1-1-2009

Interactive QOS browsing for web service selection

Preethy Sambamoorthy
Ryerson University

B1956157X

QA 278 .S36 2009

INTERACTIVE QOS BROWSING FOR WEB SERVICE SELECTION

by

Preethy Sambamoorthy

B.E in Computer Science and Engineering, Anna University, India, 2005

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, 2009

© Preethy Sambamoorthy 2009

AUTHOR'S DECLARATION

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

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

[redacted]

PREETHY SAMBAMOORTHY

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.

[redacted]

PREETHY SAMBAMOORTHY

BORROWER'S PAGE

Ryerson University requires the signatures of all persons using or photocopying this thesis.

Please sign below, and give address and date.

Name	Signature	Address	Date

INTERACTIVE QOS BROWSING FOR WEB SERVICE SELECTION

Preethy Sambamoorthy
Master of Science, Computer Science, 2009
Ryerson University

ABSTRACT

In most of the current research works on Quality of Service (QoS) based web service selection, searching is usually the dominant way to find the desired services. This approach comes with the potential problem of framing search queries properly due to requestor's lack of knowledge or vague requirement about QoS attribute values.

In this thesis, we propose an interactive QoS browsing mechanism that uses the concept of clustering to present the QoS value distribution to requestors followed by finer views of service quality. By analyzing various QoS attributes, we believe that the symbolic interval data is a proper type of representation, compared with the single valued numerical data. Therefore, we use interval data clustering algorithms to implement our browsing system. We conducted experiments on simulated QoS datasets to compare the performance of using different distance measures and show the effectiveness of the interval data clustering algorithm used. The result of the experiments show that the proposed approach provides an effective, user guided QoS based service selection approach that can conceivably overcome the problems with current approaches.

ACKNOWLEDGEMENTS

I am very happy and thankful as I write down this section of my thesis that allows me to express my heartfelt gratitude to all the prime people who helped me come this far. First, I am deeply indebted to my supervisor Dr. Cherie Ding whose guidance and support have made the foundational work for my thesis possible. Her constant support, thoughtful ideas, encouraging words, and the valuable time given to me steered me towards the finish line of this thesis work. She constantly shared with me her never-ending trail of intellectual thoughts. She was always available to meet with me and help me in spite of her very busy schedule. Words are not enough to express my sincere thanks to "Dr. Cherie" in helping me through everything possible to, achieve better.

I would like to sincerely thank the members of my thesis defence committee, Dr. Abdolreza Abhari, Dr. Alex Ferworn and Dr. Isaac Woungang for their valuable time, support and suggestions.

I am profoundly grateful to Dr. J.D.Panar for giving me his valuable time and advice in improving the quality of this thesis document. He meticulously reviewed the document and patiently outlined his suggestions, by taking time for me in spite of his busy schedule.

Now is the time to cherish my memories with Ms. Koteeshwari my primary school teacher who always inspired and showed me that I could do better. I remain obliged to Ms. Latha Parameswaran, my undergraduate studies professor, who helped me enter and pursue this program with enthusiasm.

I am grateful to all the professors at the Department of Computer Science in Ryerson University, for dedicating their time in keeping me inspired and educated.

I am very thankful to my good friends Sonal, Niranchana, Tahira, Fatema, Kristy, Kian, Shilpi, and Veshnu for their constant encouragement and knowledgeable ideas.

Finally and most importantly, I would like to thank my parents Ms. Rajalakshmi and Mr. Sambamoorthy and my little sister Shruthi for always being there for me and supporting me through every walk of my life. It is their dedication, encouragement, selflessness, and prayers that keep me going.

Table of Contents

LIST OF FIGURES

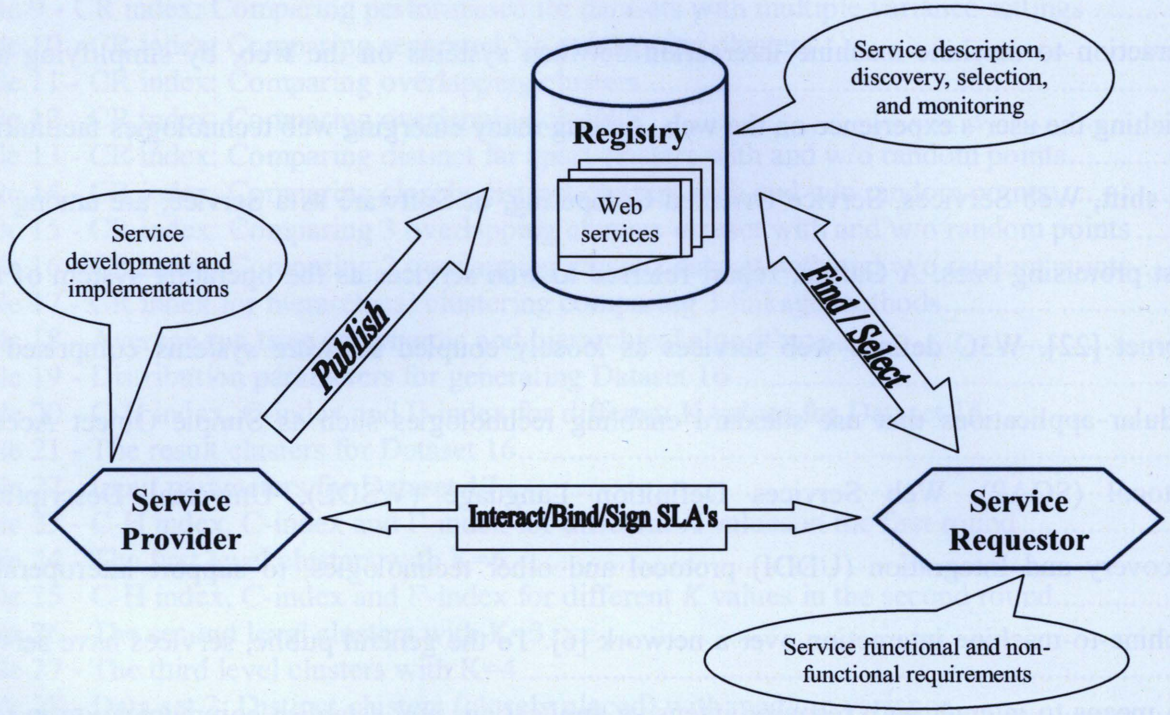
LIST OF TABLES

CHAPTER 1

INTRODUCTION

Today the world stays highly connected with the widespread growth of the World Wide Web. As time passes, the Internet builds on numerous well-adapted computing methods and underlying technologies. The roadmap for this growth suggests a shift from human-machine interaction to machine-machine interaction between systems on the web, by simplifying and enriching the user's experience on the web. Among many emerging web technologies facilitating this shift, Web Services, Service Oriented Computing, or Software as a Service, are among the most promising ones. A Gartner report referred to web services as the operating system of the Internet [22]. W3C defines web services as loosely coupled software systems comprised of modular applications that use standard enabling technologies such as Simple Object Access Protocol (SOAP), Web Services Definition Language (WSDL), Universal Description, Discovery and Integration (UDDI) protocol and other technologies, to support interoperable machine-to-machine interaction over a network [6]. To the general public, services have served as a means to interact with resource intensive applications and establish communication in real time, while to the enterprise community they enable these applications to improve their businesses and their relationships with customers. The structure of web services, mandating the separation of the application interfaces from the actual service implementation gives it the qualities of interoperability, reusability, and dynamicity. The use of standardized technologies in putting together the building blocks of a service draws industry support and ubiquity on the web, making them an appropriate choice to launch applications in a highly distributed heterogeneous environment.

In spite of the fact that the web services technology is a relatively recent development; there has been a rapid growth in the areas of web service development and evolution. However, the basic framework of web services in terms of the entities involved and the interactions made between them has remained the same. The three principal entities are the service provider, service requestor and service registry as given in Figure 1[35].



Service description, discovery, selection, and monitoring
Registry
Web services
Service development and implementations
Publish
Find / Select
Service Provider
Interact/Bind/Sign SLA's
Service Requestor
Service functional and non-functional requirements

Figure 1 - Web Service Framework

The service provider is responsible for implementing the service and providing its description (e.g. WSDL documents) and for publishing the service in a public or privately held registry that functions as a service repository. A typical WSDL document defines a set of message formats, data types, transport protocols, transport serialization formats and one or more network locations for service invocation. This sets the tone for all communications between the provider and a potential requestor. The requestor entity represents a client on the web or within an enterprise, looking for a particular service satisfying a set of functional requirements. Apart from the functional requirements, web services are also specified using their non-functional properties, e.g. Quality of Service (QoS) attributes. The requestors access the service registries with a focus of discovering relevant services that match their requirements [6, 52].

The action of discovering and selecting appropriate services is important to the requestor to be served comprehensively with respect to functionality, quality and the economics of the service in a highly competitive web services market. From the provider's perspective, it guarantees business, provides insight into the requirements of present and future requestors, and presents an opportunity to identify the shortcomings of their current service offerings. Hence, in a web service interaction framework, the process of discovering and selecting appropriate services is important for the success of services on the web.

1.1 Motivation

The current trend in software systems has led to a widespread awareness of web services and their usefulness, resulting in a steady surge in the development, publishing, and usage of web services. The earlier research efforts in the area of service discovery and selection mainly use functionality based matching techniques, ranging from syntactic matching to semantic matching to behaviour matching [19, 42, 62]. With the exponential growth in the number of services, the result sets obtained by matching services based on functionality alone, are large in number and get tedious for the service requestors to subsequently narrow down to the matching services. Hence, the need arises to filter and select services based on their non-functional properties in addition to the functional attributes [49, 60, 64].

Ideally, the service selection process is composed of two steps. In the first step, the requestor uses a set of functional requirements to select services offering matching functionalities. This could be done using search queries consisting of relevant keywords, which describe the expected functionality of the service to be selected. From the results of this operation, the user can submit further queries describing the desired quality attributes for the service. In current frameworks developed for this purpose, the results are either directly returned to the requestor using matching algorithms or returned as a ranked list of services based on predefined rules and ranking algorithms. For both the functional and non-functional requirements matching, the service information given by service descriptions in the form of WSDL documents, parameters in the SOAP messages, and quality attributes information from SLAs' and registries are used. At one point in time, research in this field considered the effectiveness of using this kind of matching systems in automating the service selection processes. This motivated the building of a service ontology that semantically modeled the services and their relationship using semantic mark up languages [4].

Thus, it is seen that there are several methods in current use. With these current methods using QoS selection, forming an accurate query is often a problem due to requestor's lack of knowledge or unclear knowledge about the values of the QoS attributes and their patterns of distribution. The fact that web services are highly dynamic in nature for several reasons further intensifies the requestor's dilemma in selecting the appropriate service using their particular QoS attributes. Some reasons supporting the dynamic nature of service are: new services emerge, old services expire, existing services are modified, the hosting environment is dynamic, resources are dynamically scheduled, hosting systems have different workload, cloud computing separates service hosting from service implementation, etc.

Given the understanding of the significance of web service QoS in selecting services, it is seen that, in reality, values of QoS attributes have mixed data types as opposed to single valued numeric data type. This further draws attention to using specialized algorithms in handling such data while using them during the service selection process.

1.2 Problem Statement

First, in the light of current advancements in enabling efficient web service discovery and selection frameworks, we understand that filtering services based on their functional properties alone are not sufficient and hence it is appropriate to consider their non-functional properties during service selection.

Second, the fundamental problem with the QoS-based selection process is that the current approaches assume that requestors can formulate a QoS query correctly, which is not always true in reality. Requestors may not have the knowledge of, which QoS attributes are measured by the registry, or, more commonly, the exact value ranges of those QoS attributes in registries, which usually leads to an unsuccessful search. This results in the requestors abandoning their search process or selecting services that might not completely match their requirements.

For instance, a service requestor wants to find a service with a high reliability level, and thus they put the request as "reliability>99%", however, none of the services in the registry achieves this level, and the maximum reliability present being 97%. In this case, no matching results could be returned, but when no other choices being available, the requestor could have accepted a service with reliability 97%. This example shows the problem with searching, when an improper query is submitted. In addition, the requestor's QoS requirements could be given by both hard and soft constraints on quality attribute values. The latter case means that requestors have a fuzzy requirement for the QoS values and it is often negotiable. For this kind of QoS requirement, searching on a fixed value is not always a good option.

Finally, we emphasize the necessity of considering the appropriate algorithms for handling QoS data based on their data types.

1.3 Objectives and Contributions of the Thesis

The primary goal of this thesis work is to integrate the inspirations drawn from the concepts of data clustering and interactive browsing in the field of information retrieval in order to improve the QoS based service selection process. We propose a QoS-based clustering mechanism using interval variables that iteratively filters through a collection of functionally similar services yielding results that could potentially benefit service providers, requestors, and registries. More specifically, it benefits requestors in their service selection process. The proposed method reinforces the importance of using non-functional properties of web services, in addition to their functionality, for guiding the service selection process. Unlike keyword based searching techniques, the proposed model presents users with summaries of web services QoS data that can help overcome ambiguities in framing relevant search queries.

By using QoS data and iteratively applying clustering in selecting services, the following objectives could be met and point two is specifically met using the approach developed in this thesis work:

First, web services could be organized into different categories based on their quality levels, in order to improve the service management in a registry.

Secondly, it is easier for requestors to understand and search a small number of service clusters instead of a large data set of functionally similar services; QoS clusters could give requestors an overview of the quality value distribution of this service set, which in turn would help them select the desired services.

Thirdly, clustering could help providers know the QoS delivered by their competitors and understand their own positions in the registry. In addition, more importantly, if the QoS data is collected from the original signed contracts such as the Service Level Agreements (SLA) between requestors and providers, they can provide some idea about the true quality requirements from different requestors, that could enable service providers to constantly improve the delivered QoS of their services.

The contributions made by this work are given as follows.

- We put forth a novel idea, which considers QoS-based service selection as an integrated activity of searching and browsing, by proposing an interactive QoS browsing mechanism for web services.
- We investigate the data types of QoS attributes for web services and conclude that QoS data should be of interval or symbolic type as compared with a single real number proposed by other research activities. We use interval data clustering instead of the traditional clustering algorithms to avoid the loss of information while clustering. We thoroughly analyze, evaluate, and choose appropriate data clustering algorithm between two popular interval data type algorithms. The clustering algorithms are iteratively used to help requestors get more refined and focused views of their interested QoS values in the interactive browsing process.

1.4 Organization of Thesis

The remainder of this thesis is organized as follows:

Chapter 2 - Literature Review: This chapter comprehensively reviews the related research efforts under the area of web service discovery and selection. It discusses various selection methods using the functional and non-functional properties of web services. In addition, it broadly covers the concept of data clustering along with an overview of current approaches used for classic, interval and symbolic type data. Finally, it provides an outline of browsing techniques from information seeking theory that inspired the proposed methodology of this thesis.

Chapter 3 - Interactive QoS Browsing Framework for Service Selection: This chapter begins with the overview of QoS attributes of web services, the fashion in which they are represented and introduces special types of data clustering algorithms. It further explains the two main interval data clustering algorithms used in this thesis work. It also explains the interactive QoS browsing process proposed in this thesis.

Chapter 4 - Experiments and Results: This chapter details the steps for designing the experiments and generating the data used for evaluating the effectiveness of the interval data clustering algorithms used in this thesis. It provides an illustrative example for QoS browsing in web service selection using sample datasets.

Chapter 5 - Conclusion: This chapter concludes by summarizing the work done in this thesis and provides recommendations for the potential future work that can stem from the current contributions.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we study the background material and the related works from four different domains namely: QoS for web services, web service discovery and selection, data clustering, and systems for information seeking on the web, which serve as an inspiration to the proposed methodology presented in this thesis.

2.1 QoS for Web Services

The QoS requirements for web services mainly refer to the quality aspect of a web service in terms of satisfying the needs of a user given by the non-functional requirements of the service [35]. QoS attributes serve as a benchmark in assessing and distinguishing several service providers from one another. Hence, they enable requestors to make informed service selection decisions based on the capabilities of the provider pool. QoS also serves as a key factor in service provisioning, which helps service providers enhance their competitiveness in the web services market place [36].

With the increasing importance of quality of service, there have been several research studies [6, 37, 55] attempting to quantify and categorize QoS attributes into meaningful groups. In [55] the author provides a clear classification of domain specific QoS attributes given as:

- Run time related: Includes scalability, capacity, performance, reliability, availability, robustness, exception handling, and accuracy.
- Transaction support related: Includes integrity, atomicity, consistency, isolation and durability.

- Configuration management and cost related: Includes regulatory, supported standard, stability, cost, and completeness.
- Security related: Includes authentication, authorization, confidentiality, accountability, traceability, data encryption, and non-repudiation.

Meanwhile, another author [65] identifies QoS attributes by type: performance, dependability, security, and application specificity.

Another aspect in the discussion of QoS is in terms of the multiple potential sources in the architectural model of web service communities, which make quality attribute values available. Some popular sources include publisher provided data from service registries, monitoring engines, SLA documents, third parties such as a broker agent, and requestor provided data in the form of feedback [49, 60, 68]. Publishers provide quality attributes along with their guaranteed values in service description documents or define ontology to model these attributes in an automated services framework. This information is found in public registries or other privately held enterprise registries. This serves as a means of advertisement for these publisher entities by increasing their visibility within the service communities. Requestor provided QoS data is in the form of accumulated ratings based on earlier experiences with service usage. This information called service reputation serves as a subjective measure of quality as opposed to objective-quantitative measures such as reliability, availability, etc. The publisher and requestor sources are, however, prone to manipulation resulting in false QoS data. In order to address this issue, earlier research studies proposing and implementing trust and reputation mechanisms evolved; for instance, the inclusion of an extra QoS sub-attribute called verity to indicate the service provider's trustworthiness [30]. SLA documents refer to mutual contracts signed between requestors and publishers upon agreeing to the terms of service with respect to quality. Even though this concept appears to be promising, there has not been widespread adoption for SLA's due to limited efforts taken to ensure their compliance. Monitoring engines and third party agents are registry demanded sources tracking the quantitative performance of services. By combining SLA conformance with monitoring engines, it is possible to compare the requestor promised QoS with the delivered QoS for improving QoS accuracy level.

Yet another aspect of discussing the quality of service is provided by surveying the way QoS attribute values are calculated and handled by repositories of such data. In [46] the authors provide a picture of how the QoS attributes tie to web services along with means to measuring such attribute values. They also details several factors affecting the dynamicity of QoS attribute values and measures such as caching and load balancing in proactively improving these values. There are also dedicated languages to specify QoS attributes, such as Quality-of-service Modeling Language (QML), NoFun language and CORBA trader, which have evolved [17]. As can be seen, over time the quality of service as a differentiating factor for web services has been identified as important for the selection and adoption of a single service among the multitude of web services published.

