Missouri S&T Library and Learning Resources

Scholars' Mine

Doctoral Dissertations | Student Theses and Dissertations

Fall 2015

Methods and algorithms for service selection and recommendation (preference and aggregation based)

Kenneth Kofi Fletcher

Follow this and additional works at: https://scholarsmine.mst.edu/doctoral_dissertations

 Part of the Computer Sciences Commons

Department: Computer Science

Recommended Citation

Fletcher, Kenneth Kofi, "Methods and algorithms for service selection and recommendation (preference and aggregation based)" (2015). *Doctoral Dissertations*. 2445.
https://scholarsmine.mst.edu/doctoral_dissertations/2445

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

www.manaraa.com

METHODS AND ALGORITHMS FOR SERVICE SELECTION AND RECOMMENDATION (PREFERENCE AND AGGREGATION BASED)

by

KENNETH KOFI FLETCHER

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2015

Approved
Xiaoqing (Frank) Liu, Advisor
Maggie X. Cheng
Wei Jiang
Sriram Chellappan
Frank Liou

www.manaraa.com

© 2015
Kenneth Kofi Fletcher
All Rights Reserved

www.manaraa.com

ABSTRACT

In order for service users to get the best service that meets their requirements, they prefer to personalize their non-functional attributes, such as reliability and price. However, the personalization makes it challenging because service providers have to deal with conflicting non-functional attributes when selecting services for users. In addition, users may sometimes want to explicitly specify their trade-offs among non-functional attributes to make their preferences known to service providers. Typically, users' service search requests with conflicting non-functional attributes may result in a ranked list of services that partially meet their needs. When this happens, it is natural for users to submit other similar requests, with varying preferences on non-functional attributes, in an attempt to find services that fully meet their needs. This situation produces a challenge for the users to choose an optimal service based on their preferences, from the multiple ranked lists that partially satisfy their request.

Existing memory-based collaborative filtering (CF) service recommendation methods that employ this recommendation technique usually depend on non-functional attribute values obtained at service invocation to compute the similarity between users or items, and also to predict missing non-functional attributes. However, this approach is not sufficient because the non-functional attribute values of invoked services may not necessarily satisfy their personalized preferences.

The main contributions of this work are threefold. First, a novel service selection method, which is based on fuzzy logic, that considers users' personalized preferences and their trade-offs on non-functional attributes during service selection is presented. Second, a method that aggregates multiple ranked lists of services into a single aggregated ranked list, where top ranked services are selected for the user is also presented. Two algorithms were proposed: 1) Rank Aggregation for Complete Lists (RACoL), that aggregates complete ranked lists and 2) Rank Aggregation for Incomplete Lists (RAIL) to aggregate incomplete ranked lists. Finally, a CF-based service recommendation method that considers users' personalized preference on non-functional attributes if proposed. Examples using real-world services are presented to evaluate the proposed methods and experiments are carried out to validate their performance.

www.manaraa.com

ACKNOWLEDGMENTS

I am indebted to many people and wish to thank all who have helped me through the course of this project. First of all, I thank my advisor, Dr. Xiaoqing (Frank) Liu, who has encouraged and challenged me throughout the research. Secondly, I also want to thank Drs. Maggie X. Cheng, Wei Jiang, Frank Liou and Sriram Chellappan for serving on my dissertation committee and taking time to review this work.

I am most grateful to my wife, Esi Adeborna, my parents Mr. and Mrs. Albert Fletcher, and my sister, Mrs. Alberta Ocran, for their encouragement and support. I also want to thank my friends, Prof. Richard Amankwah of the University of Mines and Technology, Ghana, and Mr. and Mrs. Robert Bowers. Finally, I want to show my appreciation to my grandparents, Mr. and Mrs. Dennis Meisinger. May the good Lord bless us all.

www.manaraa.com

TABLE OF CONTENTS

www.manaraa.com

www.manaraa.com

www.manaraa.com

LIST OF ILLUSTRATIONS

www.manaraa.com

www.manaraa.com

LIST OF TABLES

www.manaraa.com

www.manaraa.com

1. INTRODUCTION

Services technology is well recognized as an easy way to integrate applications without boundaries. Owing to this, most organizations tend to publish their services on the web for easy consumption by the public. This has exponentially increased the number of available services. Different service providers may offer services that are equivalent with respect to their functionality. Therefore, it becomes more challenging for users to select the services that best meet their requirements. Rather than selecting services based only on their functionality, users are increasingly paying more attention to non-functional attributes, such as *reliability*, *availability*, *reputation*, and *price*. This is because, non-functional attributes provide a distinction among the competing services with similar functionality; allowing prospective users to choose the services which best suit their requirements. Using non-functional attributes during service selection however, presents service providers with certain challenges. Firstly, users' service requirements are becoming personalized. Personalization here, describes how different users have different preferences (values) for the same non-functional attribute. It therefore becomes challenging to incorporate their personalized preferences on non-functional attributes. While most non-functional attributes may be necessary not all users prefer the same number of attributes for a particular service. Secondly, users' requests sometimes contain conflicting non-functional attributes which makes it challenging to completely meet their preferences for such requests. Conflicting non-functional attributes are those attributes where an increase in the satisfaction of one often decreases the satisfaction of the other. As a result of this conflicting relationship that may exist between non-functional attributes, users may want to explicitly trade-off some non-functional attributes for others.

Usually, a user's personal preference for a service comprises of his/her preference for non-functional attribute(s) that describe the service. These non-functional attributes may be conflicting [1, 2], resulting in ranked list of services that partially meet the user's preference. In an attempt to obtain services that completely meet his/her request, the user may submit similar multiple service requests. Here, similar requests are those requests with the same functionality and trade-offs on non-functional attributes, but with varying

www.manaraa.com

preference on those non-functional attributes. These requests may also yield ranked lists that do not fully satisfy the user's preference due to the same conflicting relationship that may exists among non-functional attributes. In such a situation, it becomes challenging for the user to compare the different ranked lists in order to choose the optimal service. This is because, each ranked list may contain huge number of services that makes it time consuming for users to compare them against services in other ranked lists. A naïve way for this comparison will be to merge the different sets of ranked lists into one list and rank the merged list with respect to the user's search intent. However, this naïve way also has its own challenges: 1) the same service may rank differently in other ranked lists and therefore becomes hard to determine its overall rank; and 2) some of the ranked lists may contain services that do not appear in other ranked lists due to the varying requests, making it difficult to determine the overall rank of such services. In order for the user to obtain an optimal service, based on the ranked lists, there is the need for a method that will produce an aggregated ranked list that addresses the challenges mentioned above.

The increase in the number of services over the internet has inundated service users with many choices. For instance, Netflix.com has over 17,000 movies in its selection, and Amazon.com has over 410,000 titles in its Kindle store alone [3]. In order to reduce the number of choices users can decide on, recommendation systems are necessary. Recommendation systems are attracting lots of attention because they provide users with prior knowledge of candidate choices to deal with information overload on the Web. They have been used to recommend books and CDs at Amazon.com, movies at Netflix.com, and news at VERSIFI Technologies [4].

Collaborative filtering (CF) is one of the widely used service recommendation techniques that bases its recommendations on the ratings or behavior of other users in the system [3]. Intuitively, it assumes that, if users agree about the quality or relevance of some service items, then they will likely agree about other service items as well. Existing memory-based CF techniques accomplish this by computing the similarity between users or service items using non-functional attribute values obtained at service invocation. However using non-functional attribute values of invoked services alone gives inaccurate similarity measure. This is because, the invoked services are typically the final choice (including any trade-offs) of users and may not necessarily satisfy their personalized

www.manaraa.com

preferences. They represent the "end" of users' service selection process and therefore when the non-functional attribute values of the invoked services used for similarity computation, they do not reflect users' personalized preferences on those non-functional attributes. Therefore, rather than focusing solely on the "end", recommendation systems must include the "means to the end" (users personalized preference) during service recommendation.

