010)

[5] S. Ran, “A Model for Web Services Discovery with QoS”, CSIRO Mathematical and Information Sciences, ACM SIGecom Exchanges, Vol. 4, No. 1, pp. 1 – 10, 2003.

[6] S. Majithia, A. Shaikh Ali, O.F. Rana, D.W. Walker, “Reputation-based Semantic Service Discovery”, 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE’04), pp. 297 – 302, 2004.

[7] C. Ding, P. Sabamoorthy, Y. Tan, “QoS Browsing for Web Service Selection”, In Proceeding of: 7th International Conference on Service Oriented Computing (ICSOC-ServiceWave 2009), LNCS 5900, pp. 285 – 300, 2009.

[8] Y. Wang, J. Vassileva, "Toward trust and reputation based web service selection a survey", International Transactions on Systems Science and Applications, Vol. 3, No. 2, pp. 118 – 132, 2007.

[9] D.R. Cutting, D.R. Karger, J.O. Pedersen, J.W. Tukey, "Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections", In Proceeding of: 15th ACM International Conference on Research and Development in Information Retrieval (SIGIR '92,), pp. 318 – 329, 1992.

[10] V.X. Tran, H. Tsuji, R. Masuda, "A new QoS ontology and its QoS-based ranking algorithm for Web services", Simulation Modeling Practice and Theory, Vol. 17, No. 8, pp. 1378 – 1398, 2009.

[11] E. Diday, M. Noirhomme, "Symbolic Data Analysis and the SODAS Software", Wiley, ISBN: 978-0-470-01883-5, 478 pages, 2008.

[12] M. Chavent, F. de A.T. de Carvalho, Y Lechevallier, R Verde, "New clustering methods for interval data", Computational Statistics, Vol. 21, pp. 211 – 229, 2006.

[13] K. Farsandaj, C. Ding, A. Sadeghian, "A New Approach to Improve the Accuracy of Online Clustering Algorithm Based on Scatter/Gather Model", In Proceeding of: IEEE, North American Fuzzy Information Processing Society's, pp. 1 – 5, 2010.

[14] W. Ke, C.R. Sugimoto, J. Mostafa, "Dynamicity vs. effectiveness: Studing Online Clustering for Scatter/Gather", In Proceeding of: 32th Annual ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, pp. 19 – 26. 2009.

[15] R. Malan, D. Bredemeyer, “Defining non-functional requirements”, Technical report, Bredemeyer Consulting, 2001.

[16] D.Z.G. Garcia, M.B.F. de Toledo, “Quality of Service Management for Web Service Compositions”, In Proceeding of: 11th IEEE International Conference on Computational Science and Engineering, pp. 189 – 196, 2008.

[17] B. Sabata, S. Chatterjee, M. Davis, J.J. Sydir, T.F. Lawrence, “Taxonomy of QoS speci_cations”, In Proceedings of the 3rd Workshop on Object-Oriented Real-Time Dependable Systems (WORDS '97), IEEE Computer Society, pp. 100 – 107, Washington, DC, USA, 1997.

[18] J. Zhou, E. Niemela, P. Savolainen, “An integrated QoS-aware service development and management framework”, In Proceeding of: 6th IEEE/IFIP Conf. on Software Architecture, IEEE Computer Society, pp. 13 – 23, 2007.

[19] K.E. Kritikos, “QoS-based Web Service Description and Discovery”, PHD Thesis, Department of Computer Science, University of Crete, 2008.

[20] T. Rajendran, P. Balasubramanie, “Analysis on the Study of QoS-Aware Web Services Discovery”, Journal of computing, Vol. 1, pp. 119 – 130, 2009.

[21] N. Ahmadi, W. Binder, “Flexible Matching and Ranking of Web Service Advertisements”, In Proceeding of: 2nd Workshop on Middleware for Service Oriented Computing (MW4SOC), ACM, ISBN: 978-1-59593-928-9, pp. 30 – 35, 2007.