2.2 Web Service Discovery and Selection

In a short span of time there has been a wide spread proliferation of web services that highlight the promise of software functionality provided by services, these being able to readily aggregate or replace the functionality that caters to the dynamic requirements of requestors. With their increasing growth, the process of service discovery and matching based on the structural and semantic information of services becomes a daunting task. This often leaves the service requestors with an information overload. At present, the area of service discovery and selection is being actively explored and several service search engines [1, 55, 61, 62, 67] and techniques have evolved as a result. An elaborate survey of service selection frameworks in the past and present suggest three main categories of available methods given by:

- Functionality Based Selection Methods
- QoS Based Selection Methods
- Trust and Reputation Based Methods

QoS refers to a set of non-functional properties given by quality attributes of web services, used for assessing services and assisting requestors in finding services focussing on the quality aspects of services aside from their functionalities. Trust refers to a personalized single user subjective score on the performance of a service [65]; reputation on the other hand is the public opinion collected on a service formed by the aggregation of scores from individuals about a particular service [40].

Often similar services are identified based on their functionality by performing a keyword search on service names, their description files, and operation parameters. Advancements in this approach widened its scope from matching services syntactically to matching them semantically, which improves the correctness of the results.

In [19] the authors consider the possibilities of using text, schema, and software component matching techniques to find the similarity between web service operations. They proposed a method using a modified agglomerative clustering algorithm that group names of parameters of web-service operations into semantically meaningful concepts. These concepts are used to determine the similarity of inputs or outputs of web-service operations. Each parameter is considered as a set of concatenated terms that are grouped together based on the co-occurrence of its component terms. The method uses the conditional probabilities of occurrence of terms to form association rules. These rules guide the merge and split operations of the clustering process.

Cohesion (similarity between concepts in the same cluster) and correlation (similarity between concepts in different clusters) are the criteria defining the stopping condition for the algorithm identifying similar services. A combination of parameter name, parameter concept and operation description similarities are used to build an exclusive functionality-based selection method.

In another research work [50], the authors propose a semantic web service-clustering algorithm that can extend the semantic representations of the web services. A combination of ontology languages are used in the description files of the web services to add meaning to the structure of services. Following this a clustering algorithm is applied to group the heterogeneous collection of similar services together. User requirements are then collected in the form of query terms and are matched with the clusters to find suitable services. This information is used to group the similar services [50].

In [62] the authors suggest that the API of public service registries like UDDI should provide the users with a categorical listing of services, which tend to be time and effort consuming in terms of navigating to the relevant services. By supplementing these, API's with a set of similarity-assessment methods applied to WSDL files provides a more automated service-discovery process. Firstly, this technique employs a traditional vector-space model along with information-retrieval methods to extract a collection of distinct words (from WSDL specifications using pre-processing steps), which is compared to the words of the query in natural language. Secondly, a structure-matching algorithm is applied on the resulting collection, and matching scores are computed by calculating the semantic distances between identifiers of WSDLs. The authors clearly define the various identifiers within WSDLs that are used for structural and semantic matching.

In [38] the concept of homogeneous service communities with similar functionalities is suggested. The system automatically queries a search engine like Google to collect description files of services with *wsdl* extensions. The files are then mined to extract the following features: content, context, service host and the service name using a cascaded word analyzer. A 2-pass tree traversing ant algorithm is applied on the extracted terms to find similar services grouped as homogeneous communities.

Subsequently research efforts have tried to solve the dilemma service requestors face, in having to make a choice from a collection of functionally similar services, by evaluating them using the non-functional QoS requirements.

In [17], services are organized as three abstract layers: concrete, abstract, and subject type. Ontology describing these services is built as an extension to the traditional public registries. The service domains and importantly the QoS specifications are treated as subspaces in a multidimensional space. The QoS parameters published by the provider are modeled as point data, while the requestor's parameter specifications are represented as constraints on these points. Subspace clustering techniques identify the published spaces falling within the demand subspace to find matching services. In [32], optimal service selection is achieved through multi-attribute decision theory methods; the declarative logic-based matching rules are specified instead of the hard-coded matching algorithms, and therefore the whole algorithm is more flexible.

The concept of context awareness from the perspective of service providers and requestors can also be used for QoS based service selection [70]. By using identity, location, activity, and time information as context types, context rules are generated. The context rules thus generated filter the relevant services following which QoS attributes are used to generate web services scores. The maximum quality attribute values are used to calculate the objective weight of providers and subjective weight of requestors to derive the weighted score for each service. Services are ranked using relevant matching and scheduling algorithms before presenting them to the requestors.

The concept of trust and reputation has been used to score services based on their conformance to QoS values. For instance, a trust and reputation model is built using three factors: ratings made by users, service quality compliance, and its verity, i.e., the changes of service quality conformance over time to rank services [60]. Another possibility as discussed in [60] uses real-valued time series forecasting techniques to predict the quality conformance values for QoS attributes from past data. User reports on QoS conformance are grouped using a convex k-means clustering algorithm. Following the trust-reputation evaluation, credibility weights are assigned to larger clusters, hence differentiating the smaller scattered clusters as cheaters. A simple additive method is used for QoS ranking in the service selection framework. In [37], a fair and open QoS computation model is proposed and implemented in a service registry. QoS values are normalized and similar qualities are grouped. Then a linear combination with user preference based weights is used to calculate the final QoS value. They also enforce a policing mechanism to prevent the manipulation of QoS values by requestors. The, concept of modeling a separate QoS registry to monitor service performance for conformance and collect user feedbacks was also implemented [10].

Another research paper defines the notion of a service pool. The pool is composed of similar web services from which consumer-centric services are discovered and a virtual provider being the representative candidate service of a cluster satisfying the consumer requirements is selected. A similarity-matching algorithm is used to cluster the functionally similar services and generate a virtual WSDL so that the service pool can be accessed as a web service. A negotiation module is used to match the QoS spectrum (of the virtual pool) with the user QoS. Finally, the virtual provider, on subscription from requestor, retrieves the appropriate service. The advantages of this method include using the semantic information of services, a better yield for the precision of the cluster results and considering hard and soft constraints for specifying requestor's requirements. However this has the drawback that the assumption is made that consumers' QoS requirements are compliant with the virtual pool; it also has a lack of accuracy while transforming the service pool to a virtual WSDL description; further it only uses persistent (steady network conditions) and personalized (downloads service from cache) servers [39].

Service discovery and selection frameworks can also be modeled using trust and reputation ranking of web services in a registry. Service selection models built on trust and reputation can further be studied under the classification criteria: 1. Centralized or decentralized - depending on the architecture of the trust and reputation-evaluating module in the model 2. Person or resource - depending on the nature and role of the entity evaluating the trust and reputation rank of services 3. Global or personalized - depending on whether a small closed community or a much larger geographically spread community is used to evaluate services [69]. Other diverse techniques researched and implemented on service selection and discovery can be referred to from the following references [1, 7, 16, 42, 49, and 48].

Even though, several models have evolved to address the issue of service discovery and selection using the functional and non-functional requirements for services, they often assume that the requestors are informed of the QoS data details and aware of their present distribution trends in registries. In reality, the requestors using these discovery and selection frameworks have at best fuzzy or even no knowledge about QoS attribute values of services. This leads to difficulties in framing input queries that are keys to obtaining the correct results using the current discovery and selection models. In our effort, we address this issue by using clustering techniques to summarize the services' non-functional data and propose interactive QoS browsing to identify services matching the requestor's preferences.

2.3 Data Clustering

The topic of data clustering is associated to our work as we intend to use clustering algorithms in the interactive selection process for summarizing and categorizing services based on their QoS information.

2.3.1 Overview

The concept of clustering is a mature yet actively growing area in the domain of data mining and knowledge discovery. Clustering is an exploratory data analysis process that performs the segregation of given data into groups of homogenous objects. It mines from a given data set; the natural groupings of data objects is based on particular distance and similarity measures between the attributes of these objects. Often, objects to be clustered are represented as a single data point or a vector of quantitative features or numerical values. Clustering is also referred to as an unsupervised form of learning. The essence of a quality-clustering algorithm is to produce clusters of data objects whose inter-cluster similarity is low and intra-cluster similarity is high. In other words, objects in the same cluster are more highly similar to one another than to objects in neighbouring clusters [21, 24]. Apart from being a method in the data analysis part of the data mining process, clustering is also used for data cleansing and outlier detection. Clustering helps identify the dense and sparse regions in a given data set, thus helping find the overall distribution patterns and interesting relationships between data attributes. The concept of clustering has been widely implemented in applications such as pattern recognition, image processing, market research, and weather forecasting to name a few useful ones [24] .

Several different algorithms for clustering have evolved over time, primarily based on the type of data to be clustered along with the application domain and method of implementation.

2.3.2 Classical Data Clustering Algorithms

Clustering techniques are traditionally categorized into two main types, partitioning methods and hierarchical methods. The partitioning type algorithms are a set of methods that divide a given set of *n* data points into *k* different clusters by adopting greedy heuristics such as iterative optimization. The approach starts by randomly choosing a set of *k* initial points as representations for cluster centers and groups the remaining data objects around these *k* points based on specific proximity or similarity measures. This process repeats iteratively by re-assigning the cluster centers and moving the data objects to be around the new centers. The algorithm terminates on convergence of the distance and similarity measure values. The popular k-means algorithm falls under this category. The significant advantages of this class of algorithms include efficiency, good performance for spherically shaped clusters, and scalability to name a few. The primary drawbacks include the need for user input in the form of *k* (number of clusters to produce). They are valid for numeric type data only, and for the other type they require a data standardization method. Several variations of the partitioning type algorithms have evolved such as the ISODATA, k-medoids, spherical k-means, bisecting k-means, SOM, etc [24, 29].

The hierarchical clustering algorithms produce a tree like structure that progressively joins the most similar data at each level of the structure. The topology of the clusters is a binary tree called the dendrogram. The hierarchical process can be either an agglomerative bottom-up strategy or divisive top-down strategy. In agglomerative method, the clustering process starts with each data object as an individual cluster. These clusters are then successively merged together to form new, larger clusters until all of the data objects are in one big cluster. In the case of divisive clustering, you start with a single large cluster, and at each level a splitting operation is performed to break it down into smaller clusters. The resulting dendrogram can be cut at the *(K+1)*th level to obtain the desired number *K* of clusters. A few variations of this class of algorithms include BIRCH, Chameleon, CURE, SOTA etc [24, 43].

Fuzzy clustering [5] is a relatively recent form of a soft computing method that produces less tolerant, non-distinctive and overlapping clusters unlike the hard clustering methods (partition and hierarchical) discussed so far. In fuzzy clustering each data object is allowed to belong to all the clusters with a certain degree of membership given by *u (i,j)*, which represents the membership coefficient of the *i*th object in the *j*th cluster and satisfies the following two constraints where *c* denotes number of cluster and *N* the number of data inputs:

$$\sum_{i=1}^{c} u_{i,j} = 1, \forall j \quad and \quad \sum_{j=1}^{N} u_{i,j} < N, \forall i \quad (1)$$

Fuzzy methods include density based, grid based, neural network based, and evolutionary type (genetic algorithms) methods, to list a few of the significant forms [21, 66]. Hybrid versions of algorithms have also evolved which combine two or more clustering methods to cancel the drawbacks of each method with the advantages of the other methods. In [20] the authors suggest approaches to combine multiple clustering algorithms forming a hybrid algorithm that delivers multiple advantages as a result.

Other topics of interest with respect to clustering algorithms having considerable dynamics in research advancement include: i) Proximity and similarity measures, ii) Cluster validity evaluation, iii) Computational complexities and corresponding solutions, iv) Application areas, etc.

2.3.3 Interval and Symbolic Data Clustering Algorithms

Symbolic Data Analysis (SDA) is a domain in the area of knowledge discovery related to multivariate analysis, pattern recognition, and artificial intelligence. The class of clustering algorithms in this domain are designed to deal with aggregated data such as symbolic and interval type. SDA is a novel way of analyzing multi-valued variables. It can handle different types of variables such as numerical, interval, categorical, enumeration, and modal, in which interval data is the most common approach [18]. For the interval data, interval-clustering algorithms could produce more accurate clustering results than applying traditional clustering algorithms that perform clustering on representative single point values (e.g. midpoints of intervals). Furthermore, by eliminating the need to normalize or convert the data into a standard form, the structure information of the interval data will not be lost [53]. Interval data clustering [8, 9, 10, 11, 53, and 59] has been used in many applications such as census, temperature, medical and geographical, etc data.

In [59], a dynamic clustering algorithm is used for interval data with a two-step relocation process. This involves identification of prototypes representing each cluster by the local optimization of an adequacy function, followed by the allocation of data individuals to the correct clusters using their proximity from the prototypes. The algorithm repeatedly re-identifies new cluster prototypes followed by the re-allocation step until the adequacy function converges. The proximity is measured by two adaptive versions of the city-block distance. The adaptive distance measures associate with each cluster a distance component defined according to the intra-class structure of the cluster. This consistently produces better clustering results when compared with the use of non-adaptive measures [47]. In another paper [11], the dynamic clustering algorithm is used with Hausdorff distance measure and the two-component dissimilarity measure. Several other distance, similarity and dissimilarity measures such as Euclidean, L_2, Mahanabolis, also exist [8].

Other than the partitioning algorithms discussed in the above papers, the hierarchical clustering also can be used for interval data. In [58], an online agglomerative hierarchical clustering algorithm based on the single-linkage method is used to cluster both symbolic and numerical data. In [23], an agglomerative algorithm for symbolic data based on the combined usage of similarity and dissimilarity measures are presented, and these proximity measures are defined based on the position, span, and content of symbolic objects. Another modified form of the hierarchical clustering algorithms produces pyramids in the place of results dendrograms for the given interval data [18]. The pyramid is referred to as a pseudo-hierarchy with overlapping clusters at each level. The results obtained by this algorithm are believed to provide accurate representation of the input sets; however, these algorithms face the issue of scalability for larger datasets. In [12] the author suggests a generalization of the pyramidal algorithm by limiting the number of internal clusters overlapping, to address the issue of scalability.

Another work is an incremental clustering approach on interval data that is implemented using the concept of rough sets to retain the uncertainty part of cluster analysis and it is efficiently computed in minimum time while addressing the issue of scalability in terms of large data sets. The algorithm does not depend on similarity metrics and is hence suitable for symbolic data. The Leader algorithm is one of these kinds, which scans the entire data set in one shot and separates the leaders of cluster representatives. A current data set becomes a new leader if its similarity measure differs from the current cluster leader beyond threshold values. If not, it is assigned to the same cluster. This algorithm uses the modified city-block distance for the dissimilarity function. Instead of taking the absolute value of the lower and upper bound differences, this measure takes their squared value. Each cluster is the interval of the form (min_c, max_c) [45].

In [34] the authors combine the dynamic clustering algorithm with the Kohonen Self Organizing Maps (SOM) algorithm to clustering symbolic data. The SOM algorithm is used to perform the data reduction step for a large symbolic data set. The results are given by micro-clusters, which are further grouped into clusters modelled by symbolic objects using the dynamic algorithm. The combination algorithm was also applied to a large waveform dataset to show the effectiveness of the results.

Other areas of research effort seen in this family of clustering algorithms involve the generation of a Minkowski metric (the standard metric for typical clustering algorithms) for mixed-feature variables to form dendrograms of interval and numeric data using single linkage clustering methods; the use of divisive clustering methods on symbolic data to furnish a hierarchy of symbolic data sets along with the characterization of each cluster in the hierarchy; and the use of weight distribution vectors as cluster prototype, to achieve context aware dynamic clustering on symbolic data [9, 60]. There are also other approaches available for interval clustering using different mechanisms such as genetic algorithm, belief functions, and fuzzy min-max neural networks [18, 33, 58].

2.4 Information Seeking Theory

A survey of techniques and systems in this domain serve as fundamental inspiration in designing the framework of the methodology suggested in this work.

2.4.1 Search and Browse Strategies

With the World Wide Web's (WWW) fast expansion, the number of sources and access points available to acquire any kind of information on the web has been overwhelming. In addition, people accessing these sources to find the most relevant piece of data, often face situations of information overload. The popular means for finding information on the web has primarily been through search engine interfaces such as Google, Vivisimo and other navigation tools (Scatter/Gather systems [14,15]). It is believed that searching and browsing are two separate but complementary techniques that help users find relevant information on the web. However interestingly, some theories suggest that information seeking on the web is an integrated activity of browsing and searching. When users need particular information, searching is a better way of finding information. When users do not have a clear idea about what they are looking for until the available options are presented, or users do not know how to formulate a query properly due to the lack of knowledge on the vocabulary or the corpus, browsing is a better way. Browsing is also better for keeping the relevant context information, which is crucial in some information seeking tasks [13, 45].

In [45] the authors suggest a combined paradigm of search for users on the web that enables them to perform browsing activities at will using hypertext links to web pages in close proximity to the current page being accessed. Such a system supporting operation on the fly is also highly efficient in terms of being interactive. Further advancements suggest the usefulness of building cognitive models for web navigation; these models can be utilized to predict web usability. This can be done based on the perceptual cues from users while searching, so they can make information seeking decisions and to gain an overall understanding of the contents of information collections accessed [54].

2.4.2 Scatter/Gather Browsing Systems

Scatter/Gather systems are a class of navigational tools that use the principle of browsing to help users navigate through a collection of documents. The basic idea of the Scatter/Gather browsing method is that: given a document collection, the system scatters it into a small number of clusters, and generates a summary for each cluster and presents it to the user. The user can then select one or more clusters for further study based on the summaries; the selected clusters are gathered together and the system then applies clustering again to scatter this sub-collection into a small number of clusters and again present these to the user; this process could continue until the individual desired document is identified [14]. The system uses an on-the-fly buckshot algorithm and an offline fractionation algorithm to cluster documents.

Since Scatter/Gather is an interactive browsing process, the efficiency of the clustering algorithms used in this system is extremely important for the success of the system. Subsequently, some research papers discuss ways to improve the efficiency of the Scatter/Gather system, which was already linear in time [15]. Since document collections are typically very large, a linear running time is not satisfactory enough while using an interactive system. In [15, 26] the authors propose to build a hierarchy of the document collection using an offline hierarchical clustering algorithm. From this hierarchy for a given collection of documents, the cluster at any particular level with the maximum number of leaf nodes is selected as a meta document representing this collection. This subset is then summarized to the user following which the partitioning algorithms are applied to the user selection. In [31], the authors suggest an algorithm that produces initial hierarchies as suggested above, followed by the application of a bisecting k-means algorithm to continue with the scatter gather step. This algorithm claims to increase the efficiency, reduce the effort needed by user, and improve the quality of results generated. The authors also investigate the effectiveness of the scatter/gather based document browsing system using a fair and open-ended user survey. Apart from being used for clustering documents, scatter/gather systems have also been implemented in peer-to-peer networks [51].