The non-functional attribute values observed by users during service invocation may not necessarily represent their satisfaction for that service. For this reason, disregarding the personalized preferences of users in similarity computation creates a gap between users' non-functional attribute value and their satisfaction. Users' personalized preferences ensures that the non-functional attribute closely aligns with their satisfaction, bridging that gap and resulting in similarity values that accurately depicts the similar relationship between two users. Intuitively, if a non-functional attribute value used in similarity computation fails to satisfy a user's personalized preference it in turn produces similarity results that are inaccurate. Thus, to accurately recommend services, which are personalized to users, it is necessary for recommendation systems to incorporate users' personalized preferences on non-functional attributes when recommending services to an active user.

A great deal of work has been done to bring attention to service discovery and selection based on non-functional attributes. Much of this work has produced similar ideas [5, 6, 7, 8, 9]. For example, many researchers have proposed that a user always give precise, quantitative constraints and preferences on each non-functional attribute. Others rely on weighted summation functions to aggregate all non-functional attributes to rank services for selection. These studies have the following shortcomings: 1) They do not allow users to specify either personal preferences or associated weights, based on elastic non-functional attributes, using linguistic terms (i.e. English), which are more practical. 2) They do not take into consideration the relationships among non-functional attributes which may lead to inappropriate trade-offs among non-functional attributes.

In this dissertation, users' personalized preferences on non-functional attributes and their trade-offs to select services that best satisfy their needs is considered. A method that takes users' personalized trade-off preferences and linguistic weights on non-

www.manaraa.com

functional attributes as inputs for service search is developed. Top ranked services in the search results are then selected. Two algorithms to compute an aggregated ranked list from each ranked list of services is also proposed. The first algorithm, **R**ank **A**ggregation for **Co**mplete **L**ists (RACoL), aggregates complete ranked lists (ranked lists given by total orders). There are however instances where the ranked lists to be aggregated come with incomplete ranked lists (incomplete orderings); i.e. some ranked list(s) contain services that do not appear in other ranked list(s). The second algorithm, **R**ank **A**ggregation for **I**ncomplete **L**ists (RAIL), is proposed to aggregate incomplete ranked lists.

A method that considers users' personalized preferences, in addition to the non-functional attribute values of invoked services, to accurately recommend service(s) to an active user is also proposed. The proposed method, accurately compute the similarity between users or service items by incorporating users' personalized preferences on non-functional attributes in our similarity function. The enhanced similarity function firstly, identify whether the two service users or items share some past experiences. If they do, the widely used Pearson Correlation Coefficient [3, 4, 10, 11] is extended to include satisfaction of users' personalized preferences on non-functional attributes. Otherwise, the similarity between the user's preferences is computed. Based on the similarity values, the top-k algorithm is employed to find similar neighbors. Finally, to predict missing non-functional attribute values, the weighted average with mean offset is extended to incorporate users' satisfaction on non-functional attributes based on their personalized preferences.

Examples using real-world services to evaluate the method are presented. It can be seen that rank aggregation results from the proposed method closely represent the sets of ranked lists than using alternative approaches. Experiments were also carried out to validate their performance.

www.manaraa.com

2. PERSONALIZED PREFERENCE AND TRADE-OFF BASED SERVICE SELECTION (PPTSS)

2.1. BACKGROUND AND RELATED WORK

In this section, related work regarding service selection methods based on non-functional attributes as well as fuzzy logic [12] are discussed.

2.1.1. Service Selection Based on Non-Functional Attributes. Following Ran's [13] work, the problem of both service discovery and service selection with respect to non-functional attributes has received lots of attention in the service computing community. Service selection is heavily based on ranking services according to their non-functional attributes. Combining multiple non-functional attributes make service selection a difficult task as users struggle to find the right service with an optimal combination of non-functional attributes.

Masri and Mahmoud [7] employed a variation of weighted summation of non-functional attributes to rank services for selection. They first normalize values of different non-functional attributes into a range. Then compute the overall satisfaction of the services by summing the normalized values. Services are ranked based on overall quality. Similarly, Comuzzi and Pernici [14] used a price model to combine multiple non-functional attributes. This price model converts each non-functional attribute of a service to a price and then adds all of the prices together. The services are then ranked according to their total prices. Benouaret et al. [15] proposed two concepts, σ- and α-dominant skylines, to improve the skyline, a concept for selecting web services based on non-functional attributes. They identified two skyline requirements, size and quality, for which σ- and α-dominant skylines were their respective solutions. Benouaret et al. [16], in another work, proposed a majority-rule-based web service selection. Although their approach does not explicitly consider users' non-functional attributes, it considers their overall preference on a service. They formulated the majority-rule-based service selection based on the dominance relationship and skyline and also proposed an algorithm that is both efficient and returns a more manageable set of services. Yau and Yin [17] proposed a service ranking and selection method which can support a more flexible non-functional

www.manaraa.com

attribute specification. They modeled the relationship among services' non-functional attributes and satisfaction scores.

Sun et al. [18] proposed a personalized Web service recommendation method based on a novel collaborative filtering (CF) approach. The method takes advantage of the little non-functional attribute information available. They employ non-functional attribute information from similar users with similar experience on the same non-functional attribute to automatically predict non-functional attribute values. Chen et al. [19] also proposed a region-based hybrid CF algorithm to predict non-functional attribute values of services for service recommendation. In their work, they discovered that a user's location greatly influences the accuracy of their prediction. Sun et al. [20] again presented a new similarity measure for Web service similarity computation and propose a novel collaborative filtering approach, called normal recovery collaborative filtering, for personalized Web service recommendation.

2.1.2. Fuzzy Logic Service Selection Methods. Traditional non-functional attribute-driven service selection methods require crisp and precise constraints and preferences on non-functional attributes from users. Examples are "the response time should be less than 1 second" and the preference degree for response time is 0.9". Such specifications are not natural and practical to users in many cases. Instead, users may like to use fuzzy logic and linguistic terms [12, 21] to represent their non-functional attributes, such as "the response time should be Short".

Wei-Lin et al. [22] proposed a fuzzy consensus on non-functional attributes in web services discovery approach based on fuzzy sets [12]. Their objective was to build consensus on non-functional attributes between service providers and consumers. Their focus was on aggregating similarities between non-functional attributes from the provider's and consumer's perspectives.

Wang [23] extends the Max-Min-Max composition of intuitionistic fuzzy sets (IFS) [21]. He categorized non-functional attributes properties of web services into functional and non-functional properties. His approach deals with the decision maker's imprecise perceptions under incomplete information. It also objectively determines the weights of non-functional attributes. Determining of weights from users requirements may result in assigning weights on non-functional attributes inappropriately. This is

www.manaraa.com

because user requirements are fuzzy and sometimes the users themselves do not know exactly what they want. Wang [24] also designed a service selection model that takes into account its non-functional attributes based on fuzzy linear programming (FLP) technologies. This was to identify service alternatives dissimilarities and assist service users in selecting most suitable services with consideration of their expectations and preferences. Xiuqin et al. [25] employed interval-valued intuitionistic fuzzy soft set theory for solving web service selection problems that take into account users' non-functional attributes. Almulla et al. [26] explored the dependencies between quality factors to improve the weights given by a user. However, this method is also based on the classical weighted summation (or average) function to rank services. This ranking method does not allow representation of personalized trade-offs among non-functional attributes in many cases. Li et al. [27] proposed a model for web service selection based on fuzzy quality of service (QoS) attributes. In their model, they classified QoS information into several multi-dimensional classes. Then based on these classes, a synthetic service selection method is used to rank is used to rank the services.

A systematic approach for specifying non-functional requirements of contracts for quality management and evaluation has been proposed by Liu and Yen [28] and Liu et al. [29]. Their work considers both qualitative and quantitative specification techniques of non-functional requirements. Also, they introduce both the crisp and elastic non-functional requirements specification. This paper adopts those concepts and is the first to apply them to service selection based on both personalized preferences and trade-offs.