[22] D.A. D’Mello, V.S. Ananthanarayana, S. Thilagam, “A QoS Broker Based Architecture for Dynamic Web Service Selection”, In Proceeding of: Second Asia International Conference on Modeling & Simulation (AICMS), pp. 101 – 106, 2008.

[23] J. Yan, J. Piao, "Towards QoS-Based Web Services Discovery", In Proceeding of: International Conference on Service Oriented Computing, ICSOC Workshops, pp. 200 – 210, 2008.

[24] M. Comuzzi, B. Pernici, "A Framework for QoS-Based Web Service Contracting", In Proceeding of: Transactions on the Web, ACM, Vol. 3, No. 3, Article 10, pp. 1 – 52, 2009.

[25] Y. Wang, J. Vassileva, "A Review on Trust and Reputation for Web Service Selection", In Proceeding of: 27th International Conference on Distributed Computing Systems Workshops, pp. 25 – 25, 2007.

[26] M. Chhetri, J. Lin, S. Goh, J. Yan, J.Y. Zhang, R. Kowalczyk, "A coordinated architecture for the agent-based service level agreement negotiation of Web service composition", In Proceeding of: 17th Australian Software Engineering Conference, pp. 90 – 99, 2006.

[27] S. Frolund, J. Koisten, "QML: A Language for Quality of Service Specification", Hewlett-Packard Laboratories, Tech Report: HPL-98-10, 1998.
<http://www.hpl.hp.com/techreports/98/HPL-98-10.html>. (Last retrieved at August 1, 2010)

[28] A. Sahai, A. Durante, V. Machiraju, "Towards Automated SLA Management for Web Services", Software Technology Laboratory, HP Labs Palo Alto, 2001.
<www.hpl.hp.com/techreports/2001/HPL-2001-310R1.pdf>. (Last retrieved at August 1, 2010)

[29] H. Ludwig, A. Keller, A. Dan, R.P. King, R. Franck, "Web Service Level Agreement Language (WSLA) Specification", IBM Corporation, 2003.
<http://www.research.ibm.com/wsla/WSLASpecV1-20030128>. (Last retrieved at August 1, 2010)

[30] V. Tosic, B. Pagurek, K. Patel, "WSOL - Web Service Offerings Language", In the book of Web Services, E-Business, and the Semantic Web, LNCS, Springer-Verlag, pp. 57 – 67, 2003.

[31] T.L. Saaty, L. Vargas, "Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process", RWS Publications, 2000.

[32] Q. Ma, H. Wang, Y. Li, G. Xie, F. Liu, "A Semantic QoS-Aware Discovery Framework for Web Services", In Proceeding of: IEEE International Conference on Web Services, pp. 129 – 136, 2008.

[33] Y. Liu, A.H.H. Ngu, L. Zeng, "QoS Computation and Policing in Dynamic Web Service Selection", In Proceeding of: 13th International World Wide Web, pp. 66 – 73, 2004.

[34] D. Skoutas, D. Sacharidis, A. Simitsis, V. Kantere, T. Sellis, "Top-k Dominant Web Services Under Multi-Criteria Matching", In Proceeding of: 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 898 – 909, 2009.

[35] A.R. Cortés, O. Martín-Díaz, A.D. Toro, M. Toro, "Improving the Automatic Procurement of Web Services Using Constraint Programming", International Journal on Cooperative Information Systems, Vol. 14, No. 4, pp. 439 – 468, 2005.

[36] A.K. Jain, M.N. Murty, P.J. Flynn, "Data clustering: A review", ACM Computing Surveys, Vol. 31, No. 3, pp. 264 – 323, 1999.

[37] J. Han, M. Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann publications, 2006.

[38] P. Andritsos, "Data Clustering Techniques", Qualifying Oral Examination Paper Department of Computer Science, University of Toronto, 2002.

[39] D. Avram Lupşa "Unsupervised Single-Link Hierarchical Clustering", Studia University Babes-Bolyai, Informatica, Vol. 1, No. 2, pp. 11 – 22, 2005.