2.5 Chapter Summary

In this chapter, the background information related to our thesis work was studied. For our discussion the related literature is categorized into four different sections: QoS attributes of web services, web service discovery and selection frameworks, different types of data clustering algorithms and, information seeking theory and related systems. The past and present service selection frameworks based on functionality, QoS data and trust and reputation levels of services are extensively identified. A large number of search and selection techniques to find services are present, which match the functional requirements of requestors. While techniques based on non-functional requirements matching that are relatively recent are studied. However all these techniques are search based matching methods, requiring the requestor's input in the form of queries and hence are all faced with the same fundamental problem of constructing the correct input query using appropriate keyword. Subsequently, the domain of data clustering is introduced, along with a special class of clustering algorithms dealing with symbolic and interval type data. The chapter concludes with a survey of popular concepts under the domain of information seeking by users on the web with a specific discussion on Scatter/Gather systems, which serve as tools for navigation.

CHAPTER 3

INTERACTIVE QOS BROWSING FRAMEWORK FOR SERVICE SELECTION

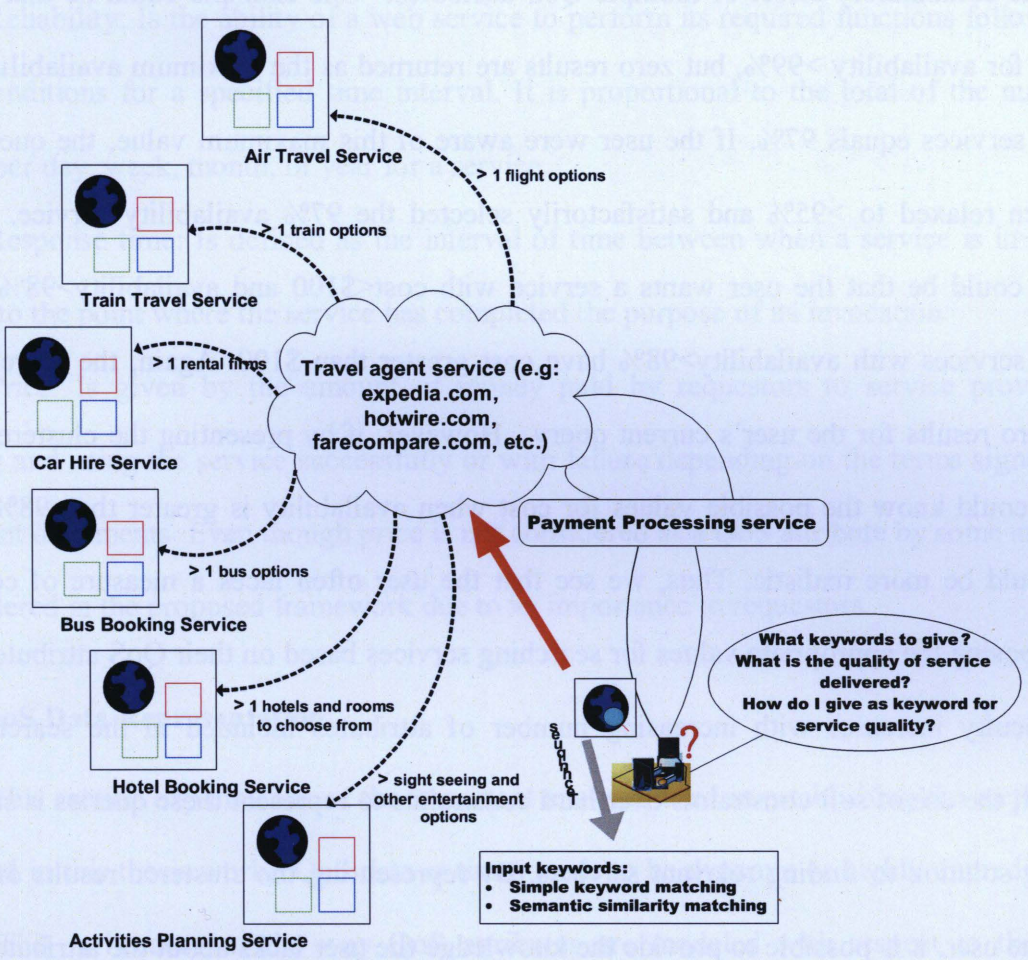
3.1 Overview

The fundamental process of choosing the right services from the vast and diverse pool of functionally similar services can be viewed as a two step process. The first step involves the discovery of relevant services whose descriptions match the functional specifications outlined by the requestor. The second step involves the careful selection of the most appropriate service from the set of services retrieved by the previous step, using non-functional requirements, in which the functionally similar services could be filtered on the hard-constraint QoS requirements, and then ranked based on the soft-constraint QoS requirements. The essence of the work done by this thesis is to introduce and verify a new approach to perform the service selection process in the second step, by using the non functional properties of functionally similar services in conjunction with the application of interval data clustering algorithms. This chapter begins with an exemplar scenario that supports the need for the technique suggested by this work in two ways:

- Establishing the reason for using non-functional service requirements in driving the service selection method
- Indirectly indicating the practicality and ease of the proposed method in carrying out service selection

3.1.1 Motivating Scenario

Consider the example of a user searching for a vacation travel package booking service, to implement on a website as shown in Figure 2 below. Typically, the user looks up a travel agent service from common search engines such as Google, Yahoo, or Bing etc. A simple keyword search, such as 'flight booking' results in several matching hits like expedia.com, hotwire.com, orbitz.com etc.



Air Travel Service
> 1 flight options
Train Travel Service
> 1 train options
Car Hire Service
> 1 rental firms
Travel agent service (e.g: expedia.com, hotwire.com, farecompare.com, etc.)
Bus Booking Service
> 1 bus options
Payment Processing service
Hotel Booking Service
> 1 hotels and rooms to choose from
What keywords to give? What is the quality of service delivered? How do I give as keyword for service quality?
Search
> sight seeing and other entertainment options
Activities Planning Service
Input keywords:
• Simple keyword matching
• Semantic similarity matching

Figure 2 - Example real time travel booking scenario

The user can further narrow down the choice from this list of functionally similar services based on the non-functional properties given as QoS attributes of these services. This could refer to attributes such as the availability of the service, cost of the package provided by the service, credibility of the service in terms of making secure and valid reservations etc. To continue to filter through the services based on the QoS attributes, the user is required to give his preferences to the search engine in the form of hard and soft quality constraints such as (cost of package <1000, availability >99). In the first place, it is possible that the user has no knowledge about the actual ranges for these attributes. Furthermore, even if the user has a partial knowledge about the actual values of the QoS attributes, it is still possible for the user to form an inappropriate query due to the combinatory effect of multiple QoS attributes. One example could be that the user searches for availability >99%, but zero results are returned as the maximum availability of the returned services equals 97%. If the user were aware of this maximum value, the query could have been relaxed to >95% and satisfactorily selected the 97% availability service. Another example could be that the user wants a service with cost<$100 and availability>98%, but all returned services with availability>98% have cost greater than $100. Again, the search would return zero results for the user's current query. However, if by presenting the clustered result, the user could know the possible values for cost when availability is greater than 98%, so the query could be more realistic. Thus, we see that the user often faces a measure of confusion when choosing the appropriate values for searching services based on their QoS attribute values. The difficulty increases with increasing number of attributes included in the search query. Moreover, the use of soft constraints over hard constraints to represent these queries is still not a satisfying solution to finding relevant services. By representing the clustered results of service QoS to the user, it is possible to provide the knowledge the user lacks about the attribute values, which are important for service selection.

3.1.2 QoS Attribute Definitions

In this section, the definitions of three popularly considered QoS attributes [35, 55, 64], used in this thesis work to analyze the proposed QoS based service selection method are presented. The three attributes are reliability, response time and price (or cost). Throughout this thesis, the terms cost and price are used synonymously. Here we provide the definitions of these three QoS attributes:

Reliability: Is the ability of a web service to perform its required functions following the stated conditions for a specified time interval. It is proportional to the total of the number of failures per day, week, month, or year for a service.

Response time: Is defined as the interval of time between when a service is invoked by the user to the point where the service has completed the purpose of its invocation.

Price: Is given by the amount of money paid by requestors to service providers on invoking and using the service successfully or with failure depending on the terms signed in the agreement documents. Even though price is not considered as a QoS attribute by some authors, it is considered in the proposed framework due to its importance to requestors.

3.1.3 QoS Data Representation

This section explains one of the important analyses used in establishing the methodology employed in this thesis work. QoS data at times tends to be dynamic or highly unpredictable in nature. This is because of the way QoS attributes are modeled with respect to the service providers and requestors or in catering to the increasing application domains under which services are published. Under these circumstances, it is important that all aspects and scenarios for describing and representing QoS attributes be considered in order to comprehensively handle the representation dynamics of QoS.

In several QoS based service selection papers [17,32,55], the value of the QoS attributes is assumed to be of single valued integer type as seen in the sample tModel given in Figure 3[67].

```
<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 tModel

We believe that this form of representation is only a simplified means to represent attribute values, which might fail to yield accurate results when used in searching services. For instance, the response time is usually different in different service invocations, and hence an average value in this example can only approximate the actual delivered values. In this case, it would be more useful for requestors' to know the provider-promised upper and lower bound of this value. From the provider's point of view, it is more reasonable for them to publish a value range of the response time instead of an average value to indicate the correct level of service promised by them. This enables providers, to duly standout in a fast growing services market with their accurate levels of quality delivered. Another scenario being that some attributes directly require the publisher to provide a maximum and minimum value while assuring a certain level of service. In the case of reliability, a maximum value for attribute indicates better performance and requires the publishers to provide at least the minimum level for this attribute in satisfying the requestor's requirement. Maximum value can also connect with the capacity of the provider. For instance, due to the limit of the server capacity, the maximum number of concurrent requests a provider is able to handle could be specified. QoS attributes could also have types such as single real data, Boolean, enumeration and others [17, 37]. In [63] the authors define service ontology, modelling QoS attributes for services. They list, integer, Boolean, string as some of the data types used in describing QoS attributes and also show some measurement types used for QoS attributes. Hence, the entire class of QoS attributes could be referred to as symbolic type of data [18].

From the above scenarios, we infer that by representing QoS attributes as an averaged single value, we certainly introduce a level of loss in information with respect to web services QoS. And from the tModel example above we can say that representing QoS attribute values as interval data is easier and is accurate too. In this example, each QoS attribute is measured by a single numerical value. We can easily make some modification as: "price: 0.01" could be converted to "price: (0.01, 0.01)", "average response time: 0.05" could be represented as "average response time: (0, 0.05)", "availability: 99.99" could be represented as "availability: (99.99, 100)", and "throughput: 500" could be represented as "throughput: (500, MAX_THROUGHPUT)" in which the value of MAX_THROUGHPUT depends on the system capacity. Furthermore, if we have a Boolean value such as "transaction support: yes", it could be converted to "transaction support: (1, 1)". It is evident that, there is certainly no room for loss of information in this case and we can handle all types of attributes in this fashion.

Additionally, the web service standards also support to represent the QoS data as interval values. For instance, in the current version of Web Service Level Agreement specification [40], we could define value ranges for a certain SLA parameter as shown in the example given by Figure 4. This capability could be further enhanced by using logic operators.

```
<ServiceLevelObjective name="g1">
<Obliged>provider</Obliged>
<Validity>
  <Start>2001-11-30T14:00:00.000-05:00</Start>
  <End>2001-12-31T14:00:00.000-05:00</End>
</Validity>
<Expression>
  <Predicate xsi:type="wsla:Less">
    <SLAParameter>AverageResponseTime
    </SLAParameter>
    <Value>5</Value>
  </Predicate>
</Expression>
<EvaluationEvent>NewValue</EvaluationEvent>
</ServiceLevelObjective>
```

Figure 4 - Sample WSLA specification

From the above analyses, we believe that interval data would be a more proper way to represent the QoS data. To enforce this concept in service selection, we use special type of interval data clustering algorithms to find similar clusters of services. From our survey of clustering algorithms, we comprehend that there is a trade off in effectiveness and quality of results when traditional clustering algorithms are used in the place of special interval data clustering algorithms to group interval vectors [53]. By using modified interval data clustering algorithms instead of traditional approaches, we also eliminate the need to standardize and further normalize the QoS data. Data standardization techniques result in loss of structure and information of the QoS attributes. The following section discusses the algorithms used for implementing our framework.

3.2 Clustering QoS Data

Firstly, based on the analyses made in the previous section we infer that the QoS data is of interval type rather than being a single valued number, as considered by present QoS based selection methods. On a broader range, certain attributes are also of symbolic nature as discussed.

Secondly, another important fact discussed throughout this thesis is the fundamental problem of the incorrect assumption made by current QoS-based service selection methods. These methods oversee the reality that requestors might often lack or possess fuzzy knowledge about their quality requirements in terms of QoS attribute values. Hence, always coming up with the correct input queries is hard. Combining the above two points, we propose to use appropriate interval data clustering algorithms to group services based on their QoS data to provide summaries of quality information to requestors. Since both partitioning and hierarchical type clustering algorithms are popularly used for interval data, we compare them in our experiments to choose the most suitable type. A set of QoS vectors is given as the input to our clustering algorithms. Each vector includes a set of intervals corresponding to the values of p QoS attributes. Let $QS = \{Q_1, Q_2, \ldots, Q_N\}$ be a set of N QoS vectors described by p interval variables. Each QoS vector Q_i (i = 1, 2, …, N) is represented as $([q_{1s,i}, q_{1e,i}], [q_{2s,i}, q_{2e,i}], \ldots, [q_{ps,i}, q_{pe,i}])$ where $q_{js,i}$ and $q_{je,i}$ (j = 1, 2, …, p) represent the start and end points of interval values for the j^{th} QoS attribute of this quality vector. And $[q_{js,i}, q_{je,i}] \in I = \{[q_s, q_e] : q_s, q_e \in \mathbb{R}, q_s < q_e\}$. In this paper, we choose three QoS attributes to experiment and evaluate our method and so the value of p is 3. The following sections detail the interval data clustering algorithms used along with the interactive service selection technique.

3.2.1 Dynamic Interval Data Clustering Algorithm

Among the different partitioning algorithms available, we propose to choose the dynamic clustering algorithm, which is widely adopted and implemented in several clustering systems using interval data. The dynamic clustering algorithm iteratively relocates the vectors from the clusters of a given data set to locally optimize the adequacy criterion given by the distance measures. The convergence of the algorithm to a stationary value of an adequacy criterion is guaranteed by the optimal fit between the type of representation of the classes by prototypes and the allocation function given by the distance measure. The algorithm is also well known for its ability to globally optimize data using simulated annealing by rerunning the steps of clustering process using different initialization conditions [9, 11, 59].

According to the dynamic clustering algorithm, our method searches for a partition $P = (C_1, C_2, \ldots, C_K)$ of QS in K clusters and a set of cluster prototypes $G = (G_1, G_2, \ldots, G_K)$ which locally optimizes an adequacy criterion $W(P, G)$. The partitioning criterion is defined as

$$W(P,G) = \sum_{k=1}^{K} \sum_{CQ_i \in C_k} D(CQ_i, G_k) \qquad (2)$$

Where $D(CQ_i, G_k)$ is a dissimilarity measure between a QoS vector $CQ_i \in C_k$ and the cluster prototype G_k of C_k.

From the different distance measures for defining $D(CQ_i, G_k)$ we choose to use the two distance measures namely, city block [59] and Hausdorff [9] to calculate the dissimilarity between two QoS vectors. The city block distance is given by the sum of the differences between the upper and lower bounds of the intervals representing each of the QoS attributes present in the input quality vector, whereas the Hausdorff distance is given by the maximum value of the corresponding interval bounds between the two QoS vectors.

The city block distance and the Hausdorff distance are given as,

$$D_{CB}(Q_i, Q_j) = \sum_{h=1}^{p} \left(|q_{hs,i} - q_{hs,j}| + |q_{he,i} - q_{he,j}|\right) \qquad (3)$$

$$D_H(Q_i, Q_j) = \sum_{h=1}^{p} \max\left(|q_{hs,i} - q_{hs,j}|, |q_{he,i} - q_{he,j}|\right) \qquad (4)$$

Where, q_s and q_e refer to the upper and lower bounds of the QoS attribute intervals of the ith and jth quality vectors. Figure 5 shows the steps of the dynamic clustering algorithm.