The works discussed in this section do not take either relationships or personalized trade-offs among non-functional attributes into consideration. Therefore, such works lack the ability to implicitly support personalized non-functional attribute tradeoffs. In addition, personalization is defined in terms of 1) linguistic terms, 2) membership functions, and 3) importance on non-functional attributes. This is to fully capture the personalized nature of a user's request and subsequently select the right services to meet the user's request.

www.manaraa.com

2.2. THE SERVICE SELECTION METHOD

The proposed service selection method is described in this section. Some basic notation, used in this paper, is also provided. Finally, the formalization of the proposed service selection method is given in this section.

Figure 2.1 shows the framework of the proposed personalized preference and trade-off based service selection. The method assumes that all service users are rational. The service selection process begins by a user submitting a new service request. The service request includes both the functional requirements and the non-functional requirements. The input of the non-functional requirements is in two parts. First, the user selects his/her preferred non-functional attributes and then specifies their satisfaction for each attribute. Next, with these attributes, the user then specifies his/her trade-off strategy for selection. The trade-off strategy includes the user's personalized elastic non-functional attributes, weights and required aggregation operators, which will be discussed in detail in Section 2.4. All of these are captured by the *input handler*.

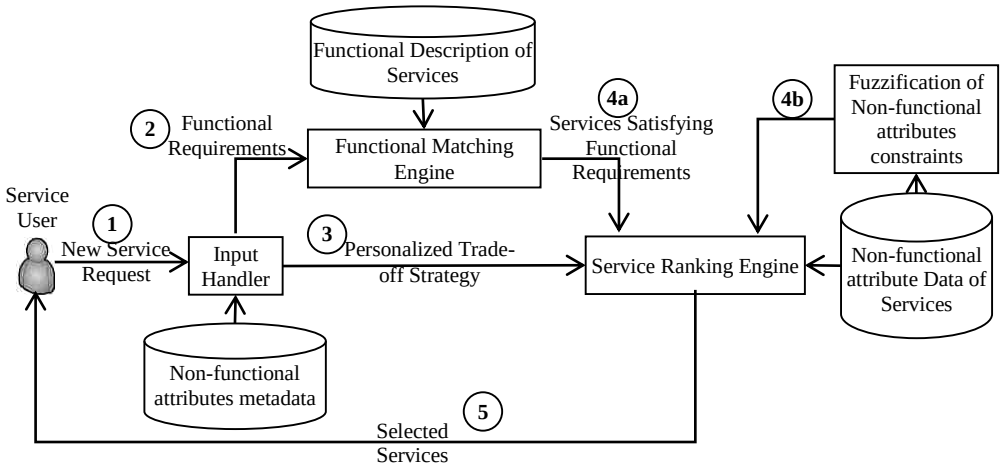

Figure 2.1. Framework of the personalized preference and trade-off based service selection

The *input handler* processes the request, extracting both the functional and non-functional requirements. The *functional matching engine* uses the functional requirements and information from the *functional description of services* repository to produce a list of

www.manaraa.com

services satisfying user's functional requirements. For this purpose, the algorithm by Sajjanhar et al. [30] was employed. The non-functional requirements, in the form of a trade-off strategy, are pushed over to the *service ranking engine*. The list of services matching user's functionality is also sent to the *service ranking engine* for evaluation based on the trade-off strategy.

The evaluation by the *service ranking engine* uses as inputs: the list, user specified non-functional requirements, linguistic fuzzified non-functional attribute constraints and non-functional attribute data. With these inputs, the *service ranking engine* computes the satisfaction degree for each service, by defuzzifying the linguistic satisfactions provided by the user using fuzzy propositions.

The *service ranking engine* also defuzzifies the linguistic weights of each non-functional attribute using the Centroid Method (CM) [31]. The CM defuzzifies the weights supplied by the user in linguistic terms.

Finally, using fuzzy connectives like fuzzy conjunction, fuzzy disjunction and fuzzy compromise, the service ranking engine computes the overall satisfaction degree of aggregated non-functional requirements for all services. The services are then ranked according to the overall satisfaction degrees. Services with high overall satisfaction degrees (top-ranked) are presented to the user for selection. Figure 2.2 illustrates this detailed process by the *service ranking engine* to complete its task.

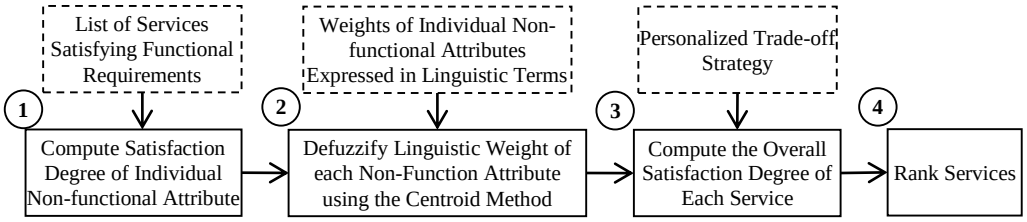

Figure 2.2. The service ranking engine process

Users' personalized non-functional attribute requirements are defined first in terms of linguistic terms, and then by users' membership function and finally users' importance on non-functional attribute.

www.manaraa.com

Assume a list of services satisfying a user's functional requirements $S = \{s_1, s_2, s_3, \ldots, s_n\}$, and a list of user's preferred non-functional attributes $N = \{nfa_1, nfa_2, \ldots, nfa_m\}$. Let $\sigma_{S_i}^{N_j}$ denote the satisfaction of non-functional attribute, N_j with respect to service S_i, where σ is the linguistic term that typifies the satisfaction of N_j. For instance, $High_{S_i}^{availability}$, may be used by a user to describe their satisfaction on *availability* non-functional attribute as being *high*.

Next the satisfaction degree of the linguistic term by a membership function is specified. Some users may have some difficulty to specify their membership functions. Due to this, membership functions were categorized into two (2) main categories depending on the maximum satisfaction of the associated non-functional attribute value. One of the main categories supports non-functional attributes whose higher values produces higher satisfaction (e.g. *reliability* and *availability* non-functional attributes). The second main category of membership function supports non-functional attributes whose lower values produces higher satisfaction (e.g. *response time* and *reputation* non-functional attributes). In this way, users need not to specify the entire membership function but the selection system will generate those functions for users based on the maximum and minimum values they provide. The categories are as follows:

1. Category 1: higher non-functional attribute values preferred (the higher, the better), e.g. *reliability*, and *availability* non-functional attributes; and
2. Category 2: lower non-functional attribute values preferred (the lower, the better), e.g. *response time* and *price* non-functional attributes.

Definition (Satisfaction Function). Given *S* and *N*, the satisfaction of a non-functional attribute, N_j, with respect to a service, S_i, is defined as

$$\sigma_{S_i}^{N_j} = \begin{cases} \alpha, & Category\ 1 \\ \beta, & Category\ 2 \end{cases} \quad (1)$$

where α and β are the membership functions for category 1 and category 2 respectively which are defined as

www.manaraa.com

$$\alpha = \begin{cases} 0 & , \quad if\ N_j \leq minimum \\ 1 & , \quad if\ N_j \geq maximum \\ \dfrac{N_j - minimum}{maximum - minimum}, & otherwise \end{cases} \tag{2}$$

$$\beta = \begin{cases} 0 & , \quad if\ N_j \geq maximum \\ 1 & , \quad if\ N_j \leq minimum \\ \dfrac{maximum - N_j}{maximum - minimum}, & otherwise \end{cases} \tag{3}$$

where *minimum* and *maximum* are the highest and lowest satisfaction degrees of the non-functional attribute, N_j, as specified by the user.