[40] K.C. Gowda, T.V. Ravi, "Agglomerative clustering of symbolic objects using the concepts of both similarity and dissimilarity", Pattern Recognition Letters, Vol. 16, No. 6, pp. 647 – 652, 1995.

[41] R.M.C.R. de Souza, F. de A.T. de Carvalho, "Clustering of interval data based on city–block distances", Pattern Recognition Letters, Vol.25 No.3, pp. 353 – 365, 2004.

[42] F. de A.T. de Carvalho, R.M.C.R. de Souza, L.X.T. Bezerra, "A dynamical clustering method for symbolic interval data based on a single adaptive Euclidean distance", In Proceeding of: 9th Brazilian Symposium on Neural Networks (SBRN'06), pp. 42 – 47, 2006.

[43] M. Chavent, Y Lechevallier, "Dynamical clustering of interval data: optimization of an adequacy criterion based on Hausdorff distance", Proceedings of: 8th Conference of the International Federation of Classification Societies (IFCS'2002), Springer, pp. 53 – 60, 2002.

[44] F. de A.T. de Carvalho, "Fuzzy clustering algorithms for symbolic interval data based on adaptive and non-adaptive Euclidean distances", In Proceeding of: 9th Brazilian Symposium on Neural Networks (SBRN'06), pp. 60 – 65, 2006.

[45] D.R. Cutting, D.R. Karger, J.O. Pedersen, "Constant Interaction-Time Scatter/Gather Browsing of Very Large Document Collections", In Proceeding of: 16th International Conference on Research and Development in Information Retrieval, ACM SIGIR, pp. 126 – 131, 1993.

[46] C.D.D. Manning, P. Raghavan, H. Schütze, "An Introduction to Information Retrieval", Cambridge University Press, 2009.

[47] Y. Liu, J. Mostafa, W. Ke, "A fast online clustering algorithm for Scatter/Gather browsing", UNC School of Information and Library Science, Chapel Hill, NC, U.S.A., 2007.

[48] W. Ke, J. Mostafa, Y. Liu, "Toward responsive visualization services for Scatter/Gather browsing",In Proceeding of: The Annual Meeting of the American Society for Information Science and Technology (In ASIS&T '08), pp. 1 – 10, 2008.

[49] F. de A.T. de Carvalho, R.M.C.R. de Souza, M. Chavent, Y. Lechevallier, "Adaptive Hausdorff distances and dynamic clustering of symbolic interval data", Pattern Recognition Letters, Vol.27, No.3, pp.167 – 179, 2006.

[50] R.C. Dubes, G. Zeng, "A Test for Spatial Homogeneity in Cluster Analysis", Journal of Classification, Vol. 4, pp. 33 – 56, 1987.

[51] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier, H. Ralambondrainy, "Classification automatique des donneés", Environment statistique et informatique, Dunod, 1989.

[52] K.Y. Yeung, W.L. Ruzzo, "Details of the adjusted rand index and clustering algorithms", supplement to the paper "an experimental study on principal component analysis for clustering gene expression data", Bioinformatics Vol. 17, pp. 763 – 774, 2001.

[53] E. Al-Masri, Q.H. Mahmoud, "Discovering the best web service", (poster) In Proceeding of: 16th International Conference on World Wide Web, pp. 1257 – 1258, 2007. (For QWS Dataset Version 1.0 or QWS Dataset Version 2.0).

[54] E. Al-Masri, Q. H. Mahmoud, "QoS-based Discovery and Ranking of Web Services", In Proceeding of: IEEE 16th International Conference on Computer Communications and Networks, pp. 529 – 534, 2007. (For QWS Dataset Version 1.0 or QWS Dataset Version 2.0).

[55] E. Al-Masri, Q. H. Mahmoud, "Investigating Web Services on the World Wide Web", In Proceeding of: 17th International Conference on World Wide Web, Beijing, pp. 795 – 804, 2008. (For QWS-WSDLs Dataset Version 1.0).