```
Let $Q_N$ be the $N$ number of QoS vectors to be clustered
    1. Choose $(C_1, C_2, \ldots, C_k)$ partitions in $P^{(t)}$ of $Q_N$ randomly || select $K$ distinct vectors $(G_1, G_2, \ldots, G_K)$ in $Q_N$ AND assign remaining $i$ vectors in $(i = 1, \ldots N)$ around $G_k$ in $(G_1, G_2, \ldots, G_K)$ for $argmin(D_{CB}(Q_i, Gk))$ || $argmin(D_H(Q_i, Gk))$
    2. For $k = 1,2,\ldots, K$; $G_{k*}$ is given as $([gq_{1s,k}, gq_{1e,k}], ([gq_{2s,k}, gq_{2e,k}], \ldots, [gq_{ps,k}, gq_{pe,k}])$ where $gq_{js,k}$ is the median of $\{cq_{js,i}, CQ_i \in C_k\}$ and $gq_{je,k}$ is the median of $\{cq_{je,i}, CQ_i \in C_k\}$ $(j = 1, 2, \ldots, p)$.
    3. Reassign $Q_N$ into $(C_1, C_2, \ldots, C_{k*})$ partitions for $P^{(t)} = min(W(P, G_{k*}))$
    4. If $P^{(t+1)} = P^{(t)}$ then Stop.
```

Figure 5 - Steps in the dynamic clustering algorithm

The first step is for initializing the algorithm by either choosing random partitions in the given dataset or by choosing random input vectors as cluster prototypes. When the latter is chosen, the remaining vectors are clustered around these prototypes using the city block or Hausdorff distance equations. The second step provides representations for the cluster prototypes as median values of the upper and lower bounds of the interval vectors in the partition. The following step is for allocation of the remaining vectors around the cluster prototypes by optimally minimizing the adequacy function. The algorithm terminates on attaining a stable value for the adequacy function.

While initializing with the first case of step 1, representation of each prototype changes when all the vectors have been assigned to the partitions. In the second, it is modified after the assignment of each vector to a new class *Pi* of the partition. In the above steps, the running time of the algorithm can be controlled for the optimal fit of convergence by changing the number of runs and iterations of the first three steps in the algorithm. The algorithm also requires the users input in the form of parameter *K*, referring to the desired number of clusters in the result.

On running the algorithm, the services are clustered according to their QoS attributes and the result *K* clusters are presented at the end.

3.2.2 Hierarchical Interval Data Clustering Algorithm

In the case of hierarchical clustering algorithms, the clusters formed are organized in a tree-like structure called a dendrogram. There are two types of hierarchical algorithms namely, agglomerative and divisive. The agglomerative hierarchical clustering is a bottom up approach to generating the dendrogram, where the algorithm is initialized with each input QoS vector in its own cluster. N, numbers of subsequent merge operations are performed to terminate the algorithm in a single cluster containing all the QoS vectors. There are three ways to merge the clusters at each level. The single-link clustering approach uses the shortest distance (difference between the interval limits) between any members each from the two clusters to be merged. The complete-link clustering method considers the longest distance between members of the two clusters, while the average-link clustering considers the average distance between the members of the two clusters [28]. The divisive algorithm on the contrary is a top down approach to constructing the result dendrogram [24]. We propose to use the agglomerative hierarchical clustering [23, 53] in our experiments. The dissimilarity measure between any two QoS vectors in $QS = \{Q_1, Q_2, \ldots, Q_N\}$ and $(h = 1,2,\ldots,p)$ is given by:

$$Diss_A(Q_i, Q_j) = \sum_{h=1}^{p} (D_{pos}(Q_i, Q_j) + D_s(Q_i, Q_j)) \quad (5)$$

$$D_{pos}(Q_i, Q_j) = cos\left[\left(1 - \left(\left|\frac{q_{he,i} - q_{he,j}}{U_k}\right|\right)\right) \times 90\right] \quad (6)$$

$$D_s(Q_i, Q_j) = cos\left[\left(\frac{(q_{hl,i} - q_{hl,j})}{2} \times ls\right) \times 90\right] \quad (7)$$

$$q_{hl,i} = |q_{hs,i} - q_{he,i}| \quad (8)$$

$$q_{hl,j} = |q_{hs,j} - q_{he,j}| \quad (9)$$

$$ls = |max(q_{hs,i} - q_{hs,j}) - \min(q_{he,i} - q_{he,j})| \quad (10)$$

$$U_k = \max(ls\,(Q_i, Q_j)) \quad (11)$$

The details of the agglomerative hierarchical clustering we used are shown in Figure 6.

1. The algorithm is initialized by assigning each QoS vector in $QS = \{Q_1, Q_2, \ldots, Q_N\}$ to N separate clusters C_n
2. Merge two clusters C_i, C_j containing Q_i, Q_j in C_n for $\min(Diss_A(Q_i, Q_j))$
3. Compute single *OR* average *OR* complete link distance between newly merged and other old clusters.
4. Repeat steps 2 and 3 until $n(C) = 1$

Figure 6 - Basic steps of agglomerative hierarchical clustering algorithm

The results of hierarchical clustering process are in the form of quality vector trees. The desired number of clusters, K can be extracted from the results by cutting the K-1 longest links from the dendrogram.

3.3 Interactive QoS Browsing for Service Selection

This section details the remaining yet important step to our proposed methodology for QoS based service selection. We start by reiterating the problem in current QoS based service selection techniques. These algorithms require the requestor to provide their quality requirements in the form of input queries, which becomes difficult due to the lack of knowledge or vagueness in quality preferences of requestors and the dynamism of publisher provided QoS. Hence, it is possible that often the services returned by the system are either incomplete or incorrect due to improper keywords in input queries.

In addressing this issue, we consider the concepts of searching and browsing under the domain of information seeking on the web. The theory suggests that searching is a better way for users who are looking for a particular information and are familiar with certain aspects of this information. On the other hand, when users do not have a clear idea about what they are looking for until the available options are presented, or if users do not know how to formulate a proper query due to the lack of knowledge on the vocabulary or the corpus, browsing is a better way.

From the arguments and problem statement above, we are able to relate the users of the information seeking on the web interface with web service requestors. Based on this analogy, we propose an interactive QoS browsing mechanism, which could guide requestors in this selection process. Also, pure browsing is not feasible for a big collection such as the entire web, but for a smaller collection, it is an effective information seeking approach, which in fact is the case for our study that considers QoS based service selection from a smaller collection of functionally similar services. Another valid point supporting the interactive aspect of our QoS browsing method is that QoS-based selection usually involves the decision-making on the trade-off among different QoS attributes. Using automatic decision-making algorithms to rank and select services based on QoS might end in adjustments made to quality attributes of significant importance to the requestor in contrary to the assumption of the system. For this purpose, it is more reasonable to include requestors in this process than doing it automatically for them. Thus interactive browsing helps in keeping the relevant context information, which is crucial to requestors.

Having established the purpose of the browsing component of our proposed methodology, we progressed to consider the potential algorithms that can be used to aid in the browsing process. We believe that the principle of clustering algorithms in grouping similar objects from a large collection into smaller meaningful groups is appropriate for summarizing the QoS data of web services. The clustered QoS summaries are then presented to the user. The interactivity with the user could be achieved through letting the user choose a few clusters of his/her interest, followed by re-clustering of the data in the selected clusters. To further effectively fine tune the clustering and browsing process, we use the results of the analysis made in section 3.2 by representing QoS attribute values as intervals and using special kind of interval data clustering algorithms to summarize them.

The browsing framework of our algorithm is inspired by the Scatter/Gather system, used as a navigation tool in document searching frameworks [14]. However, there are some key differences between our framework and the Scatter/Gather system, which will be discussed following the discussion of the steps to our browsing algorithm. The interactive browsing algorithm is shown below in Figure 7.