Definition (Personalized Non-Functional Attribute). Let $U = \{u_1, u_2, \dots, u_m\}$ be a set of service users, and W_{U_i,N_j} be a weighting factor of user $U_i's$ linguistic importance on a non-functional attribute N_j. Some possible linguistic weights are *extremely important*, *very important*, *important* and *somewhat important*. Details of all the linguistic weights used in this work are discussed in Section 2.5. The personalized non-functional attribute, $P_{U_i}^{N_j}$, for a user U_i on a non-functional attribute N_j is a membership function *MF* with the weighting factor W_{U_i,N_j} given as

$$P_{U_i}^{N_j} = MF(U_i, N_j) \times W_{U_i,N_j} \tag{4}$$

Definition (Overall Trade-Off Strategy). The overall personalized trade-off strategy (requirement), R_{U_i}, for a user U_i, can be described using individual personalized non-functional attributes, $P_{U_i}^{N_j}$, and an aggregation operator (discussed in detail in Section 2.4.2), $\coprod$, as follows:

$$R_{U_i} = \coprod_{j=1}^{m} P_{U_i}^{N_j} \tag{5}$$

where

www.manaraa.com

$\coprod$ is either of the fuzzy connective operators $\wedge$, $\vee$, or $\otimes$

For instance, let $N = \{reliability,\ price,\ reputation\}$ and $P_{U_i}^{reliability}$, $P_{U_i}^{price}$, and $P_{U_i}^{reputation}$ be the preferences for reliability, price and reputation non-functional attributes respectively for user U_i, then R for the user can be

$$R = P_{U_i}^{reliability} \wedge P_{U_i}^{price} \otimes P_{U_i}^{reputation}$$

Definition (Service Selection). Given *S*, *N*, *W* and *R*, the service selection process can be modelled as a ranking in terms of the satisfaction of requirement *R* so that for any two services S_i and S_j the following is true.

$$S_i \succ S_j \Leftrightarrow Sat_R(S_i) \geq Sat_R(S_j) \quad (6)$$

where $Sat_R(S_i)$ represents the satisfaction of service S_i with respect to some user requirement *R*.

2.3. PERSONALIZED INDIVIDUAL NON-FUNCTIONAL ATTRIBUTE REQUIREMENT SPECIFICATION USING FUZZY PROPOSITIONS

This section discusses how to specify individual non-functional requirements in terms of non-functional attributes. The specification is done using fuzzy proposition, a statement in fuzzy logic which is satisfied to a degree, and linguistic terms. Some linguistic terms used are ‘high’, ‘affordable’, ‘good’, ‘short’ and ‘few’. The membership function of these linguistic terms in fuzzy logic typifies satisfaction of some non-functional attributes.

The non-functional attributes considered in the illustrative example include *reliability*, *availability*, *throughput*, *response time*. A service user may specify their non-functional requirements on, for instance, service *response time* (the lower the value, the better the non-functional attribute) as follows: The response time for a prospective

www.manaraa.com

service must be ‘high’. Figure 2.3 shows the membership function of ‘high’ that satisfies the *response time* non-functional attribute.

On the vertical axis in Figure 2.3, 0 and 1 represent the lowest and highest levels of service satisfaction, respectively, in terms of response time. If the service responds, on average, 3 secs or lower, the non-functional attribute is understood to be met. If the response time is increased from 3 secs to anything above 10 secs, the service satisfaction reduces accordingly. However, if the response time is greater than the threshold of 10 secs, the satisfaction degree is 0 and the service is completely unacceptable.

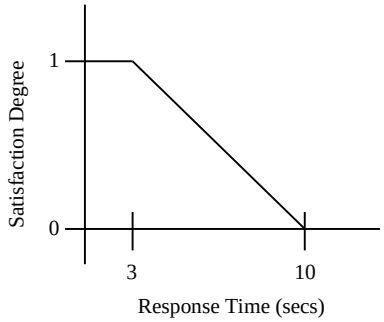

Figure 2.3. Membership function for *Response Time* non-functional attribute

The membership functions of the non-functional attributes that are considered in the airline application case study is presented in Section 2.7.2. They include *price (airline price)*, *reputation*, *duration*, and *number-of-stops*.

The *reputation* of an airline in this work is based on the Airline Quality Rating (AQR) [32]. The AQR is an objective method for assessing airline quality on combined multiple performance criteria [32]. The formula for calculating the AQR score is:

$$\text{AQR}=\frac{(+8.63\times OT)+(-8.03\times DB)+(-7.29\times MB)+(-7.17\times CC)}{(8.63+8.03+7.29+7.17)} \quad (7)$$

where OT (On-Time), DB (Denied Boarding), MB (Mishandled Baggage), and CC (Customer Complains) are variables considered. Data for all criteria is drawn from the

www.manaraa.com

U.S. Department of Transportation's monthly Air Travel Consumer Report[1] [32]. The AQR values used in this work are based on the April 2012 reported values. Higher AQR values indicate excellent reputation. AirTran Airways (FL) for example, had the best rating in 2011 with an AQR value of -0.48. Since most normal users are not aware of AQR, the ranking of an airline which is based on its AQR value was used to denote its reputation. For instance, in the 2012 AQR reported values, out of 14 airlines, AirTran Airways (FL) ranked first (1^{st}) and American Eagle (MQ) ranked fourteenth (14^{th}). Figure 2.4 is the membership function of 'medium' that satisfies the *reputation* non-functional attribute.

The *number-of-stops* of a flight indicates the number of different flights (which have different flight numbers) that makes up any flight between two cities by an airline. For instance, consider a flight from St. Louis to New York which goes through Memphis and Atlanta. Assume that flights from St. Louis to Memphis, Memphis to Atlanta and Atlanta to New York all have different flight numbers. Then the *number-of-stops* for this flight is 2 because it stops at Memphis and Atlanta.

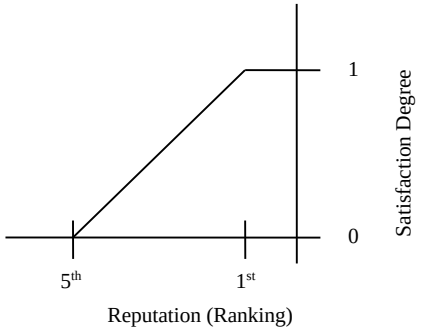

Figure 2.4. Membership function for *Reputation* non-functional attribute

[1] http://dot.gov/airconsumer/

www.manaraa.com

2.4. PERSONALIZED SERVICE TRADE-OFFS

The relationships among a service's non-functional attributes are extremely important when considering trade-offs. These relationships reveal the interaction among non-functional attributes. Additionally, selecting a suitable aggregation operator for the aggregation of non-functional attributes depends on these relationships. The relationships that exist among non-functional attributes are discussed and aggregation operators based on the relationships are presented.

2.4.1. Relationships Among Non-Functional Attributes. For any two non-functional attributes, there exist some relationship. The relationship between any two non-functional attributes are classified into three (3) different types: *conflicting*, *cooperative*, and *mutually exclusive* [28]. These relationships are based on the outcome or impact on the satisfaction degree of one non-functional attribute when the satisfaction degree of another non-functional attribute changes. In addition, the relationship between two non-functional attributes is based on the published services and their non-functional attributes value.

Conflicting Non-Functional Attributes ($\ominus$). Two non-functional attributes are said to be conflicting if an increase in the satisfaction degree of one often decreases the satisfaction degree of the other. If an increase in the satisfaction degree of one non-functional attribute always decreases the satisfaction degree of the other, they are said to be completely conflicting [28].

Cooperative Non-Functional Attributes ($\oplus$). Contrary to conflicting non-functional attributes, two non-functional attributes are referred to as cooperative if an increase in the satisfaction degree of one often increases the satisfaction degree of the other. If an increase in the satisfaction degree of one non-functional attribute always leads to an increase in the satisfaction degree of the other, they are said to be completely cooperative [28].

Mutually Exclusive Non-Functional Attributes ($\odot$). It is typical that two non-functional attributes cannot be satisfied at all at the same time. That is, if the satisfaction degree of one non-functional attribute is satisfied to a certain degree, the other cannot be satisfied at all, and vice versa. When this occurs, they are considered to be mutually exclusive non-functional attributes.

www.manaraa.com

To show an example of mutually exclusive non-functional attributes, consider a scenario where a user wants an airline service that costs not more than $200, gets to their destination in not more than 3 hours and has a reputation less than the 5th ranking airline. Assume that Table 2.1 is the list of services satisfying the user's functional requirement. Services 2 and 3 satisfy this user's cost non-functional attribute. However, it can be seen that neither the duration nor reputation non-functional attribute of either service 2 or 3 can be satisfied at the same time. In such a situation, the non-functional attribute duration and reputation are mutually exclusive.