```
1. Let $N$ = Number of functionally similar services.
2. Let $QS = \{Q_1, Q_2, \ldots, Q_N\}$ be a set of $N$ QoS vectors and $p$ the number of QoS
   attributes in $QS_i$ for $(i=1,2,\ldots N)$.
3. Input $QS$ to the interval clustering algorithm. Present the clustering results of $k$
   clusters to requestors as i) $G_k$ given by $([gq_{1s,k}, gq_{1e,k}], ([gq_{2s,k}, gq_{2e,k}], \ldots, [gq_{ps,k},$
   $gq_{pe,k}])$ prototypes in $(C_1, C_2, \ldots, C_k)$ clusters ii) $n$; the size of cluster $C_j$ for $(j = 1\ldots,$
   $k)$ and iii) Interval $[q_{maxs,i}, q_{maxe,i}]$ for entire partition P($k$) and $(i = 1,2\ldots,p)$.
4. Input condn; condn- requestor's input
5. While ( condn = yes)
6. Request select($k$*); $k$* is the requestor's selection from K clusters based on QoS
   attribute values.
7. Repeat steps 3-4 for $QS_{k*}$
8. End browsing.
```

Figure 7 - Interactive QoS Browsing Algorithm

After carefully implementing both interval clustering algorithms given in section 3.2 for QoS data, we propose to choose the dynamic interval clustering algorithm in our interactive browsing process based on some preliminary experiment results. Above are the steps to our interactive browsing algorithm. As seen from the steps above, our algorithm starts with the supposition that a collection of similar services can be gathered by using algorithms that find all the web services satisfying the functional requirements of requestors. In addition, the QoS data for these services can be collected from the potential sources as indicated in the earlier sections. These QoS vectors are then given as the input to our algorithm, which applies the dynamic interval type clustering algorithm. The algorithm groups similar service QoS vectors into *K* quality clusters of these services. The results of the cluster composition are presented to requestors along with the prototype for each cluster (given by the median of the upper and lower bounds of QoS vectors in the cluster), the size of the cluster, and the range of all QoS attribute values in the cluster given by intervals. With these clusters and their attached information, requestors could have an idea about how the QoS values are distributed within the set. Subsequently, based on requestors' QoS requirements, they could choose one or more clusters among these *K* groups. For instance, considering the quality attribute of reliability, response time and price, cluster 1 could be a collection of services having high reliability, low response time and high price values. While cluster 2 could have high reliability, medium response time and low price values for services. Now, the requestor could either choose cluster 1 if reliability and response time are more important to him over the price or choose cluster 2 if price is a deciding factor in his web service selection process. At this point, the requestors can both make their selection from the results and allow the algorithm to iteratively run the dynamic clustering to produce new service clusters and narrow down to the desired service using browsing. Alternatively, the requestors can using the QoS knowledge gained from the results of initial clustering or any level of clustering results, effectively continue with the search process by formulating a better QoS query.

Considering the useful aspects of our proposed framework, we are faced with a problem of choosing the correct values for the parameters *k* and *k** to our algorithm. There are a couple of possibilities available to handle this situation. We could a) Let requestors choose this value each time b) Fix it to a pre-defined number depending on the size of the dataset c) Or use standard statistical measurements to find an optimal value. After surveying research works addressing this issue for clustering, we propose to choose the third option above [25, 44]. The method tries to find the value for k that optimizes three different statistical indices listed below.

$$\text{C-H index: } (B/(c-1))/(W/(n-c)) \tag{12}$$

$$\text{C-index: } (V - V_{min})/(V_{max} - V_{min}) \tag{13}$$

$$\Gamma\text{-index: } (\Gamma_{+} - \Gamma_{-})/(\Gamma_{+} + \Gamma_{-}) \tag{14}$$

In (12), n is the total number of QoS vectors, and c is the number of clusters in the partition of the data set. B and W denote the total between-cluster sum of squared distances (distance between cluster prototypes) and the total within-cluster sum of squared distances, respectively.

In (13), V is the sum of within-cluster pair-wise distance. Optimal K value is fixed for the best minimal value 0 for *C-index*. This absolute minimum is attained when in a partition the biggest within-cluster dissimilarity is less than the smallest between-cluster dissimilarity.

Equation (14) is the measure that compares the within-cluster and between-cluster pair-wise distances. The comparison is consistent (Γ_{+}) if within-cluster distance is strictly smaller than between-cluster distance and is inconsistent (Γ_{-}) otherwise. The maximum value for the index indicates an optimal K value.

The combination of a greater value for C-H index, a value closer or equal to 0 for C-index and a value closer or equal to 1 for Γ-index corresponds to the optimal K value. We will discuss the remaining details of this method in chapter 4 of this work.

3.3.1 Interactive QoS browsing vs. Scatter/Gather

We complete discussing our proposed methodology by highlighting some key differences between our algorithm and the Scatter/ Gather system as mentioned earlier. Firstly, in

Scatter/Gather system, the item to be clustered is a document, and it is usually represented as a vector of term weights, which are numerical values. Whereas in our system, the clustering unit is a vector of service QoS values, and oftentimes, the QoS attribute is represented as symbolic data, or more commonly interval data. Secondly, the Scatter/Gather method uses partitioning clustering algorithm to form clusters, and in order to find seeds, they use two agglomerative hierarchical clustering algorithms: one is Buckshot, which is faster and used in the real-time clustering, and the other is Fractionation, which is more effective and used in initial offline clustering. In our system, we use the dynamic interval clustering algorithm in both the initial offline and the later iterative on-the-fly step. The seed points or cluster prototypes are chosen randomly. It is more efficient than using the hierarchical clustering algorithm to choose the seed points as in Scatter/Gather, at the same time the effectiveness is not sacrificed according to our experiment results. Thirdly, the number of seeds in Scatter/Gather is a randomly chosen small number, whereas in our system, an optimal *K* could be identified by optimizing some statistical indices [14, 15].

3.4 Chapter Summary

The framework for a novel QoS based service selection algorithm was introduced in this chapter. The problem with current QoS based service selection frameworks is revisited with the help of a web services usage scenario in real time. The ideal case for representing QoS attribute values of services is identified as interval data type in contrary to the commonly used single valued numeric type. Following this, the idea of using clustering algorithms in the proposed interactive QoS browsing process is introduced. Based on these analyses, two popular interval data clustering algorithms were discussed along with the corresponding similarity and distance measure used in these algorithms. The concepts of searching and browsing in the information seeking domain are discussed and compared. Linking the observations of this discussion with the service selection problem, an interactive QoS browsing algorithm for service selection was introduced. The steps of the interactive browsing algorithm are clearly outlined. Finishing this chapter, the proposed approach was compared to a popular system used in document clustering and selection frameworks.

CHAPTER 4
EXPERIMENTAL EVALUATION AND RESULTS

4.1 Data Generation

Before setting up the steps to our experiment, we searched through academic sources [43] of benchmark datasets, to find standard web services QoS data, to evaluate our proposed methodology. However, there are no popular datasets of this kind available in the best of our knowledge. For this reason, we use simulated datasets to conduct our experiments on.

Prior to proceeding with the data simulation, we carried out two analysis steps. We referred to related research works based on QoS monitoring and collection that provide quality metrics data [49]. This data gives us a picture of the limits and ranges for QoS metric values of services in real applications. Conversely, this dataset does not contain quality information for functionally categorized services, but corresponds to a random collection of services. Hence, it is not suitable to directly use this data for testing our service selection approach. We also referred to publicly available information on web service pricing [2, 56]. In parallel to this step, we studied the distribution patterns of simulated datasets used by interval data clustering algorithms to test their efficiency and effectiveness [11, 59]. Based on the understanding and conclusion made by the above steps, we propose to simulate a set of datasets comprising the QoS vectors for services to be clustered. As mentioned in Chapter 3 of this thesis, we consider three popular QoS attributes for web services namely, reliability, response time and price. We believe that this combination covers a typical requirement for quality from requestors. Reliability is expressed as a percentage value (on scale of 100), response time is given in milliseconds (ms), and price is expressed in dollars ($). Another important factor considered during data simulation, is the possible combination of the three QoS attributes at varied value levels (such as high, medium and low). For instance, one dataset might consist of QoS vectors with low reliability, high response time and low price, while another has vectors with high reliability, low response time and higher price. Considering all these factors, we proceed with the data generation step as explained in the following section.

Using the above simulation principles, we proceeded to generate a total of seventeen data sets, and each data set has three clusters. The steps to the interval type QoS vector simulation are twofold. We use MATLAB functions to generate *N* number of seed points for each dataset following a multivariate normal distribution with the independent components, using mean vectors (μ) and covariance matrices (σ) as shown below.

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} \qquad \sum_{11} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix} \tag{15}$$

These seed points are then used to compute the interval vectors using the equation:

$$\left(\left[a - {\gamma_1}/{2},\ a + {\gamma_1}/{2}\right], \left[b - {\gamma_2}/{2},\ b + {\gamma_2}/{2}\right], \left[c - {\gamma_3}/{2},\ c + {\gamma_3}/{2}\right]\right) \tag{16}$$

The variables γ_1, γ_2 and γ_3 are values randomly drawn from predefined intervals and *a*, *b* and *c* refer to the three attributes of the seed point vectors [9, 59]. The input parameters corresponding to the datasets generated are given in section 4.1.2. The SODAS package [57] is used to implement the clustering algorithms on the generated datasets.

4.1.1 Simulation Scenarios

We use the four sets of predefined intervals *a-d* given below while generating our simulated datasets:

a: [1,4] [1,8] [1,8]

b: [1,8] [1,16] [1,16]

c: [1,12] [1,24] [1,24]

d: [1,16] [1,32] [1,32]

The following cases outline the distribution pattern of datasets with respect to the fashion in which they are generated.

A. Separated clusters:

Dataset 1: Distinct far-apart clusters generated with large difference in mean values and small variance using four sets of pre-defined intervals *a-d*

Dataset 2: Distinct closely placed clusters generated with small difference in mean values and medium variance using four sets of pre-defined intervals *a-d*

Dataset 3: Distinct closely placed clusters generated with small difference in mean values and large variance using four sets of pre-defined intervals *a-d*

Dataset 4: Distinct closely placed clusters generated with cluster 1 having maximum variance along reliability, cluster 2 having maximum variance along time and cluster 3 having maximum variance along price has maximum variance value. Four sets of pre-defined intervals *a-d* are used

B. Overlapping clusters:

Dataset 5: All the clusters, overlapping across all three attributes generated with small difference in mean values and large variance using four sets of pre-defined intervals *a-d*

Datasets 6: Two out of three clusters overlapping across all 3 attributes with very small difference in mean values and medium variance using four sets of pre-defined intervals *a-d*

Dataset 7: All the clusters, overlapping across two attributes generated with very small difference in mean values and large variance using four sets of pre-defined intervals *a-d*

Dataset 8: All the clusters, overlapping across all three attributes generated with very small difference in mean values and large variance using one set of pre-defined intervals *a*

Dataset 9: All the clusters, overlapping across reliability and time attributes generated with very small difference in mean values and large variance using one set of pre-defined intervals *a*

Dataset 10: All the clusters, overlapping across time and price attributes generated with very small difference in mean values and large variance using one set of pre-defined intervals *a*

Dataset 11: All the clusters, overlapping across price and reliability attributes generated with very small difference in mean values and large variance using one set of pre-defined intervals *a*

(Note: The mean values are fixed in Datasets 2 & 3 and variance values are fixed in Datasets 8-11.)

C. *Separated clusters with random data:*

Datasets 12: Addition of random data vectors to *datasets 1*

Datasets 13: - Addition of random data vectors to *datasets 4*

D. *Overlapping clusters with random data:*

Datasets 14: Addition of random data vectors to *datasets 5*

Datasets 15: Addition of random data vectors to *datasets 6*

When we generate the datasets, we considered different combinations of the three attributes – reliability, response time, and price, such as *"Medium reliability, low response time, and medium price", "High reliability, high response time, high price", "Low reliability, medium response time, and low price", etc*. Here, high reliability refers to a large value for the reliability attribute (large = 91-100; medium = 76-90; small = 60-75) and vice versa. Low response time refers to large value for response time (large >= 700; medium = 301-699; small = 1-300) and vice versa. High price refers to a large value for the price (large >= 121; medium = 86-120; small = 1-300) attribute and vice versa. In addition, it is ensured that the interval limits for the attribute values fall within a valid number range. This is prepared following the earlier analyses made on QoS metrics data available from the web. From the discussed cases, we can observe that an number of datasets considering possible distribution patterns are generated with distinct, highly distinct, overlapping, greatly overlapping, densely, sparsely distributed QoS data vectors.

4.1.2 Input Parameters

In this section, we present the input parameters of three sample datasets selected from the list in the previous section. For the input parameters for all data sets, please refer to Appendix A.

Table 1 - Input parameters for Dataset1 with distinct clusters without random points

Input parameters	

Data set 1: distinct clusters (far apart)
Total # of points = 450
Predefined interval sets *a-d*
	Group 1 (# of points = 150):
	$\mu_1 = 167, \mu_2 = 1400, \mu_3 = 220, \sigma_1^2 = 0.25, \sigma_2^2 = 9, \sigma_3^2 = 4$
	Group 2 (# of points = 200):
	$\mu_1 = 186, \mu_2 = 140, \mu_3 = 300, \sigma_1^2 = 0.5, \sigma_2^2 = 9, \sigma_3^2 = 4$
	Group 3 (# of points = 100):
	$\mu_1 = 138, \mu_2 = 690, \mu_3 = 120, \sigma_1^2 = 2, \sigma_2^2 = 4, \sigma_3^2 = 2$

Table 2 - Input parameters for Dataset5 with overlapping clusters without random points

Input parameters	

Data set 5: overlapping clusters –all 3 are overlapping
Total # of points = 300
Predefined interval sets *a-d*
	Group 1(# of points = 100):
	$\mu_1 = 172, \mu_2 = 1200, \mu_3 = 220, \sigma_1^2 = 36, \sigma_2^2 = 100, \sigma_3^2 = 49$
	Group 2(# of points = 100):
	$\mu_1 = 175, \mu_2 = 1207, \mu_3 = 225, \sigma_1^2 = 25, \sigma_2^2 = 144, \sigma_3^2 = 64$
	Group 3(# of points = 100):
	$\mu_1 = 178, \mu_2 = 1217, \mu_3 = 223, \sigma_1^2 = 16, \sigma_2^2 = 196, \sigma_3^2 = 36$

Table 3 - Input parameters for Dataset1 with distinct clusters with random points

	Input parameters

Data set 12: distinct clusters (far apart) with random points
Total # of points = 500
	Group 1 (# of points = 150):
	$\mu_1 = 167, \mu_2 = 1400, \mu_3 = 220, \sigma_1^2 = 0.25, \sigma_2^2 = 9, \sigma_3^2 = 4$
	Group 2 (# of points = 200):
	$\mu_1 = 186, \mu_2 = 140, \mu_3 = 300, \sigma_1^2 = 0.5, \sigma_2^2 = 9, \sigma_3^2 = 4$
	Group 3 (# of points = 100):
	$\mu_1 = 138, \mu_2 = 690, \mu_3 = 120, \sigma_1^2 = 2, \sigma_2^2 = 4, \sigma_3^2 = 2$

Random set 1 (50) for dataset12 with pre-defined interval set *a*	[(66-93), (69-95)], [(63-702), (67-707)], [(40-152), (58-156)]
Random set 2 (50) for dataset12 with pre-defined interval set *b*	[(64-93), (68-97)], [(59-702), (67-710)], [(51-151), (60-160)]
Random set 3 (50) for dataset12 with pre-defined interval set *c*	[(60-93), (68-100)], [(56-703), (68-715)], [(45-152), (60-165)]
Random set 4 (50) for dataset12 with pre-defined interval set *d*	[(55-95), (65-100)], [(50-705), (65-725)], [(40-155), (55-170)]

4.2 Experiment Design

There are two main purposes of the experiments conducted as part of this work, given as follows:

- To study the effectiveness of the two clustering algorithms used in our proposed framework
- Evaluate the feasibility of our QoS based service selection framework

The dynamic and hierarchical clustering algorithms are first implemented using the Sodas package [57]. The two algorithms are individually executed on simulated datasets for evaluation.

The city block and Hausdorff distance measures given by equations (3) and (4) are used separately for initializing the dynamic clustering algorithms twice for each dataset. Every run of the dynamic clustering algorithms is initialized with input values *k*, 50, and 50, which correspond to the desired number of clusters generated, number of runs, and the number of iteration parameters of the algorithm. The number of runs parameter is used to control the number of times the algorithms is executed until it finds an optimal solution for the given dataset. The number of iterations parameter is used to prevent the algorithm from getting into an endless loop on repeating to find an optimal solution. All the datasets are tested for the dynamic clustering algorithms.

The hierarchical clustering algorithm is executed by first computing the dissimilarity measure given by equation (5) between all the QoS vectors in each dataset. Following this, the single link, complete link and average link measure are used separately to build the corresponding result dendrograms for four representative datasets.

The observations and inference made by evaluating the results of the two clustering algorithms are used to pick one clustering algorithm that would be used to implement the interactive QoS browsing method, supplemented with the optimal *k* finding method. This step of the experiment is discussed in Section 4.4.

4.3 Evaluation

In this section, we discuss the steps and methods used to evaluate the results of the two clustering algorithms along with results of the optimal *k* finding method for dynamic clustering algorithm.

4.3.1 Proof of effectiveness of clustering algorithms

The results of clustering for any two algorithms can be compared and evaluated statistically using external and internal criteria and alternatively using relative criteria [21]. In this paper, we statistically assess the quality of the clustering algorithms using the Corrected rand index (CR) given by (17).