Table 2.1. List of Services Satisfying User's Functionality

Service	Cost($)	Duration (hrs)	Reputation (ranking)
1	215	2.70	1st
2	195	2.80	7th
3	187	3.50	3rd

For a service request that includes multiple non-functional attributes, it may be challenging to satisfy all attributes to their highest degrees. This is due to the relationship that exists between any two non-functional attributes. Thus, trade-offs among them are desirable. Typically, it has been observed that each of the following pairs of non-functional attributes; (*reliability* and *availability*) and (*throughput* and *response time*) are cooperative [McCall 2002]. Also, (*reliability* and *throughput*), (*reliability* and *response time*), (*availability* and *throughput*), and (*availability* and *response time*) are conflicting non-functional attributes [33] (see Figure 2.5). It must be noted that the relationships are application domain dependent and also depend on the published services and their non-functional attributes value. Therefore a relationship that holds in an airline domain may not necessarily hold in healthcare domain.

2.4.2. Aggregating Non-Functional Attributes Using Fuzzy Connectives. Multiple non-functional attributes must be aggregated based on the relationships that exist among them to obtain an overall non-functional attribute satisfaction value. This is

www.manaraa.com

achieved by employing aggregation operators. Three of such operators, fuzzy compromise [28], fuzzy conjunction [12], and fuzzy disjunction [12], are discussed in this section.

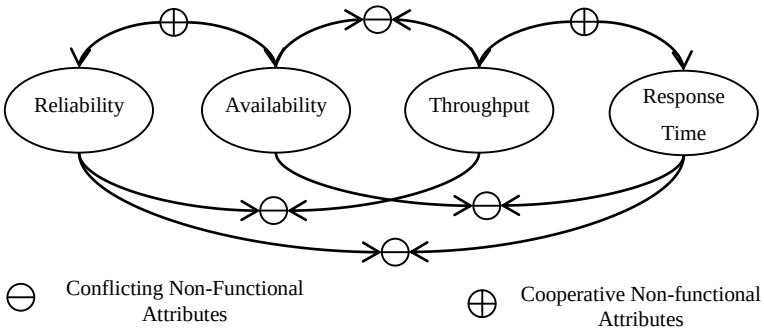

Figure 2.5. Typical relationships among non-functional attributes

Fuzzy Compromise Operator (⊗). Consider the following set of non-functional attributes: N_1, N_2, …,N_m. If the relationship among them is conflicting, then they should be combined with the fuzzy compromise operator. The resulting compromise is the minimal and maximal degree of the membership function. The operator AVERAGE, which is an example of a fuzzy compromise operator, is used in this work. Let S = {s_1, s_2, …,s_n} be the set of services. If both N_1 and N_2 are two non-functional attributes of a service (S_i), then the resulting trade-off value using the AVERAGE operator is

$$N_1(S_i) \otimes N_2(S_i) = \frac{N_1(S_i) + N_2(S_i)}{2} \tag{8}$$

Fuzzy Conjunction Operator (∧). Cooperative non-functional attributes can be satisfied at the same time and hence, fuzzy conjunction operator becomes a suitable operator to combine them. Consider the following set of non-functional attributes: N_1, N_2, …,N_m. If the relationship among them is cooperative, then they should be combined with the fuzzy conjunction operator. The operator MIN, which is an example of a fuzzy

www.manaraa.com

conjunction operator, is used in this work. Let S = {s_1, s_2, …,s_n} be the set of services. If both N_1 and N_2 are two non-functional attributes of a service (S_i), then the resulting trade -off value using the MIN operator is

$$N_1(S_i) \wedge N_2(S_i)=\text{MIN}\{N_1(S_i), N_2(S_i)\} \quad (9)$$

Fuzzy Disjunction Operator (∨). The fuzzy disjunction operator serves as an efficient way to combine mutually exclusive non-functional attributes as they cannot all be satisfied at the same time. Consider the following set of non-functional attributes: N_1, N_2, …,N_m. If the relationship among them is mutually exclusive, then they should be combined with the fuzzy disjunctive operator. The operator MAX, which is an example of a fuzzy disjunction operator, is used in this work and is defined as follows. Let S = {s_1, s_2, …,s_n} be the set of services. If both N_1 and N_2 are two non-functional attributes of a service (S_i), then the resulting trade-off value using the MAX operator is

$$N_1(S_i) \vee N_2(S_i)=\text{MAX}\{N_1(S_i), N_2(S_i)\} \quad (10)$$

2.5. DEFUZZIFICATION OF LINGUISTIC WEIGHTS

The Centroid Method, also known as either the center of gravity (CoG) or center of area (CoA) method, is the most commonly used defuzzification technique. This technique which provides a crisp value based on the center of gravity of the fuzzy set [31]. It also determines the best point for dividing the fuzzy set into exactly two masses. Because weights of non-functional attributes, in this work, are specified using linguistic terms (which can be decomposed in a triangular shape), the centroid method becomes a very suitable approach for defuzzifying the linguistic weight terms. The centroid method is a weighted average method in which the membership function is used for weighting [31]. For a triangular fuzzy number F = (μ, λ, ρ), the weighted value (ω) can be calculated as:

$$\omega=\lambda+\frac{(\mu-\lambda)+(\rho-\lambda)}{3} \quad (11)$$

www.manaraa.com

In this work, seven linguistic terms are decomposed into triangular fuzzy numbers using the triangular fuzzy set shown in Figure 2.6. These linguistic terms are tabulated in Table 2.2 and are provided for assigning the weights to each non-functional attribute. The weight value of each of the linguistic term is calculated using Equation (11).

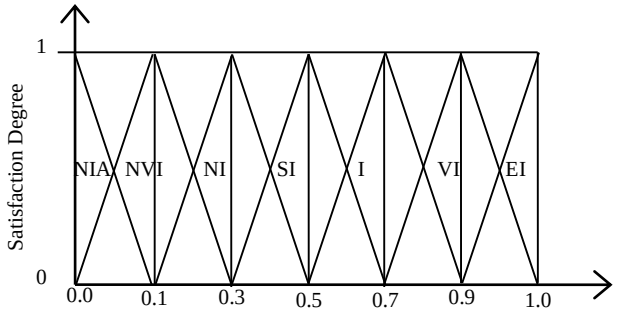

Figure 2.6. Triangular membership functions for the seven linguistic terms

Table 2.2. The Seven Linguistic Terms, Their Fuzzy Numbers, and Corresponding Importance Value

Linguistic Term	Triangular Fuzzy Number	Importance Value
Extremely Important (EI)	(0.9, 1.0, 1.0)	0.97
Very Important (VI)	(0.7, 0.9, 1.0)	0.87
Important (I)	(0.5, 0.7, 0.9)	0.70
Somewhat Important (SI)	(0.3, 0.5, 0.7)	0.50
Not Important (NI)	(0.1, 0.3, 0.5)	0.30
Not Very Important (NVI)	(0.0, 0.1, 0.3)	0.13
Not Important At All (EL)	(0.0, 0.0, 0.1)	0.03

2.6. ILLUSTRATIVE EXAMPLE

An example to illustrate the service selection method is presented. In this example, five (5) users want to use an accounting software service to manage their business finances. This is show detail process of how the proposed service selection method works. Our choice of accounting application is the fact that unlike the airline

www.manaraa.com

application presented later in Section 2.7.2 (which considers domain-depended non-functional attributes), the accounting application considers infrastructural non-functional attributes. This is to demonstrate that the proposed service selection method works for other domains with different non-functional attribute dimension. The accounting software should support automated banking, invoicing, and reporting. One area of concern is good pricing. Additional key quality factors should include *reliability*, *availability*, *response time*, and *throughput*. Each user submits their service request. The request includes both the functionality of the service and the personalized trade-off strategy, as tabulated in Table 2.3. Each non-functional attribute and its associated weight are specified using the following notation: NFA_{Name}^{Weight}. For instance, NFA_{Price}^{EI} indicates an *extremely important* weight, on a *price* non-functional attribute. The trade-off strategy is specified using logical AND (∧), logical OR (∨), or COMPROMISE (⊗) operators. After the functionality search in the data repository is completed, five (5) services satisfying the accounting functionality is obtained as shown in Table 2.4.

Table 2.3. List of Service Requests from 5 Different Users

User	Functionality	Fuzzy logic-based personalized trade-off strategy
1	Accounting	$(NFR_{price}^{EI} \wedge NFR_{reliability}^{I}) \vee (NFR_{price}^{EI} \wedge NFR_{response\ time}^{VI})$
2	Accounting	$(NFR_{price}^{EI} \wedge NFR_{reliability}^{I}) \vee ((NFR_{price}^{EI} \wedge NFR_{throughput}^{SI}) \vee (NFR_{price}^{EI} \wedge NFR_{availability}^{SI}))$
3	Accounting	$(NFR_{price}^{VI} \wedge NFR_{response\ time}^{VI}) \wedge (NFR_{throughput}^{I} \wedge NFR_{availability}^{I})$
4	Accounting	$(NFR_{price}^{EI} \vee NFR_{response\ time}^{EI}) \wedge ((NFR_{price}^{NI} \otimes NFR_{availability}^{I}) \vee (NFR_{price}^{NI} \otimes NFR_{throughput}^{I}))$
5	Accounting	$((NFR_{price}^{SI} \otimes NFR_{response\ time}^{EI}) \vee (NFR_{price}^{SI} \otimes NFR_{reliability}^{EI})) \wedge ((NFR_{price}^{NI} \otimes NFR_{availability}^{I}) \vee (NFR_{price}^{NI} \otimes NFR_{throughput}^{I}))$

The satisfaction degree of each non-functional attribute for all the users can be computed using both the membership function descriptions in Section 2.3 and the non-functional attribute values in Table 2.4. For instance, the satisfaction degree of

www.manaraa.com

throughput non-functional attribute can be computed as 0.16. This is achieved using the *throughput* value of service 4 (8.29 MBps) according to Table 2.4 and based on the membership function of the *throughput* non-functional attribute illustrated in Figure. 2.7.

Table 2.4. The List of Services and Their Non-Functional Attribute Values with Accounting Functionality

Service	Reliability (months)	Availability (%)	Throughput (mbps)	Response time (seconds)	Price (dollars)
$Service_1$	6	90	18.13	5	41
$Service_2$	10	97	28.25	7	21
$Service_3$	8	92	25.34	2	45
$Service_4$	6	98	8.29	1	27
$Service_5$	10	96	18.65	4	30

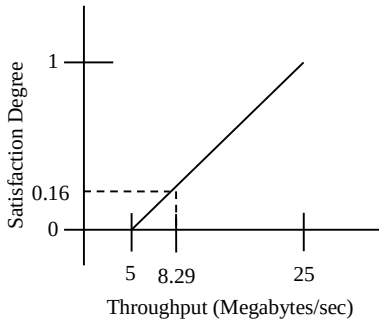

Figure 2.7. Fuzzified *throughput* value of service 4

Once the satisfaction degrees of individual non-functional attributes are obtained, an overall satisfaction of the non-functional attribute for a service can be computed. This computation is done based on how individual non-functional attributes are aggregated based on the discussion in Section 2.4. For user 1, the overall satisfaction value of service 1 can be computed as follows:

$$(NFR_{price}^{EI}(s_1) \wedge NFR_{reliability}^{I}(s_1)) \vee (NFR_{price}^{EI}(s_1) \wedge NFR_{response\ time}^{VI}(s_1))$$

$$= MAX\ \{MIN\ (0.36 \times 0.97, 1 \times 0.7), MIN\ (0.36 \times 0.97, 0.72 \times 0.87)\}$$

www.manaraa.com

= MAX {MIN (0.35, 0.7), MIN (0.35, 0.63)}

= MAX {0.35, 0.35}

= 0.35

Similarly, the values for services 2 to 5 can be computed for user 1. The services are ranked according to these overall satisfaction values. Table 2.5 shows the ranked services based on user 1's preferences.

For user 2, the overall satisfaction value of service 1 can be computed as follows:

$(NFR_{price}^{EI}(s_1) \wedge NFR_{reliability}^{I}(s_1)) \vee ((NFR_{price}^{EI}(s_1) \wedge NFR_{throughput}^{SI}(s_1)) \vee (NFR_{price}^{EI}(s_1) \wedge NFR_{availability}^{SI}(s_1)))$

= MAX {MIN (0.36 × 0.97, 1 × 0.7), MIN {MIN (0.36 × 0.97, 0.66 × 0.50), MIN (0.36 × 0.97, 0 × 0.5)}}

= MAX {MIN (0.35, 0.7), MIN {MIN (0.35, 0.33), MIN (0.35, 0)}}

= MAX {MIN (0.35, 0.7), MIN (0.33, 0)}

= MAX {0.35, 0}

= 0.35

Again, the overall satisfaction values of services 2 to 5 for user 2 can be computed and based on these values, the services ranked. Table 2.6 shows the ranked services based on user 2's preferences.

User 3's overall satisfaction value of service 1 can be computed as follows:

$(NFR_{price}^{VI}(s_1) \wedge NFR_{response\ time}^{VI}(s_1)) \wedge (NFR_{throughput}^{I}(s_1) \wedge NFR_{availability}^{I}(s_1))$

= MIN {MIN (0.36 × 0.87, 0.72 × 0.87), MIN (0.66 × 0.7, 0 × 0.7)}

= MIN {MIN (0.31, 0.63), MIN (0.46, 0)}

= MIN {0.31, 0}

= 0

For services 2 to 5, the overall satisfaction value can be computed in a similar manner and the services are ranked based on these values. The ranked services based on user 3's preferences are shown in Table 2.7.

For user 4, the overall satisfaction value of service 1 can be computed as follows:

$(NFR_{price}^{EI}(s_1) \vee NFR_{response\ time}^{EI}(s_1)) \wedge ((NFR_{price}^{NI}(s_1) \otimes NFR_{availability}^{I}(s_1)) \vee (NFR_{price}^{NI}(s_1) \otimes NFR_{throughput}^{I}(s_1)))$

= MIN {MAX (0.36 × 0.97, 0.72 × 0.97), MAX (AVERAGE (0.36 × 0.3, 0 × 0.7),

www.manaraa.com

AVERAGE (0.36 × 0.3, 0.66 × 0.7)}

= MIN {MAX (0.35, 0.7), MAX (AVERAGE (0.11, 0), AVERAGE (0.11, 0.46)}

= MIN {MAX (0.35, 0.7), MAX (0.06, 0.29)}

= MIN {0.7, 0.29}

= 0.29

Table 2.5. Ranked Services Based on User 1's Trade-off Strategy

Service (by rank)	Reliability (months)	Availability (%)	Throughput (mbps)	Response time (seconds)	Price ($/month)	Score
$Service_4$	6	98	8.29	1	27	0.87
$Service_5$	10	96	18.65	4	30	0.74
$Service_2$	10	97	28.25	7	21	0.7
$Service_1$	6	90	18.12	5	41	0.35
$Service_3$	8	92	25.33	2	45	0.19

Table 2.6. Ranked Services Based on User 2's Trade-Off Strategy

Service (by rank)	Reliability (months)	Availability (%)	Throughput (mbps)	Response time (seconds)	Price ($/month)	Score
$Service_2$	10	97	28.25	7	21	0.7
$Service_5$	10	96	18.65	4	30	0.7
$Service_1$	6	90	18.12	5	41	0.35
$Service_4$	6	98	8.29	1	27	0.33
$Service_3$	8	92	25.33	2	45	0.19

Similarly, the values of services 2 to 5 can be computed. Based on this overall satisfaction values, the services are ranked. Table 2.8 shows the ranked services based on user 4's preferences.