It is an external measure because it compares the clusters produced in an a priori classification with the results of the clustering algorithm or compares the results of two separate clustering algorithms using the same steps of calculation. The a priori classification in our case refers to the partition in the seed points data generated, which equals k=3 for our algorithms [27, 69].

$$CR = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i\binom{n_i}{2}\sum_j\binom{n_j}{2}\right]\Big/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{n_i}{2} + \sum_j\binom{n_j}{2}\right] - \left[\sum_i\binom{n_i}{2}\sum_j\binom{n_j}{2}\right]\Big/\binom{n}{2}} \qquad (17)$$

Let $U = \{u_1, u_2,..u_r\}$ and $V = \{v_1, v_2,...v_k\}$ represent the set of clusters produced by the a priori classification and clustering algorithm respectively. Let $0<i<k$ and $0<j<r$, then n_{ij} represent the number of vectors that fall under the same cluster in both U and V, n_i represents the number

of vectors under clusters V and n_j the number of vectors under clusters U. N represents the total number of vectors in the data set.

The CR index is a good choice of assessment because it is insensitive to the number of clusters in a given partition and to the distribution of the data vectors within a cluster. The index value ranges from [-1,1], with values closer to 1 indicating the correctness of the clustering algorithm results and values closer to -1 indicating a lower level of agreement between the results of clustering and prior classification. The index values are also used to compare and choose the ideal distance measure of similarity for the dynamic clustering algorithms. In the case of hierarchical agglomerative clustering, it is used to identify the most suitable link clustering method from the three experimented types. We also use them to compare the results of the dynamic and hierarchical clustering algorithms on our simulated datasets for choosing the algorithm for our browsing step.

4.3.2 Finding optimal K

We also tested the feasibility of using the optimal K finding method with our QoS datasets to help users fix this input parameter value for the dynamic clustering algorithm. Based on the results of applying the method, the optimal K value found is always 3, which matches with the actual value for our a priori partition. Therefore, we verify the viability of using this method to find optimal K during the interactive QoS browsing process. Here we use the data set 2 corresponding to a dataset with distinct service clusters and dataset 5 representing overlapping clusters, to show the steps to find optimal K.

In order to find the optimal K, we measure the three statistical indices given by equations (12-14) when $0 < K < 11$, for datasets 2 and 5 and the corresponding results are shown in Table 4 and 5. The ideal case is to find a K value which is consistently the best for all three indices. On the event of a conflict for the best value of *K* for different indices, we try to find the corresponding *K* which performs the best for two indices, or the next optimal option is a *K* which has a more obvious advantage on one index than the other two. As seen in Table 4, we could achieve the best performance on all three indices when *K* equals to 3. So we directly fix the optimal value of *K* to 3 for dataset 2 .

Table 4 - C-H index, C-index and Γ-index for different *K* values for data set 2

K	C-H index	C-index	Γ-index
10	190.02486	0.02491	0.86263
9	207.87110	0.02218	0.88733
8	228.79019	0.02338	0.88668
7	244.57205	0.02643	0.88775
6	277.00010	0.01665	0.95142
5	323.72426	0.01775	0.95273
4	369.84917	0.02136	0.95519
3	*490.17645*	*0.01465*	*0.96936*
2	323.73771	0.08659	0.89706

In Table 5, when *K* equals 3, it performs the best for C-H index and Γ-index although it is not the best for C-index. In this situation, we can still fix the optimal value to 3 for dataset 5 according to the earlier mentioned rules. Thus, we observe that, the *K* fixing method is suitable for datasets with both distinct and overlapping clusters that typically represent the possible distribution patterns of data.

Table 5 - C-H index, C-index and Γ-index for different *K* values for data set 5

K	C-H index	C-index	Γ-index
10	149.60961	0.03070	0.83570
9	162.67945	0.03016	0.86421
8	176.85918	0.03172	0.87391
7	197.98333	0.02987	0.89173
6	219.33315	0.03287	0.88491
5	245.63578	0.03942	0.88619
4	276.81127	0.04016	0.88887
3	*358.48577*	*0.03638*	*0.91364*
2	339.90349	0.09023	0.83824

4.3.3 Analysis and discussion

The scheme of our experimental evaluation step is to analyse a list of items before selecting the appropriate clustering algorithms for our interactive QoS browsing process. The list of items includes, the evaluation of similarity and distance measures to be used, comparing the performance of the clustering algorithms and cover the possible patterns in QoS data distribution while running the experiments on simulated datasets. We decided to experiment with and use the dynamic partitioning type and hierarchical clustering algorithms in this step for the following reasons: These algorithms are considered as two classical algorithms in the domain of data clustering which form the basis of several other clustering approaches. They have been widely applied in different domains for performing cluster analysis and duly evaluated for various properties including efficiency and effectiveness. The next reason being, these algorithms are widely implemented for clustering systems dealing with symbolic data including interval type and found successful. In addition, from the results of our experiments we believe that these algorithms are effective in handling QoS type interval data.

We first compare the performance of the dynamic clustering algorithm for distinct clusters when using city block and Hausdorff distance measures respectively. As seen in Table 6 and 7 the CR index of the clustering results using the two distance measures are represented for the distinct and overlapping cluster datasets. The tables list the CR index values for the dynamic clustering algorithm with *K* set to 3.

Table 6 - CR index: Comparing city block vs. Hausdorff similarity for distinct clusters

Predefined intervals	Data set 2	
γ1, γ2, γ3	City block	Hausdorff
[1,4] [1,8] [1,8]	1.0000	1.0000
[1,8] [1,16] [1,16]	0.9900	0.9900
[1,12] [1,24] [1,24]	0.9900	0.9900
[1,16] [1,32] [1,32]	0.9800	1.0000

Table 7 - CR index: Comparing city block vs. Hausdorff similarity for overlapping clusters

Predefined intervals	Data set 5	
γ1, γ2, γ3	City block	Hausdorff
[1,4] [1,8] [1,8]	0.1392	0.1519
[1,8] [1,16] [1,16]	0.1156	0.1194
[1,12] [1,24] [1,24]	0.1185	0.0819
[1,16] [1,32] [1,32]	0.0930	0.0850

From the above two tables, it can be observed that the Hausdorff distance yields better results for distinct clusters, and overlapping clusters when predefined interval values are small. For overlapping clusters generated using very large intervals, the city block measure yields slightly better performance. We could get the similar conclusion on other datasets: Hausdorff distance measure constantly performs better than city-block for distinct clusters, whereas when the degree of overlapping between clusters is getting bigger, there is no obvious winner between the two measurements. We proceed to choose the Hausdorff distance measure to run the algorithm and present the results that follow.

In addition, when clusters are well separated, as in Table 6, we could achieve a very high CR index value. On the other hand, when there is a certain level of overlap due to the widened intervals, the CR index value decreases in number as in Table 7.

The next step is to compare the effectiveness of the dynamic clustering algorithm for various patterns of data distribution. The first condition is to compare the results for far-apart distinct clusters with closely placed distinct clusters.

Table 8 - CR index: Comparing far apart distinct vs. closely distinct clusters

Predefined intervals	Data set 1	Data set 2
γ1, γ2, γ3	Hausdorff	Hausdorff
[1,4] [1,8] [1,8]	1.0000	1.0000
[1,8] [1,16] [1,16]	1.0000	0.9900
[1,12] [1,24] [1,24]	1.0000	0.9900
[1,16] [1,32] [1,32]	1.0000	1.0000

In Table 8 above, we observe that for the far-apart distinct clusters, the values are maximum indicating the correctness level of the clustering results to be perfect. While with increasing variance and deceasing distance of separation between clusters, the CR index value is slightly less than the maximum value but still indicates very good performance levels.

For our next step, we further break down the closely distinct clusters dataset into three other types using different variance settings. We calculate the CR index values for the dynamic clustering algorithm on the respective datasets.

As seen in Table 9, the performance of the dynamic clustering algorithms remains effective with a CR index value equal to or closer to the desired value of 1 for alternate data distribution patterns following a multitude of variance settings.

Table 9 - CR index: Comparing performance for datasets with multiple variance settings

Predefined intervals	Data set 2	Data set 3	Data set 4
γ1, γ2, γ3	Hausdorff	Hausdorff	Hausdorff
[1,4] [1,8] [1,8]	1.0000	0.970	0.9899
[1,8] [1,16] [1,16]	0.9900	0.980	0.9701
[1,12] [1,24] [1,24]	0.9900	0.980	0.9800
[1,16] [1,32] [1,32]	1.0000	0.932	0.9799

Subsequently, we compare the performance for datasets with well separated clusters against datasets overlapping across all 3 attributes for all 3 clusters of the dataset.

Table 10 - CR index: Comparing separated Vs overlapping clusters

Predefined intervals	Data set 1	Data set 5
γ1, γ2, γ3	Hausdorff	Hausdorff
[1,4] [1,8] [1,8]	1.0000	0.1519
[1,8] [1,16] [1,16]	1.0000	0.1194
[1,12] [1,24] [1,24]	1.0000	0.0819
[1,16] [1,32] [1,32]	1.0000	0.0850

As seen in Table 10 the dynamic clustering algorithms yields the perfect performance for well separated clusters and yields slightly worse performance for overlapping clusters.

We now compare the effectiveness of the dynamic clustering algorithm for datasets with overlapping clusters following different trends in the way the clusters overlap. We consider datasets that overlap across all 3 clusters, 2 out of 3 clusters, across all 3 attributes for all 3 clusters and for 3 different combinations across any two attributes of all 3 clusters.

Table 11 - CR index: Comparing overlapping clusters

Predefined intervals	Data set 5	Data set 6	Data set 7
γ1, γ2, γ3	Hausdorff	Hausdorff	Hausdorff
[1,4] [1,8] [1,8]	0.1519	0.7593	0.8491
[1,8] [1,16] [1,16]	0.1194	0.7390	0.8409
[1,12] [1,24] [1,24]	0.0819	0.5287	0.8761
[1,16] [1,32] [1,32]	0.0850	0.4806	0.7716

Table 12 - CR index: Comparing overlapping clusters

Predefined intervals	Data set 8	Data set 9	Data set 10	Data set 11
γ1, γ2, γ3	Hausdorff	Hausdorff	Hausdorff	Hausdorff
[1,4] [1,8] [1,8]	0.7758	0.9217	0.9407	0.9701

It is quite evident from the Table 11 and 12 above that, when the degree of overlap is increasing, CR index value is decreasing. The low CR index value does not mean the clustering quality is poor; it is only an indication that the results do not match with the prior classification, which is very likely to happen for overlapping part. Within the same data set, when predefined intervals are becoming wider, the degree of overlapping is higher. Thus, we observed that the dynamic clustering algorithm is effective for both distinct and overlapping cluster combinations.

Now that we have evaluated the dynamic clustering algorithm for a good set of simulated datasets, representing different data distribution patterns that are possible, we try to test with modified datasets that are closer to the real data. For this purpose, we use selected datasets discussed above and add random points to these dataset. This way we not only consider datasets with vectors falling under well-defined boundaries, but regard widely spread data. We believe that this form of representation is closer to data in real applications.

Tables 13-16 below show that even with the addition of random points to the dataset, the values for the CR indexes are still above *0* indicating better performance for a range of [-1,1] of the CR index value. Although the CR index for datasets with random points might be lower, it is still in an acceptable level.

Table 13 - CR index: Comparing distinct far apart clusters with and w/o random points

Predefined intervals	Data set 1	Data set 12
γ1, γ2, γ3	Hausdorff	Hausdorff
[1,4] [1,8] [1,8]	1.0000	1.0000
[1,8] [1,16] [1,16]	1.0000	1.0000
[1,12] [1,24] [1,24]	1.0000	1.0000
[1,16] [1,32] [1,32]	1.0000	1.0000

Table 14 - CR index: Comparing closely distinct clusters with and w/o random points

Predefined intervals	Data set 2	Data set 13
γ1, γ2, γ3	Hausdorff	Hausdorff
[1,4] [1,8] [1,8]	1.0000	1.0000
[1,8] [1,16] [1,16]	0.9900	1.0000
[1,12] [1,24] [1,24]	0.9900	1.0000
[1,16] [1,32] [1,32]	1.0000	1.0000

Table 15 - CR index: Comparing 3 overlapping clusters dataset with and w/o random points

Predefined intervals	Data set 5	Data set 14
γ1, γ2, γ3	Hausdorff	Hausdorff
[1,4] [1,8] [1,8]	0.1519	0.1382
[1,8] [1,16] [1,16]	0.1194	0.1124
[1,12] [1,24] [1,24]	0.0819	0.0750
[1,16] [1,32] [1,32]	0.0850	0.0735

Table 16 - CR index: Comparing 2 overlapping clusters dataset with and w/o random points

Predefined intervals γ1, γ2, γ3	Data set 6 Hausdorff	Data set 15 Hausdorff
[1,4] [1,8] [1,8]	0.7593	0.3018
[1,8] [1,16] [1,16]	0.7390	0.3018
[1,12] [1,24] [1,24]	0.5287	0.3018
[1,16] [1,32] [1,32]	0.4806	0.3018

After we finish the experiment on the dynamic clustering algorithms, we move on to the hierarchical algorithms. The next step is to select the appropriate linkage method for running the hierarchical clustering algorithm in our future tests. We also compare the performance of the dynamic algorithm with the hierarchical clustering algorithms for datasets 1, 2, 5 and 6. The table below provides the CR index values for the hierarchical clustering algorithm using the three different linkage methods.

Table 17 - CR index for hierarchical clustering comparing 3 linkage methods

Data set	Single link	Average link	Complete link
1	-0.0029	0.0015	0.0018
2	-0.00002	0.0036	0.0144
5	0.000044	0.000691	0.000368
6	-0.0028	0.0137	0.1040

From the results in Table 17, we could see that the complete link method yields the highest CR index values for data set 1, 2 and 6, while for data set 5, the average link method performs the best. If we compare the results from the dynamic clustering algorithm with those from the hierarchical clustering algorithm on the same data sets, we find that dynamic clustering consistently achieves a much better result. We also compare the average run time of the two algorithms using all the distance and link measures as given by the table below. We see that the dynamic clustering algorithm is the most efficient in terms of its run time as given by the values in Table 18 below.

Table 18 - Average run time of dynamic and hierarchical algorithms

Methods	Average run time (sec) for dataset with at least 300 input vectors
Dynamic clustering without optimal k method	5
Dynamic clustering with optimal k for k=5 method	33
Hierarchical – Single link	526
Hierarchical – Average link	405
Hierarchical – Complete link	732

Although in the context of information retrieval, many document clustering systems use hierarchical algorithms due to their effectiveness, for the QoS-based clustering, we prefer the dynamic clustering algorithm over the hierarchical algorithm for three reasons. Firstly, in the case of document clustering systems, the data set usually comprises of a large number of documents to be clustered, hence it is harder to set a proper *K* value and find proper initial seed points for partitioning algorithms. As a result, the performance of partitioning algorithms may not be as good as the hierarchical algorithms. However, in our case, the data set of service QoS vectors is much smaller as they are already filtered from a larger set of published services based on their functionalities. Hence, partitioning algorithms perform much better. Secondly, it is desirable to have flat clusters that distinctly group QoS vectors in comparison to a hierarchy that allows a single service to be grouped under multiple clusters. The latter case usually poses an added level of cognitive overload for users to understand QoS distribution patterns. Whereas grouping documents into a hierarchy is desirable because it matches with the natural hierarchical relationships between different subjects or concepts. Thirdly, from our experiment, we note that dynamic clustering algorithm has minimum run time of 5000 milliseconds, while the best average run time for all hierarchical methods is not less than 250,000 milliseconds. Hence, efficiency wise the dynamic clustering algorithm is a better choice. Moreover, efficiency of the algorithm is highly desirable in the browsing step of our proposed framework, to improve the interactive capability of the system. Based on the above inferences we select the dynamic clustering algorithm using the Hausdorff distance measure to implement our interactive QoS browsing process.

4.4 Illustrating the Interactive QoS Browsing Process

To evaluate our proposed framework for selection, we conducted our experiments on two new datasets. The datasets used possess the following characteristics that differentiate it from the earlier datasets used for evaluating the clustering algorithms and the distance measures.

Data set 16:

- Total number of clusters equals three. Cluster 1 is further composed of three closely distributed regions of QoS vectors.
- The distribution patterns for the QoS attributes are shown below:

  Cluster 1: High reliability, low response time, and low price

  Cluster 2: Medium reliability, high response time, and medium price

  Cluster 3: Low reliability, low response time, and high price

Data set 17:

- Total number of clusters equals three. Each of the three clusters is composed of three closely distributed regions of QoS vectors.

- The distribution patterns for the QoS attributes are shown below:

  Cluster 1: Medium reliability, high response time, and low price

  Cluster 2: High reliability, high response time, and medium price

  Cluster 3: Low reliability, high response time, and high price

- In addition we add random points within and around the three clusters

We can note from the above listed points that the properties of the simulated datasets have been carefully chosen to possibly mimic the typical scenario of services distributed in registries and other sources, based on QoS. The first dataset shows a set of functionally similar services that are also similar to a certain degree in their QoS attribute values. In the second dataset, by adding random points, we show that not all functionally similar services are required to have quality attribute values falling in a closer range. We believe that this type of representation is ideal in real world scenarios. The experiment results are intended to show that the interactive browsing through services based on QoS is rational and effective in selecting services. The browsing approach does not require the user to face dilemmas in specifying his non-functional requirements as input queries for service search. On the other hand, the proposed technique aids the requestor in continuing with a more accurate searching process by using the results of his browsing process as deemed necessary.

Before we proceed with the example illustrations, we quickly review the suggested methodology of this work with the help of a simple flow diagram given by Figure 8.

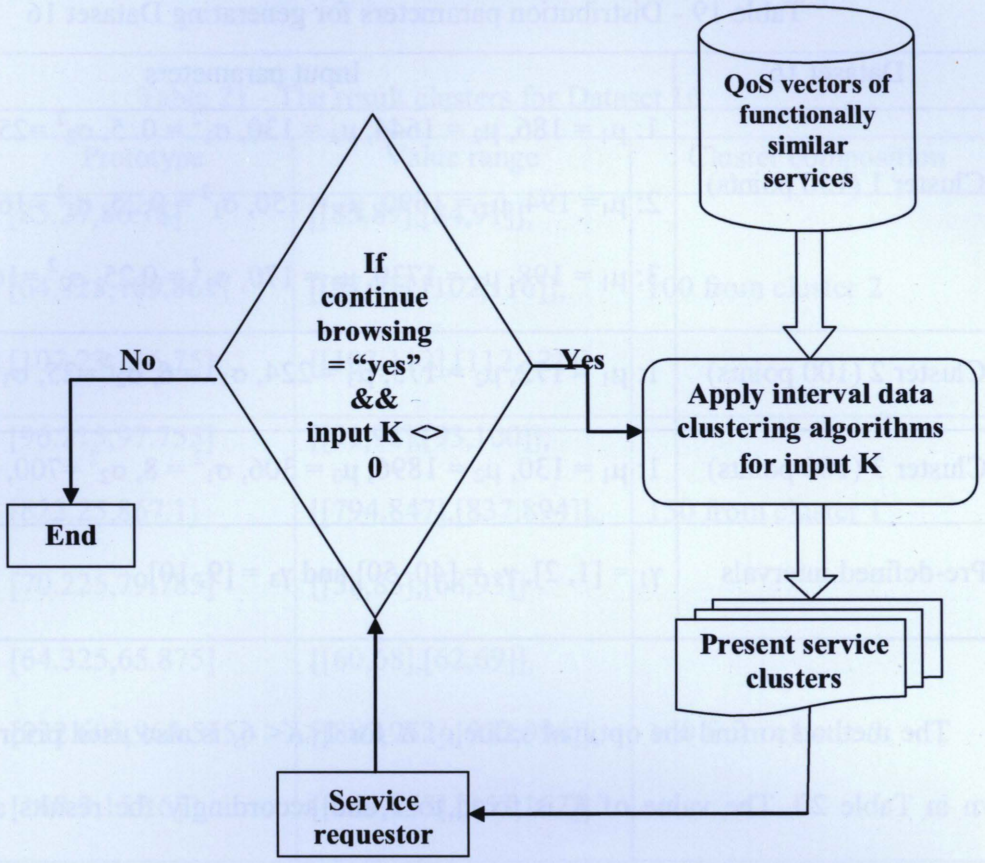

QoS vectors of functionally similar services
Apply interval data clustering algorithms for input K
Present service clusters
Service requestor
If continue browsing = "yes" && input K <> 0
Yes
No
End

Figure 8 - Flow diagram for interactive QoS browsing

Table 19 shows the distribution parameters for generating datasets 16. We first illustrate the first level of the browsing process using dataset 16.

Table 19 - Distribution parameters for generating Dataset 16

Dataset 16	Input parameters