User 5's overall satisfaction value of service 1 can be computed as follows:

$((NFR_{price}^{SI}(s_1) \otimes NFR_{response\ time}^{EI}(s_1)) \vee (NFR_{price}^{SI}(s_1) \otimes NFR_{reliability}^{EI}(s_1))) \wedge$

www.manaraa.com

$((NFR_{price}^{NI}(s_1) \otimes NFR_{availability}^{I}(s_1)) \vee (NFR_{price}^{NI}(s_1) \otimes NFR_{throughput}^{I}(s_1)))$

= MIN {MAX (AVERAGE (0.36 × 0.5, 0.72 × 0.97), AVERAGE (0.36 × 0.5, 1 × 0.97)), MAX (AVERAGE (0.36 × 0.30, 0 × 0.7), AVERAGE (0.36 × 0.3, 0.66 × 0.7))}

= MIN {MAX (AVERAGE (0.18, 0.7), AVERAGE (0.18, 0.97)), MAX (AVERAGE (0.11, 0), AVERAGE (0.11, 0.46))}

= MIN {MAX (0.44, 0.58), MAX (0.06, 0.29)}

= MIN {0.58, 0.29}

= 0.29

In a similar manner, the overall satisfaction function values of services 2 to 5 can be computed and the services are ranked based on these values. Table 2.9 shows the services ranked according to user 5's preferences.

Table 2.7. Ranked Services Based on User 3's Trade-off Strategy

Service (by rank)	Reliability (months)	Availability (%)	Throughput (mbps)	Response time (seconds)	Price ($/month)	Score
$Service_4$	6	98	8.29	1	27	0.47
$Service_2$	10	97	28.25	7	21	0.31
$Service_5$	10	96	18.65	4	30	0.16
$Service_3$	8	92	25.33	2	45	0
$Service_1$	6	90	18.12	5	41	0

Table 2.8. Ranked Services Based on User 4's Trade-off Strategy

Service by rank)	Reliability (months)	Availability (%)	Throughput (mbps)	Response time (seconds)	Price ($/month)	Score
$Service_2$	10	97	28.25	7	21	0.5
$Service_4$	6	98	8.29	1	27	0.49
$Service_3$	8	92	25.33	2	45	0.38
$Service_5$	10	96	18.65	4	30	0.36
$Service_1$	6	90	18.12	5	41	0.29

www.manaraa.com

The results displayed in Tables 2.5 through 2.9 indicate that the recommended services for selection depend on a user's personal trade-off preferences. For instance, the top-3 recommended services for user 1 are service 4, service 5, and service 2. Those for user 3 are service 4, service 2, and service 5. The difference in the results is due to the different personal trade-off preferences of the two users (user 1 and user 3).

Table 2.9. Ranked Services Based on User 5's Trade-off Strategy

Service (by rank)	Reliability (months)	Availability (%)	Throughput (mbps)	Response time (seconds)	Price ($/month)	Score
$Service_2$	10	97	28.25	7	21	0.5
$Service_4$	6	98	8.29	1	27	0.49
$Service_3$	8	92	25.33	2	45	0.38
$Service_5$	10	96	18.65	4	30	0.36
$Service_1$	6	90	18.12	5	41	0.29

2.7. PROTOTYPE IMPLEMENTATION AND ITS EVALUATION

2.7.1. Service Selection Prototype. This section describes the prototype for the framework. This prototype was implemented using both Microsoft Visual C# on .NET framework 3.5 and Microsoft Visual Studio 2008 Professional Edition under 64-bit Windows 7 Enterprise platform on AMD FX™-8350 8 core processor. The primary components of the prototype include: a *services repository*, *a personalized trade-off strategy input parser*, and the *service ranking engine*. With the exception of the services repository (which was implemented using SQL Server 2008), these components were implemented using both C# and regular expressions.

2.7.1.1 The input handler. The usability of the input handler was the main consideration during its design. The component was developed such that users will have convenience to specify their service request with ease. The trade-off strategy was captured in a sentence-like fashion (see Figure 2.8). The notation used is *NFA_1.weight Operator NFA_2.weight* … An autocomplete feature to speed up the user-system interactions was also included with the trade-off strategy field. The data in the *Attribute*

www.manaraa.com

combo box as well as the highest and lowest satisfactions are populated from a data source which makes this implementation very adaptable to different domains. As users select and add their preferred non-functional attributes, the system analyzes the relationship between pairs of non-functional attributes and recommends the appropriate operator for aggregation (see Figure 2.8). This is also provided in the *Relationships* box.

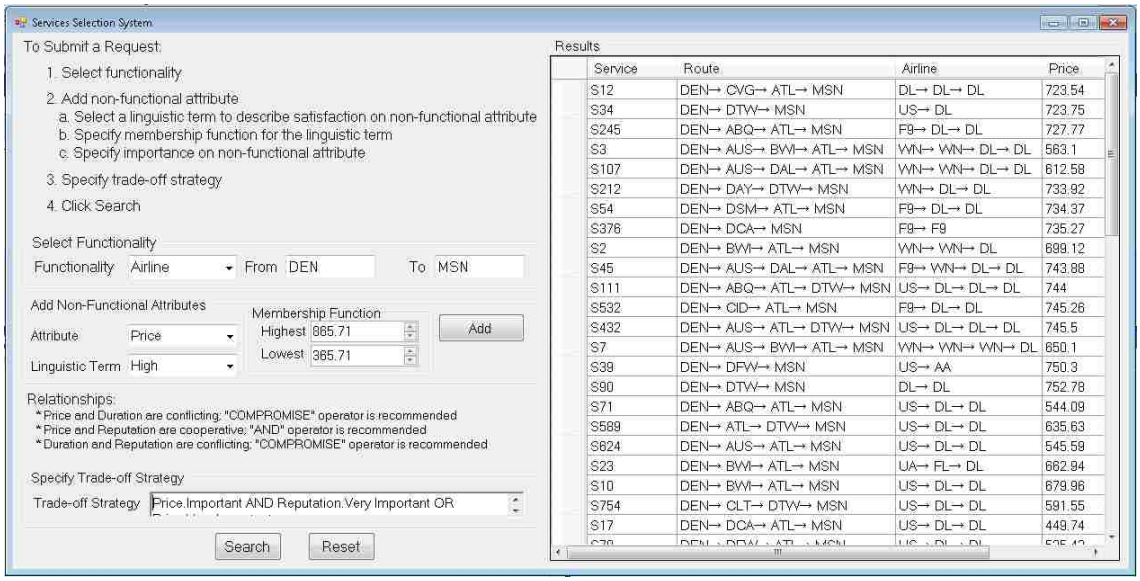

Figure 2.8. Screenshot of the personalized preference and trade-off based service selection prototype

It must be noted that the trade-off strategy does not impact the relationships that exist between any two non-functional attributes. The relationships between any two non-functional attributes reveal the interactions between those non-functional attributes. If the user's choice of aggregation operators in the trade-off strategy is consistent with recommended aggregation operators, services with the best possible satisfaction of their non-functional attributes are returned back to the user. On the contrary, if the user's choice of aggregation operators in the trade-off strategy is inconsistent with the recommended aggregation operators, services with a less satisfactory result as compared to the former results are returned.

www.manaraa.com

To demonstrate this, a comparison of the results of two separate requests applied to the services in Table 2.4 was made. Request 1 has aggregation operators which are consistent with the recommended aggregation operators as follows:

Request 1: $(NFR_{price}^{EI} \wedge NFR_{reliability}^{I}) \vee (NFR_{price}^{EI} \wedge NFR_{response\ time}^{VI})$

Request 2, which is similar to request 1 with respect to the number and types of non-functional attributes, has different aggregation operators than those recommended.