Cluster 1 (150 points)	1: $\mu_1 = 186, \mu_2 = 1644, \mu_3 = 130, \sigma_1^2 = 0.5, \sigma_2^2 = 25, \sigma_3^2 = 4$ 2: $\mu_1 = 194, \mu_2 = 1690, \mu_3 = 150, \sigma_1^2 = 0.25, \sigma_2^2 = 16, \sigma_3^2 = 9$ 3: $\mu_1 = 198, \mu_2 = 1730, \mu_3 = 170, \sigma_1^2 = 0.25, \sigma_2^2 = 16, \sigma_3^2 = 9$
Cluster 2 (100 points)	1: $\mu_1 = 172, \mu_2 = 175, \mu_3 = 224, \sigma_1^2 = 6, \sigma_2^2 = 25, \sigma_3^2 = 12$
Cluster 3 (100 points)	1: $\mu_1 = 130, \mu_2 = 1896, \mu_3 = 306, \sigma_1^2 = 8, \sigma_2^2 = 700, \sigma_3^2 = 16$
Pre-defined intervals	$\gamma_1 = [1, 2], \gamma_2 = [40, 50]$ and $\gamma_3 = [9, 10]$

The method to find the optimal value of K for $1<K<6$, is also used prior to clustering as shown in Table 20. The value of K is fixed to 3 and accordingly the results are displayed to requestors in the form of three service QoS clusters.

Table 20 - C-H index, C-index and Γ-index for different K values for Dataset 16

K	C-H index	C-index	Γ-index
5	609695.91140	0.00001	0.94352
4	722964.45236	-0.00000	1.00000
3	***252551.07373***	***0.00002***	***0.98767***
2	139343.85255	0.00000	1.00000

For the dataset 16, the clustering algorithm is run once to present the users with the results. They are in the form of: cluster size, the prototype - $[gq_{1s,k}, gq_{1e,k}]$, ($[gq_{2s,k}, gq_{2e,k}]$, $[gq_{3s,k}, gq_{3e,k}]$) (1 for reliability, 2 for response time and 3 for price, and k is from 1 to 6), the value range for each QoS attribute ([min-reliability, max-reliability] [min-time, max-time] [min-price, max-price]), and the composition of the clusters corresponding to the input. The results are given in Table 21.

Table 21 - The result clusters for Dataset 16

	Size	Prototype	Value range	Cluster composition
1	100	[85.29,86.78] [64.325,109.865] [107.23,116.75]	[[83,89],[84,91]], [[57,71],[102,116]], [[103,112],[112,121]]	100 from cluster 2
2	150	[96.215,97.755] [822.25,867.1] [70.225,79.785]	[[91,98],[93,100]], [[794,847],[837,894]], [[58,83],[68,93]]	150 from cluster 1
3	100	[64.325,65.875] [922.605,968.515] [148.2,157.67]	[[60,68],[62,69]], [[887,952],[932,996]], [[143,153],[153,162]]	100 from cluster 3

In Table 21, the names cluster 1, 2 and 3 correspond to the input clusters in the a priori classification. From the result clusters, the requestors can observe that cluster 1 is more desirable with respect to time, while cluster 2 would be chosen with respect to both reliability and price and cluster 3 is the least attractive choice in comparison. In addition, we can say that the algorithm is effective in summarizing the QoS information to requestors in the form of coarsely grained service clusters.

We proceed to demonstrate an advanced run of the interactive QoS browsing using dataset 17. Figure 9 shows the 3D representation of dataset 17. Table 22 gives the various distribution parameters and the min-max value ranges for generating random points in the order of reliability, response time and price for dataset 17.

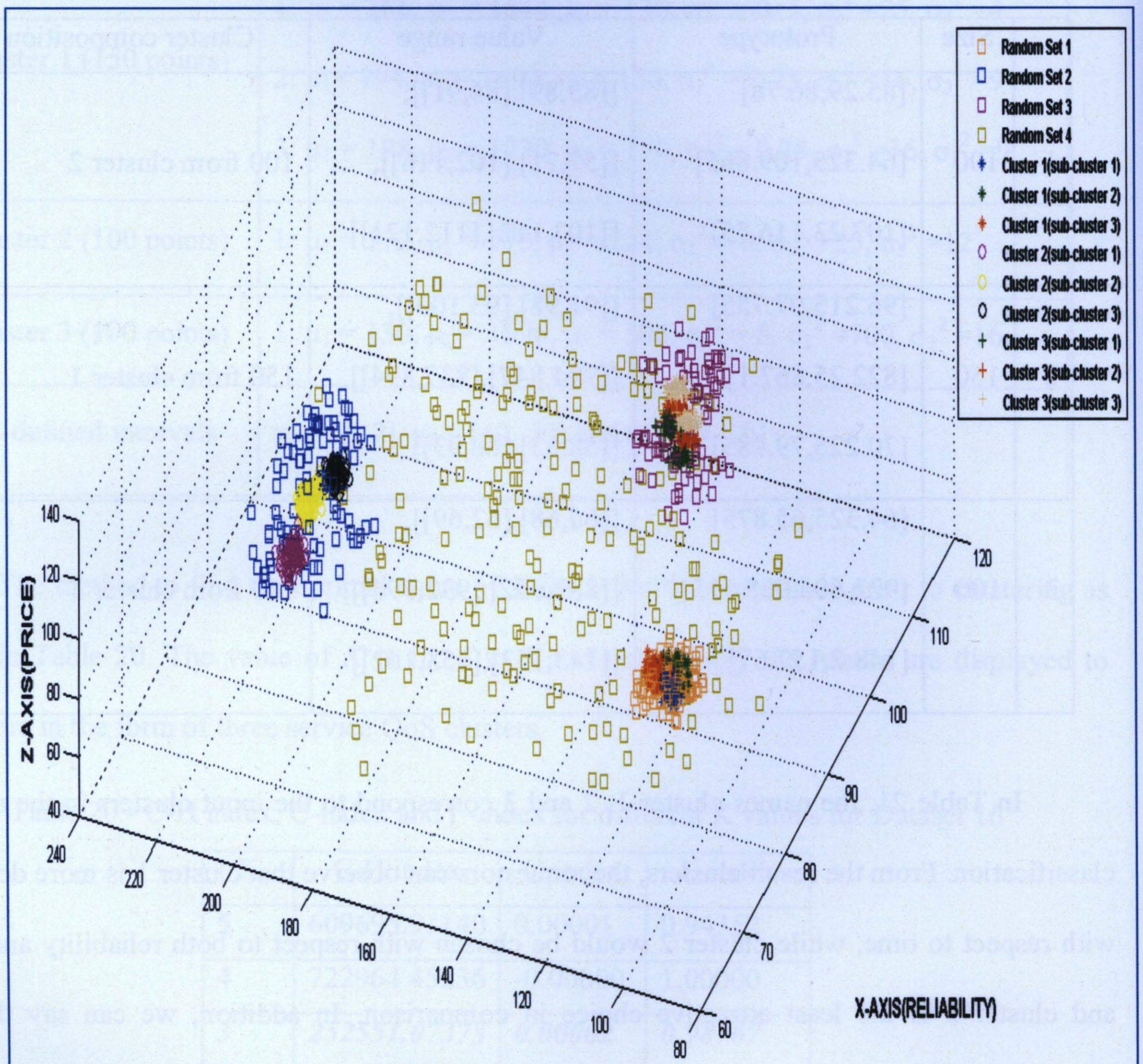


Figure 9 - 3D representation for Dataset 17

Table 22 - Input parameters for Dataset 17

	Input parameters
Cluster 1 (150 points)	1: $\mu_1 = 154$, $\mu_2 = 212$, $\mu_3 = 188$, $\sigma_1^2 = 0.45$, $\sigma_2^2 = 6.5$, $\sigma_3^2 = 3$ 2: $\mu_1 = 157$, $\mu_2 = 213$, $\mu_3 = 189$, $\sigma_1^2 = 0.45$, $\sigma_2^2 = 7.25$, $\sigma_3^2 = 5$ 3: $\mu_1 = 155$, $\mu_2 = 220$, $\mu_3 = 190$, $\sigma_1^2 = 0.65$, $\sigma_2^2 = 10$, $\sigma_3^2 = 4$ $\gamma_1 = [0.5, 1]$, $\gamma_2 = [1, 2]$ and $\gamma_3 = [1, 5]$
Cluster 2 (140 points)	1: $\mu_1 = 165$, $\mu_2 = 420$, $\mu_3 = 160$, $\sigma_1^2 = 2.5$, $\sigma_2^2 = 6$, $\sigma_3^2 = 1.75$ 2: $\mu_1 = 178$, $\mu_2 = 435$, $\mu_3 = 162$, $\sigma_1^2 = 2.98$, $\sigma_2^2 = 3$, $\sigma_3^2 = 1.5$ 3: $\mu_1 = 186$, $\mu_2 = 420$, $\mu_3 = 161$, $\sigma_1^2 = 1.95$, $\sigma_2^2 = 6$, $\sigma_3^2 = 1.5$ $\gamma_1 = [0, 1]$, $\gamma_2 = [1, 2]$ and $\gamma_3 = [2, 3]$
Cluster 3 (150 points)	1: $\mu_1 = 191$, $\mu_2 = 250$, $\mu_3 = 240$, $\sigma_1^2 = 0.65$, $\sigma_2^2 = 6$, $\sigma_3^2 = 3$ 2: $\mu_1 = 195$, $\mu_2 = 251$, $\mu_3 = 241$, $\sigma_1^2 = 0.95$, $\sigma_2^2 = 8$, $\sigma_3^2 = 3$ 3: $\mu_1 = 192$, $\mu_2 = 248$, $\mu_3 = 261$, $\sigma_1^2 = 0.5$, $\sigma_2^2 = 6$, $\sigma_3^2 = 5$ $\gamma_1 = [0, 1]$, $\gamma_2 = [3, 7]$ and $\gamma_3 = [5, 10]$
Random set 1 (50)	[(74-80), (75-81)], [(100-113), (102-114)], [(89-95), (93-99)]
Random set 2 (50)	[(80-91), (89-100)], [(197-224), (205-225)], [(69-83), (78-91)]
Random set 3 (40)	[(88-96), (97-105)], [(111-132), (118-136)], [(114-127), (124-138)]
Random set 4 (100)	[(69-104), (82-116)], [(94-185), (120-210)], [(47-114), (59-127)]

We start by entering the whole data set as input to the dynamic clustering algorithm. In order to find the optimal *K*, using the method that finds the three statistical indices when $0<K< 9$, and the result is shown in Table 23.

In the event of conflict for the best value of *K* in terms of the values of 2 or more indices, we will follow the steps discussed in section 4.4.2 to fix *K*. Following this principle, we choose optimal *K* as 6 because it is the best for C-index and Γ-index, although it is not the best for C-H index.

Table 23 - C-H index, C-index and Γ-index for different *K* values in the first round

K	C-H index	C-index	Γ-index
9	804.96862	0.01479	0.95104
8	910.78817	0.01091	0.94830
7	743.93457	0.02902	0.93490
6	***1182.20291***	***0.00525***	***0.96293***
5	1144.69582	0.01144	0.95409
4	1274.30594	0.02241	0.94014
3	1068.44013	0.08229	0.81188
2	1407.47450	0.04466	0.93666

After we set *K* as 6, we do the first iteration of clustering. Table 24 shows the clustering result in the first level. For each cluster, we present the results in the same fashion as for dataset 16. From this table, we could see that all 3 clusters in the original data set have been correctly identified, and the random data is clustered into different groups based on their values.

Table 24 - The first level clusters with K=6

	Size	Prototype	Value range	Cluster composition
1	69	[95.75, 96.44], [122.21, 125.8], [125.98, 134.35]	[[95, 97], [96, 97]], [[119, 125], [122, 128]], [[124, 129], [132, 137]]	50 from sub-cluster 3 of cluster 3, and 19 from random set 3
2	191	[88.66, 89.34], [210.55, 212.16], [79.27, 81.88]	[[80, 94], [80, 95]], [[207, 213], [208, 215]], [[77, 81], [80, 84]]	140 from cluster 2, 50 from random set 2, and 1 from random set 4
3	51	[84.52, 100.65], [156.39, 174.20], [84.19, 85.18]	[[69, 104], [82, 116]], [[94, 185], [120, 210]], [[47, 114], [59, 127]]	51 from random set 4
4	130	[96.02, 96.64], [123.60, 126.84], [116.04, 124.40]	[[88, 96], [97, 104]], [[111, 131], [118, 135]], [[114, 126], [125, 137]]	100 from sub-cluster 1 and 2 of cluster 3, 21 from random set 3, 9 from random set 4
5	33	[82.36, 99.48], [113.65, 175.93], [84.00, 97.28]	[[69, 104], [82, 116]], [[94, 185], [120, 210]], [[47, 114], [59, 127]]	33 from random set 4
6	206	[77.29, 78.06], [106.35, 107.98], [92.82, 95.89]	[[75, 79], [76, 80]], [[102, 113], [103, 114]], [[89, 96], [92, 99]]	150 from cluster 1, 50 from random set 1, 6 from random set 4

Suppose by checking these clusters, a requestor selects cluster 2 and 4 based on price and reliability. Then we will do the re-clustering on these selected QoS vectors. Again, we need to find optimal *K* for this level. Table 25 shows the results for the 3 indices.

Table 25 - C-H index, C-index and Γ-index for different *K* values in the second round

K	C-H index	C-index	Γ-index
9	1363.27418	0.01314	0.86488
8	1475.31765	0.01150	0.87493
7	1819.30277	0.00856	0.82631
6	1945.48732	0.01556	0.81564
5	2090.66573	0.01459	0.78485
4	2615.39297	0.01775	0.78276
3	***3763.48829***	***0.01364***	***0.85353***
2	6961.86844	0.00002	1.00000

From Table 25, we could see that the optimal choice is *K*=2. But since we have 2 clusters already and we want to zoom in to see more details about these two clusters, we would choose the second optimal choice instead, which is *K*=3.

Now we set *K* as 3 and do the second iteration of clustering. Table 26 shows the clustering result in the second level. We could see that sub-clusters have been successfully identified. If we choose cluster 2 and continue the process, we could get results as shown in Table 27.

Table 26 - The second level clusters with K=3

	Size	Prototype	Value range	Cluster composition
1	44	[82.65, 83.13], [209.00, 210.58], [78.61, 81.13]	[[80, 85], [80, 86]], [[207, 212], [209, 214]], [[77, 81], [80, 83]]	40 from sub-cluster 1 of cluster 2, 3 from random set 2, and 1 from random set 4
2	130	[96.02, 96.64], [123.60, 126.84], [116.04, 124.40]	[[88, 96], [97, 104]], [[111, 131], [118, 135]], [[114, 126], [125, 137]]	100 from sub-cluster 1 and 2 of cluster 3, 21 from random set 3, and 9 from random set 4
3	147	[89.50, 90.22], [211.26, 212.87], [79.51, 82.16]	[[80, 94], [80, 95]], [[207, 213], [208, 215]], [[77, 81], [80, 84]]	100 from sub-cluster 2 and 3 of cluster 2, and 47 from random set 2

Table 27 - The third level clusters with K=4

	Size	Prototype	Value range	Cluster composition
1	52	[82.65, 83.13], [209.00, 210.58], [78.61, 81.13]	[[94, 96], [95, 97]], [[121, 127], [124, 130]], [[113, 117], [121, 126]]	50 from sub-cluster 1 of cluster 3, 1 from sub-cluster 2 of cluster 3, and 1 from random set 4
2	20	[96.02, 96.64], [123.60, 126.84], [116.04, 124.40]	[[88, 96], [97, 105]], [[111, 132], [118, 136]], [[114, 127], [124, 138]]	20 from random set 3
3	6	[89.50, 90.22], [211.26, 212.87], [79.51, 82.16]	[[69, 104], [82, 116]], [[94, 185], [120, 210]], [[47, 114], [59, 127]]	6 from random set 4
4	52	[89.50, 90.22], [211.26, 212.87], [79.51, 82.16]	[[96, 98], [97, 99]], [[121, 128], [123, 130]], [[114, 119], [122, 128]]	49 from sub-cluster 2 of cluster 3, 1 from random set 3, and 2 from random set 4

By running through the steps of our QoS based selection framework, we observe that the browsing process helps the requestors gain a finer view of the QoS distribution for services with every iteration of the clustering algorithm. It allows them to zoom in to their desired services with an improved degree of confidence about the quality of service they are being served. In the above illustration, each time we try to find the optimal *K* first, and then do the clustering. Alternatively, requestors could specify a fixed *K* value, skip the step of finding optimal *K* and make the process faster. It is the requestor's decision about how to balance between the accuracy and the efficiency.

Clustering algorithms might not work well for a tightly formed cluster in which all data points are very close to each other, and in this case, the cluster will be randomly partitioned into a few highly overlapping groups. When this kind of clustering result is presented to the requestor, it does not help much to understand the dataset better. In this kind of situation, our proposed browsing method could not help the service selection process, and we might need to consider some other mechanisms.

4.5 Chapter Summary

In this chapter, we present the experiments conducted to test our proposed framework for service selection with the interactive browsing component. Since there are no benchmark datasets available for web services QoS data, we used simulated datasets in our experiments. Our simulated datasets are generated following the value ranges we observed from the real QoS data. Detailed explanations of the steps to dataset simulation were provided along with various scenarios used. The procedures to compare and evaluate the clustering algorithms are outlined along with reasonable conclusions made in selecting the clustering algorithm. The main steps to the QoS browsing process are illustrated with the help of sound examples. Additionally, an optimal *K* finding method was used in connection to fix one of the input parameters for the dynamic clustering algorithm.

CHAPTER 5

CONCLUSION

5.1 Summary and Results

The thesis presents a unique and interactive QoS browsing framework for web service selection, motivated by the findings from an extensive review of frameworks and models for QoS based service selection.

The importance of the functional and non-functional (e.g. QoS) requirements for users assessing a service are pointed out. A QoS-based clustering mechanism to group functionally similar services together, showing how the quality attribute values are distributed within the service collection, is used. The resulting clusters could help requestors select their desired services or help providers understand what kind of services are currently available in the registry and are being used by requestors. We believe that browsing is necessary for QoS-based service selection in comparison with the current techniques employing a searching based approach. The search techniques often leave the requestors with a cognitive overload making the formulation of the right queries difficult. The search process is further influenced by the requestor's lack of knowledge on QoS distributions in the registry, vagueness of the QoS requirements, and dynamism of the QoS values offered by providers. With browsing integrated with an iterative clustering algorithm, the requestors can proceed with a guided service selection process that produces service clusters, portraying the distribution of services based on their QoS attribute values. We also analyzed the data types used for representing web services QoS data as seen in current service selection methods, which suggest a generalized form of representation for the QoS attribute values. We propose to use interval data types for QoS representation that meaningfully represent the attributes in a comprehensive manner. With this type of data representation, it is seen that the information lost in representation can be avoided and the level of accuracy for publisher-provided QoS could be improved. Each of the three QoS attributes considered in this work are expressed symbolically as a combined interval data vector, which is more accurate when compared with single valued numerical data representation.

The importance of using special types of clustering algorithms to group the multidimensional QoS attributes with interval data types is emphasized. An elaborate set of experiments have been performed to evaluate the interval data clustering algorithms along with an assessment of the suitable distance and similarity measure used by these algorithms. Additionally, a method to find the optimal value of an input parameter to one of the clustering algorithms has been implemented. Our approach is also easily extensible for multiple QoS attributes with varying data types, by extending the clustering algorithm to cover the complete set of symbolic data. Based on the evaluation of the experimental results, the dynamic clustering algorithm is chosen over the hierarchical interval-data clustering algorithm for implementing the proposed interactive QoS browsing framework. Starting from the initial set of services with the similar functionality, we apply the dynamic interval-data clustering algorithm on their QoS vectors to get the initial clusters. Then with requestors' selection of a subset of clusters and re-clustering on this subset, the requestor could add a number of detailed views on their preferred QoS vectors. Thus the browsing process enables the requestors to get a finer and finer view of the QoS distribution patterns for services with every iteration of the algorithm.

Our QoS browsing approach provides the requestors with a flexible option for searching web services by exercising their non-functional service requirements in the search process. By involving the requestors in the search process, they are able to make selection decisions with an increased level of confidence.

5.2 Future Work

By utilizing the findings and results of this thesis, it is feasible to use a simple and interactive QoS browsing method to select web services based on their quality attributes data. However, keeping in mind the difficulties and important observations made during the work a few directions for potential future work along these lines would be pointed out.

First, a visual interface showing the distribution of data points in result clusters and related information such as prototypes, sizes, value ranges, and deviation levels of the clusters could be implemented. This would enable requestors to make a more informed decision so as to choose the best service, satisfying their QoS requirements and simplify the effort needed to understand the clustering results in the case of requestors searching for public services.

Second, in the current work, we specify the potential QoS information sources to find QoS attribute values. As part of this the handling of QoS data from multiple sources could be more clearly elaborated. When the QoS data is collected from publishers, SLAs or monitoring agents, it might affect how we do the clustering. For instance, if the QoS data is from SLA documents, for one service, there could be multiple sets of QoS values for different requestors or even for the same requestors over different periods in time. Should these be counted equally as separate instances or grouped under one service? This is a complicated issue requiring a dedicated study to find satisfying results.

Third, as indicated in this work, there is an absence of standard data sets for QoS data in evaluating the performance of QoS based WS selection frameworks in the real world. This serves as an important limitation for the experiments given in this work. Steps could be taken along these lines to develop standard QoS datasets for services, which would require an entirely new piece of work devoted to obtain results.

Fourth, the experiments discussed in this work consider limited number of QoS attributes for web services during WS clustering and selection. Alternatively, the simulation experiments could be repeated for exponential number of times covering the entire data set of QoS attributes. By applying, more generic symbolic clustering algorithms, the performance of the approach can further be evaluated using different scenarios for initialization of the algorithms. In addition to the single normal and uniform distribution patterns considered during the data simulation, other types of distribution patters can also be employed and experimented with.

Fifth, it would be interesting to implement a multilevel clustering of services based on the preferential QoS requirements of requestors. The method could consider the preferential value ranges for one QoS attribute at a time in each level of the browsing process. This approach would thus implement a context aware service selection approach for web services.

Sixth and finally, the study of whether providers and requestors have different expectations or perspectives on the clustering results could be made, so that the clustering process could be customized for them. This could be done conducting user surveys with open-ended questions evaluating the effectiveness of the system with respect to the user.

REFERENCES

1. W. Abramowicz, K. Haniewicz, M. Kaczmarek and D. Zyskowski, "Architecture for web services filtering and clustering," *In Proceedings Of Second International Conference on Internet and Web Applications and Services, ICIW'07,* Art. No.: 4222920, Mauritius, pp. 18, 2007.
2. Amazon. http://aws.amazon.com/.
3. S. Asharaf, M. N. Murty and S. K. Shevade, "Rough set based incremental clustering of interval data," *Pattern Recognition Letters,* Vol. 27, pp. 515-519, 2006.
4. D. Bachlechner, K. Siorpaes, H. Lausen, and D. Fensel, "Web service discovery a reality check", *Technical report 1, Digital Enterprise Research Institute (DERI), Galway, Innsbruck, Seoul, Stanford,* 2006.
5. A. Baraldi and P. Blonda, "A survey of fuzzy clustering algorithms for pattern recognition Part I and II", *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, Vol. 29, No. 6, pp. 778–801, 1999.
6. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, "Web Services Architecture", W3C Working Group Note 11 Feb. 2004. Available: http://www.w3.org/TR/ws-arch/, last retrieved on June 3, 2009.
7. G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms", H.-G. Beyer and U.-M. O'Reilly, editors, *In Proceedings of Genetic and Evolutionary Computation Conference, GECCO,* Washington, DC, USA, pp. 1069–1075, 2005.
8. F. Carvalho, P. Brito and H.H. Bock, "Dynamic clustering for interval data based on L2 distance", *Computational Statistics*, Vol. 21, Issue 2, pp. 231-250, 2006.

9. F. D. A. T. De Carvalho, R. Souza, M. Chavent, and Y.Lechevallier, "Adaptive Hausdorff Distances and Dynamic Clustering of Symbolic Interval Data," *Pattern Recognition Letters*, 27(3), pp.167-179, 2006.

10. F. D. A. T. De Carvalho, R. M. C. R. De Souza and L. X. T. Bezerra, "A dynamical clustering method for symbolic interval data based on a single adaptive Euclidean distance," In *Proceedings Of Ninth Brazilian Symposium on Neural Networks, SBRN'06,* Art. No.: 4026808, Ribeirao Preto, Brazil, pp. 8-13, 2006.

11. M. Chavent, F.de A.T. de Carvalho, Y. Lechevallier, and R. Verde, "New Clustering Methods for Interval Data", *Computational Statistics*, Vol. 21, Issue 2, pp. 211-229, 2006.

12. S.T. Chelcea. "Agglomerative 2-3 Hierarchical Classication: Theoretical and Applicative Study". PhD thesis, Université de Nice-Sophia Antipolis, 2004.

13. W. Choo, B. Detlor and D. Turnbull, "Information Seeking on the Web – an Integrated Model of Browsing and Searching", *In Proceedings of the 62nd Annual Meeting of the American Society for Information Science*, Washington DC, pp 3-16, 1999.

14. D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Turkey. "Scatter/gather: A cluster-based approach to browsing large document collections", *In Proceedings of the 15th Annual Int'l ACM SIGIR Conference on R&D in IR*, New York , pp. 330-337, June 1992.

15. D.R. Cutting, D. Karger, and J. Pedersen, "Constant interaction-time Scatter/Gather browsing of very large document collections", *In Proceedings of the 16th Annual International ACM/SIGIR Conference*, Pittsburgh, PA, pp. 126-135, 1993.

16. J. Day and R. Deters. “Selecting the best web service”. *In Proceedings of the IBM Centers for Advanced Study Conference (CASCON ’04)*, Ontario, Canada, pp. 293–308, 2004.

17. B. Devis, V. de Antonellis, and M. Melchiori, “QoS in Ontology-based Service Classification and Discovery”, In *Proceedings of the 15th International Workshop on Database and Expert Systems Applications*, Saragossa, Spain, pp. 145-150, 2004.

18. E. Diday and M. Noirhomme-Fraiture, “*Symbolic Data Analysis and the SODAS Software”*. New York, NY, USA: Wiley-Interscience, 2008.

19. X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, “Similarity Search for Web Services”, *In VLDB*, Toronto, Canada, pp. 372-383, 2004.

20. A. Fred and A.Lourenco, “Cluster Ensemble Methods: from Single Clusterings to Combined Solutions”, *Studies in Computational Intelligence*, Vol. 126, pp. 3-30, 2008.

21. G. Gan, C. Ma, and J. Wu, “Data Clustering: Theory, Algorithms, and Applications”, *(ASA-SIAM Series on Statistics and Applied Probability)*, illustrated edition ed. SIAM, Society for Industrial and Applied Mathematics, May 2007.

22. Gartner. http://www.gartner.com/technology/research.jsp.

23. K.C. Gowda, and T.R. Ravi, “Agglomerative Clustering of Symbolic Objects Using the Concepts of Both Similarity and Dissimilarity”, *Pattern Recognition Letters*, Vol. 16, Issue 6, pp. 647-652, 1995.

24. J.Han and M.Kamber, “Data Mining: Concepts and Techniques”, London: Morgan Kaufmann. pp. 2-27, 335-391, 2005.

25. A.Hardy and J. Baune, “Clustering and Validation of Interval Data”, *Selected Contributions in Data Analysis and Classification*, Part I, pp. 69-82, 2007.

26. M. A. Hearst, and J. O. Pedersen (), “Re-examining the cluster hypothesis: scatter/gather on retrieval results”, *In Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval,* New York, NY, USA, pp. 76-84, 1996.

27. H. Hubert and P. Arabie, ”Comparing partitions”, *Journal of Classification*, Issue 2, pp. 193–218, 1985.

28. A.Irpino and V. Tontodonato, “Clustering reduced interval data using Hausdorff distance” , *Computational Statistics*, Vol. 21, Issue 2, pp. 271-288, 2006.

29. A.K. Jain, M.N. Murty, P.J.Flynn, “Data clustering: A review”, *ACM Computing Surveys*, Vol. 31, Issue 3, pp. 264-323. Retrieved November 4, 2007, from ABI/INFORM Global database.

30. S. Kalepu, S. Krishnaswamy, and S.W. Loke, “Reputation = f(user ranking, compliance, verity)”, *In Proceedings of IEEE International Conference on web services*, Washington, DC, USA, pp. 200- 207, 2004.

31. W.Ke, C.R. Sugimotoand J. Mostafa, “Dynamicity vs. Effectiveness: A User Study of a Clustering Algorithm for Scatter/Gather”, To appear in *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Boston, Massachusetts, 2009.

32. S. Lamparter, A.Ankolekar, R.Studer, and S.Grimm, “Preference-based Selection of Highly Configurable Web Services”, *In Proceedings of the 16th International Conference on World Wide Web*, Banff, Alberta, Canada, pp. 1013-1022, 2007.

33. Y. Lechevallier, R. Verde,” Crossed Clustering method: An efficient Clustering Method for Web Usage Mining”, *Complex Data Analysis*, China, 2004.

34. Y. Lechevallier, R. Verde, and F.de A.T. de Carvalho, “Symbolic Clustering of Large Datasets”, *Data Science and Classification,* Vol. 4, pp. 193-201, 2006.

35. K.-C. Lee et al., “QoS for Web Services: Requirements and Possible Approaches,” World Wide Web Consortium (W3C) note, Nov. 2003; Available online at (last accessed on Nov. 30, 2006): www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

36. S.M. Li, C. Ding, C.H. Chi, and J. Deng, “Adaptive Quality Recommendation Mechanism for Software Service Provisioning”, In *Proceedings of the IEEE International Conference on Web Services*, Beijing, China, pp. 169-176, 2008.

37. Y.T. Liu, A.H. Ngu, and L.Z. Zeng, “QoS Computation and Policing in Dynamic Web Service”, In *Proceedings of the 13th International Conference on World Wide Web*, New York, pp. 66-73, 2004.

38. W. Liu and W. Wong, “Discovery homogenous service communities through web service clustering”, *Service-Oriented Computing: Agents, Semantics, and Engineering*, Vol. 5006, pp. 69-82, 2008.

39. X.Z. Liu, L. Zhou, G. Huang, and H. Mei, “Consumer-Centric Web Services Discovery and Subscription”, *In Proceedings of IEEE International Conference on e-Business Engineering*, Hong Kong, China, pp. 543-550, 2007.

40. H. Ludwig, A. Keller, A. Dan, R.P. King, and R. Franck, “Web Service Level Agreement Language Specification”, 2003, Available: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf, last retrieved on June 20, 2009.

41. Z. Luo, K. Qian, D. Cai, and J. S. Li, “QoS driven web services assessment and selection,” *International Journal. Services Operations and Informatics*, Vol. 1, pp. 1–2, 2006.

42. J. Ma, Y. Zhang and J. He, “Efficiently Finding Web Services Using a Clustering Semantic Approach”, *In Proceedings of international workshop on Context enabled source and service selection, integration and adaptation*, Beijing, China, Vol. 292, pp. 5, 2008.

43. O. Maimon and L. Rokach, “The Data Mining and Knowledge Discovery Handbook”, *Springer*, 2005.

44. K. Mali, S. Mitra, “Clustering and its validation in a symbolic framework”. *Pattern Recognition Letters*, Vol. 24, pp. 2367-2376, 2003.

45. U. Manber, M. Smith, and B. Gopal, “WebGlimpse - Combining Browsing and Searching”, *In Proceedings of the 1997 USENIX Technical Conference*, Los Angeles, CA, pp. 195–206, 1997.

46. A.Mani and A. Nagarajan, “Understanding quality of service for Web services”, IBM Software labs Jan 2002. Available: http://www.ibm.com/developerworks/library/ws-quality.html, last retrieved in June, 2009.

47. C. Marie, and Y. Lechevallier, “Dynamical Clustering Algorithm of Interval Data: Optimization of an Adequacy Criterion Based on Hausdorff Distance,” Sokolowsky, Bock, H.H. (Eds.), *In Classification, Clustering and Data Analysis*, Springer-Verlag, Heidelberg, pp. 53-59, 2002.

48. E. Al-Masri and Q. H. Mahmoud, “Discovering the best web service”, *In Proceedings of 16th International Conference on World Wide Web*, Banff, Alberta, Canada, pp. 1257-1258, 2007.

49. E. Al-Masri, and Q.H. Mahmoud, “QoS-based Discovery and Ranking of Web Services”, *In Proceedings of the 6th International Conference on Computer Communications and Networks*, Honolulu, Hawaii , pp. 529-534, 2007.

50. R. Nayak and B. Lee, “Web Service Discovery with additional Semantics and clustering”, *IEEE/WIC/ACM International Conference on Web Intelligence*, Silicon Valley, USA, pp. 555-558, 2007.

51. Nottelmann and G.Fischer, “Search and browse services for heterogeneous collections with the peer-to-peer network Pepper”, *International Journal Information Processing and Management*, Vol. 43, pp. 624-642, 2007.

52. M. Papazoglou, “Web service technologies and standards”, *ACM Computing Surveys*, pp. 1-41, 2006.

53. W. Peng and T. Li, "Interval Data Clustering with Applications," In *Proceedings of 18th IEEE International Conference on Tools with Artificial Intelligence,* Arlington, VA, pp. 355-362, 2006.

54. P. Pirolli, W.Fu, E.Chi and A. Farahat, “Information scent and web navigation: Theory, models and automated usability evaluation”, *In Human-Computer Interaction International,* pp.5-12, 2006.

55. S. Ran, “A Model for Web Services Discovery with QoS”, *ACM SIGecom Exchanges*, Vol. 4, Issue 1, pp. 1-10, 2003.

56. Salesforce. http://www.salesforce.com.

57. SODAS software, http://www.info.fundp.ac.be/asso/.

58. Y. El-Sonbaty, M.A.Ismail ,"Fuzzy clustering for symbolic data", *IEEE Transactions on Fuzzy Systems,* Vol 6, pp. 195–204, 1998.

59. R.M.C.R. de Souza, and F.de A.T. de Carvalho, "Clustering of Interval Data Based on City-Block Distances", *Pattern Recognition Letters*, Vol. 25, Issue 3, pp. 353-365, 2004.

60. L.H. Vu, M. Hauswirth, and K. Aberer, "QoS-based Service Selection and Ranking with Trust and Reputation Management", *In Proceedings of the International Conference on Cooperative Information Systems*, Switzerland, pp. 446-483, 2005.

61. L. -. Vu, M. Hauswirth, F. Porto and K. Aberer, "A search engine for QoS-enabled discovery of semantic web services," *International Journal of Business Process Integration and Management,* Vol. 1, pp. 244-255, 2006.

62. Y. Wang and E. Stroulia, "Semantic structure matching for assessing web-service similarity," *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* Vol. 2910, pp. 194-207, 2003.

63. X.Wang, T. Vitvar, M. Kerrigan and I. Toma, "A QoS-Aware Selection Model for Semantic Web Services", *In Proceedings of the 4th Intl. Conf. on Service Oriented Computing*, Chicago, USA, pp. 390-401, 2006.

64. H. Wang, P. Tong, P. Thompson and Y. Li, "QoS-Based Web Services Selection", *In Proceedings of the IEEE International Conference on e-Business Engineering*, Hong Kong, China, pp. 631-637, 2007.

65. Y. Wang and J. Vassileva, "Toward Trust and Reputation Based Web Service Selection: A Survey," *International Transactions on Systems Science and Applications (ITSSA) Journal,* Vol. 3, pp. 118-132, 2007.

66. R.Xu and D.WunschII, "Survey of clustering algorithms", *IEEE Transactions on Neural Networks*, Vol. 16, Issue 3, pp. 645-678, 2005.

67. Z. Xu, P.Martin, W.Powley, and F. Zulkernine, "Reputation-Enhanced QoS-based Web Services Discovery", *In Proceedings of the IEEE International Conference on Web Services*, Salt Lake City, Utah, pp. 249-256, 2007.

68. S.J.H. Yang, J. Zhang and B. C.W. Lan, "Service-level agreement-based QoS analysis for web services discovery and composition", *International Journal of Internet and Enterprise Management,* Vol. 5, No.1, pp. 39 - 58, 2007.

69. K.Yeung and W.L. Ruzzo, "Details of the Adjusted Rand index and Clustering algorithms Supplement to the paper "An empirical study on Principal Component Analysis for clustering gene expression data", Available: http://faculty.washington.edu/kayee/pca/supp.pdf, last retrieved on June, 2009.

70. W. Zhongxin, Y. Dongbo, W. Yongjian, Y. Bingheng and Q. Depei, "Context-aware web service selection based on multi-aspects regulating," *In Proceedings Of The 2nd IEEE Asia-Pacific Services Computing Conference, APSCC 2007*, Tsukuba Science City, Japan, pp. 254-259, 2007.

APPENDIX A: INPUT PARAMETERS FOR DATA GENERATION

A. Separated clusters:

Table 28 - Data set 2: Distinct clusters (closely placed) with medium variance

Input parameters	

Total # of points = 300
Predefined interval
sets *a-d*
	Group 1 (# of points = 100):
	$\mu_1 = 155, \mu_2 = 618, \mu_3 = 390, \sigma_1^2 = 25, \sigma_2^2 = 4, \sigma_3^2 = 16$
	Group 2 (# of points = 100):
	$\mu_1 = 168, \mu_2 = 623, \mu_3 = 370, \sigma_1^2 = 2, \sigma_2^2 = 1, \sigma_3^2 = 4$
	Group 3 (# of points = 100):
	$\mu_1 = 175, \mu_2 = 620, \mu_3 = 410, \sigma_1^2 = 1, \sigma_2^2 = 16, \sigma_3^2 = 25$

Table 29 - Data set 3: Distinct clusters (closely placed) with large variance

Input parameters	

Total # of points = 300
Predefined interval
sets *a-d*
	Group 1 (# of points = 100):
	$\mu_1 = 155, \mu_2 = 618, \mu_3 = 390, \sigma_1^2 = 36, \sigma_2^2 = 36, \sigma_3^2 = 36$
	Group 2 (# of points = 100):
	$\mu_1 = 168, \mu_2 = 623, \mu_3 = 370, \sigma_1^2 = 4, \sigma_2^2 = 9, \sigma_3^2 = 16$
	Group 3 (# of points = 100):
	$\mu_1 = 175, \mu_2 = 620, \mu_3 = 410, \sigma_1^2 = 2, \sigma_2^2 = 49, \sigma_3^2 = 36$

Table 30 - Data set 4: Distinct clusters (closely placed) with multiple variance combinations

Input parameters	

Total # of points = 300
 	
Predefined interval
 	
sets *a-d*
	Group 1 (# of points = 100):
	
	$\mu_1 = 155, \mu_2 = 700, \mu_3 = 180, \sigma_1^2 = 64, \sigma_2^2 =225, \sigma_3^2 =144$
	
	Group 2 (# of points = 100):
	
	$\mu_1 = 170, \mu_2 = 770, \mu_3 = 210, \sigma_1^2 = 25, \sigma_2^2 =169, \sigma_3^2 =196$
	
	Group 3 (# of points = 100):
	
	$\mu_1 = 180, \mu_2 = 840, \mu_3 = 240, \sigma_1^2 = 9, \sigma_2^2 =256, \sigma_3^2 =169$

B. *Overlapping clusters:*

Table 31 - Data set 6: Overlapping across all 3 attributes for 2 out of 3 clusters

Input parameters	

Total # of points = 350
Predefined interval
sets *a-d*
	Group 1 (# of points = 150):
	$\mu_1 = 150, \mu_2 = 210, \mu_3 = 280, \sigma_1^2 = 25, \sigma_2^2 = 16, \sigma_3^2 = 9$
	Group 2 (# of points = 100):
	$\mu_1 = 140, \mu_2 = 212, \mu_3 = 275, \sigma_1^2 = 25, \sigma_2^2 = 16, \sigma_3^2 = 9$
	Group 3 (# of points = 100):
	$\mu_1 = 133, \mu_2 = 1745, \mu_3 = 90, \sigma_1^2 = 0.5, \sigma_2^2 = 9, \sigma_3^2 = 4$

Table 32 - Data set 7: Overlapping across 2 attributes for all 3 clusters

Input parameters	

Total # of points = 300
Predefined interval
sets *a-d*
	Group 1 (# of points = 100):
	$\mu_1 = 167, \mu_2 = 539, \mu_3 = 345, \sigma_1^2 = 25, \sigma_2^2 = 81, \sigma_3^2 = 36$
	Group 2 (# of points = 100):
	$\mu_1 = 154, \mu_2 = 500, \mu_3 = 243, \sigma_1^2 = 9, \sigma_2^2 = 64, \sigma_3^2 = 25$
	Group 3 (# of points = 100):
	$\mu_1 = 178, \mu_2 = 520, \mu_3 = 342, \sigma_1^2 = 16, \sigma_2^2 = 100, \sigma_3^2 = 36$

Table 33 - Data set 8: Overlapping across 3 attributes for all 3 clusters for one set of predefined intervals

Input parameters	

Total # of points = 300
 	
Predefined interval
 	
set *a*
	Group 1 (# of points = 100):
	
	$\mu_1 = 190, \mu_2 = 688, \mu_3 = 348, \sigma_1^2 = 12, \sigma_2^2 = 16, \sigma_3^2 = 25$
	
	Group 2 (# of points = 100):
	
	$\mu_1 = 188, \mu_2 = 700, \mu_3 = 345, \sigma_1^2 = 9, \sigma_2^2 = 12, \sigma_3^2 = 16$
	
	Group 3 (# of points = 100):
	
	$\mu_1 = 181, \mu_2 = 690, \mu_3 = 340, \sigma_1^2 = 9, \sigma_2^2 = 16, \sigma_3^2 = 25$

Table 34 - Data set 9: Overlapping across reliability and time for all 3 clusters and for one set of predefined intervals

Input parameters	

Total # of points = 300
 	
Predefined interval
 	
set *a*
	Group 1 (# of points = 100):
	
	$\mu_1 = 190, \mu_2 = 689, \mu_3 = 420, \sigma_1^2 = 12, \sigma_2^2 = 16, \sigma_3^2 = 25$
	
	Group 2 (# of points = 100):
	
	$\mu_1 = 188, \mu_2 = 700, \mu_3 = 398, \sigma_1^2 = 9, \sigma_2^2 = 12, \sigma_3^2 = 16$
	
	Group 3 (# of points = 100):
	
	$\mu_1 = 185, \mu_2 = 695, \mu_3 = 380, \sigma_1^2 = 9, \sigma_2^2 = 16, \sigma_3^2 = 25$

Table 35 - Data set 10: overlapping across time and price for all 3 clusters and for one set of predefined intervals

Input parameters	

Total # of points = 300
 	
Predefined interval
 	
sets *a*
	Group 1 (# of points = 100):
	
	$\mu_1 = 173, \mu_2 = 1250, \mu_3 = 119, \sigma_1^2 = 12, \sigma_2^2 = 16, \sigma_3^2 = 25$
	
	Group 2 (# of points = 100):
	
	$\mu_1 = 160, \mu_2 = 1242, \mu_3 = 113, \sigma_1^2 = 9, \sigma_2^2 = 12, \sigma_3^2 = 16$
	
	Group 3 (# of points = 100):
	
	$\mu_1 = 147, \mu_2 = 1253, \mu_3 = 110, \sigma_1^2 = 9, \sigma_2^2 = 16, \sigma_3^2 = 25$

Table 36 - Data set 11: overlapping across reliability and price for all 3 clusters and for one set of predefined intervals

Input parameters	

Total # of points = 300
 	
Predefined interval
 	
sets *a*
	Group 1 (# of points = 100):
	
	$\mu_1 = 158, \mu_2 = 500, \mu_3 = 345, \sigma_1^2 = 12, \sigma_2^2 = 16, \sigma_3^2 = 25$
	
	Group 2 (# of points = 100):
	
	$\mu_1 = 157, \mu_2 = 540, \mu_3 = 343, \sigma_1^2 = 9, \sigma_2^2 = 12, \sigma_3^2 = 16$
	
	Group 3 (# of points = 100):
	
	$\mu_1 = 156, \mu_2 = 522, \mu_3 = 342, \sigma_1^2 = 9, \sigma_2^2 = 16, \sigma_3^2 = 25$

C. Separated clusters with random data:

Table 37 - Data set 13: Dataset 4 with random points

	Input parameters

Distinct clusters (closely placed) with multiple variance combinations
Total # of points = 350
	Group 1 (# of points = 100):
	$\mu_1 = 155, \mu_2 = 700, \mu_3 = 180, \sigma_1^2 = 64, \sigma_2^2 = 225, \sigma_3^2 = 144$
	Group 2 (# of points = 100):
	$\mu_1 = 170, \mu_2 = 770, \mu_3 = 210, \sigma_1^2 = 25, \sigma_2^2 = 169, \sigma_3^2 = 196$
	Group 3 (# of points = 100):
	$\mu_1 = 180, \mu_2 = 840, \mu_3 = 240, \sigma_1^2 = 9, \sigma_2^2 = 256, \sigma_3^2 = 169$

Random set 1 (50) for dataset13 with pre-defined interval set *a*	[(63-93), (67-95)], [(332-440), (338-447)], [(73-132), (77-138)]
Random set 2 (50) for dataset13 with pre-defined interval set *b*	[(65-94), (72-98)], [(321-431), (334-442)], [(70-130), (75-143)]
Random set 3 (50) for dataset13 with pre-defined interval set *c*	[(62-92), (70-100)], [(325-441), (340-452)], [(66-136), (81-144)]
Random set 4 (50) for dataset13 with pre-defined interval set *d*	[(61-92), (72-100)], [(320-431), (334-451)], [(62-131), (80-143)]

D. ***Overlapping clusters with random data:***

Table 38 - Data set 14: Dataset 5 with random points

	Input parameters

Overlapping across all 3 attributes and clusters
Total # of points = 350
	Group 1(# of points = 100):
	$\mu_1 = 172, \mu_2 = 1200, \mu_3 = 220, \sigma_1^2 = 36, \sigma_2^2 = 100, \sigma_3^2 = 49$
	Group 2(# of points = 100):
	$\mu_1 = 175, \mu_2 = 1207, \mu_3 = 225, \sigma_1^2 = 25, \sigma_2^2 = 144, \sigma_3^2 = 64$
	Group 3(# of points = 100):
	$\mu_1 = 178, \mu_2 = 1217, \mu_3 = 223, \sigma_1^2 = 16, \sigma_2^2 = 196, \sigma_3^2 = 36$

Random set 1 (50) for dataset13 with pre-defined interval set *a*	[(78-93), (80-97)], [(585-627), (590-631)], [(100-110), (102-123)]
Random set 2 (50) for dataset13 with pre-defined interval set *b*	[(76-92), (78-96)], [(581-622), (589-633)], [(91-117), (102-125)]
Random set 3 (50) for dataset13 with pre-defined interval set *c*	[(72-93), (82-99)], [(581-625), (587-637)], [(93-114), (105-127)]
Random set 4 (50) for dataset13 with pre-defined interval set *d*	[(72-92), (80-99)], [(577-615), (591-635)], [(89-113), (107-131)]

Table 39 - Data set 15: Dataset 6 with random points

	Input parameters

Overlapping across all 3 attributes for 2 clusters
 	
Total # of points = 400
	Group 1 (# of points = 150):
	
	$\mu_1 = 150, \mu_2 = 210, \mu_3 = 280, \sigma_1^2 = 25, \sigma_2^2 = 16, \sigma_3^2 = 9$
	
	Group 2 (# of points = 100):
	
	$\mu_1 = 140, \mu_2 = 212, \mu_3 = 275, \sigma_1^2 = 25, \sigma_2^2 = 16, \sigma_3^2 = 9$
	
	Group 3 (# of points = 100):
	
	$\mu_1 = 133, \mu_2 = 1745, \mu_3 = 90, \sigma_1^2 = 0.5, \sigma_2^2 = 9, \sigma_3^2 = 4$

Random set 1 (50) for dataset13 with pre-defined interval set *a*	[(62-84), (64-85)], [(95-875), (100-879)], [(40-143), (43-147)]
Random set 2 (50) for dataset13 with pre-defined interval set *b*	[(61-80), (66-87)], [(92-875), (100-883)], [(33-141), (44-150)]
Random set 3 (50) for dataset13 with pre-defined interval set *c*	[(60-81), (66-86)], [(88-875), (100-888)], [(31-143), (44-155)]
Random set 4 (50) for dataset13 with pre-defined interval set *d*	[(58-79), (61-89)], [(87-873), (100-891)], [(28-141), (43-159)]

④ BL-61-342