Request 2: $(NFR_{price}^{EI} \otimes NFR_{reliability}^{I}) \vee (NFR_{price}^{EI} \otimes NFR_{response\ time}^{VI})$

Tables 2.5 and 2.10 show the results for request 1 and request 2 respectively. The results show that, the top-ranked services produced by request 1 have a higher satisfaction of the non-functional attributes than the top-ranked services produced by request 2. Therefore, it is highly recommend that the recommended aggregation operators should be used by users when submitting their service request.

Table 2.10. Ranked Services Based on Request 2

Service (by rank)	Reliability (months)	Response time (seconds)	Price ($/month)	Score
$Service_1$	6	5	41	0.97
$Service_3$	8	2	45	0.92
$Service_4$	6	1	27	0.88
$Service_2$	10	7	21	0.83
$Service_5$	10	4	30	0.76

2.7.1.2 Functional matching engine. For this component, the algorithm proposed by Sajjanhar et al. [30] was adopted. The choice was mainly based on the fact that they employed the singular value decomposition in linear algebra which reveals relationship

www.manaraa.com

among services. This ensures an efficient functional service match. Also since the functional match of services is just an intermediary step, a fast algorithm was necessary. The algorithm as presented by Sajjanhar et al. [30] for the functional matching engine was therefore implemented.

2.7.1.3 Service ranking engine. The service ranking engine requires the functional matching of services, user trade-off strategy, fuzzified non-functional attribute constraints and the published non-functional attribute data. As already stated, the trade-off strategy is provided in a sentence-like fashion. In its implementation, first of all, a regular expressions to validate the user trade-off strategy was employed. Next the binary operators used (AND, OR, COMPROMISE) as well as the operands they operate on were identified. Finally, the definitions of the respective operators as described in Section 2.4.2 were then applied.

2.7.1.4 Evaluation of the implementation. The entire implementation is very simple and straight forward. First there is a search for services satisfying functional requirements, which is upper bounded by the number of services. Then satisfaction degree of each non-functional attribute for each service is also computed. This is also upper bounded by the number of user preferred non-functional attributes being considered. Finally the individual satisfaction degrees of each service are aggregated which is also upper bounded by the number of services. Therefore the entire service selection system is linear with the bottlenecks being the number of available services and the number of non-functional attributes under consideration.

2.7.1.5 Parallel implementation. For each application domain, the number of non-functional attributes is limited. For example, the accounting and airline applications discussed in this work used five (5) and four (4) non-functional attributes respectively. However, the number of available services is not limited and keeps growing. Due to this, the execution time of the service selection system is high. For instance, the service selection system takes 6.4 secs to respond when there are 100K services and 8 non-functional attributes to consider.

To overcome this bottleneck, a parallel implementation of the service selection system was performed since most of the processes as described in Section 2.7.1.4 are independent of each other. The parallel implementation is based on the divide, conquer

www.manaraa.com

and combine approach as depicted in Figure 2.9. The divide step basically divides the dataset into sub datasets based on the number of available processors.

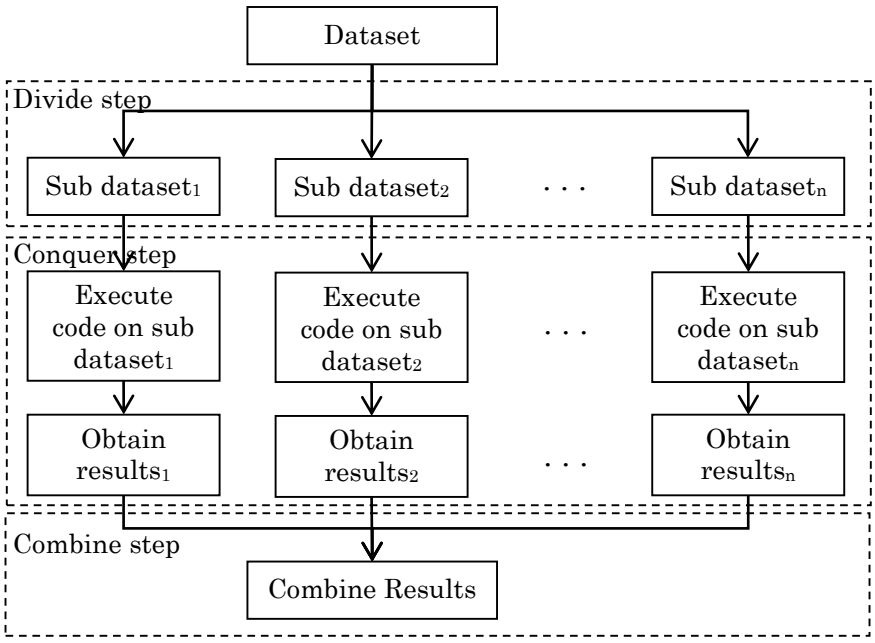

Figure 2.9. Parallel implementation of the service selection system

Each processor then executes the non-parallel version of the selection system on each sub dataset. Each processor will return result which represents the ranked services in the sub dataset. This is the conquer step. Finally, the results obtained from each processor is combined to obtain the final result which represents the ranked services for the dataset. By doing so, the execution time of the selection system was reduced based on the number of multicore processors used. The results of the parallel implementation is discussed in Section 2.7.3.

2.7.2. Application with Real Airline Services. In previous accounting example, a demonstration of how the service selection method works with infrastructural non-functional attributes was given. In this section, a case study where the proposed method is applied to domain-specific non-functional attributes of services is presented. A service's selection prototype that has been implemented to support the framework described in Section 2.2 was employed for the case study. This is to evaluate the application of the framework. The prototype was then applied to real-world airline services, the Openflights

www.manaraa.com

Dataset [34]. Five (5) different normal-user requests were supplied as input to the prototype. As a reminder, each request includes both the functionality of the service and the personalized trade-off strategy. The results obtained from each input are discussed under Section 2.7.2.2.

2.7.2.1 Dataset description. The Openflights Dataset [34], used in this implementation contains 61,199 routes between 3341 airports on 565 airlines spanning the globe. Each record in the dataset corresponds to an existing airline service as of February 2013. Each record contains the source and destination airports, airline, flight duration, flight distance and the number of stops. Since there was no price and reputation information for each airline service, two additional non-functional attributes, price and reputation, were added to the dataset. It was however challenging obtaining the reputation for all airlines. Due to that, the dataset was limited to only domestic (US) airlines. Their reputation values are readily available from the U.S. Department of Transportation.

2.7.2.2 User inputs and satisfaction functions. Using the dataset described above in Section 2.7.2.1, the prototype was tested on different user requests (i.e. different functionalities and personalized trade-off strategy) as summarized in Table 2.11. The satisfaction and dissatisfaction values of each non-functional attributes used are also tabulated in Table 2.12. Tables 2.13 to 2.17 show the results.

Table 2.11. List of Service Request from 5 Users

User	Functionality		Fuzzy logic-based personalized trade-off strategy
	From	To	
1	Denver Intl (DEN)	Madison-Dane Co (MSN)	$(NFR_{price}^{EI} \wedge NFR_{reputation}^{I}) \vee (NFR_{price}^{EI} \otimes NFR_{number\ of\ stops}^{VI})$
2	Dallas Fort Worth Intl (DFW)	St. Louis-Lambert (STL)	$(NFR_{price}^{EI} \wedge NFR_{reputation}^{I}) \vee ((NFR_{price}^{EI} \wedge NFR_{duration}^{SI}) \vee (NFR_{price}^{EI} \wedge NFR_{number\ of\ stops}^{SI}))$
3	Atlanta-Hartsfield Jackson (ATL)	Detroit Metro Wayne (DTW)	$(NFR_{price}^{VI} \wedge NFR_{number\ of\ stops}^{VI}) \wedge (NFR_{duration}^{I} \wedge NFR_{reputaiton}^{I})$
4	Austin Bergstrom Intl (AUS)	New York-John F Kennedy In. (JFK)	$(NFR_{price}^{EI} \vee NFR_{number\ of\ stops}^{EI}) \wedge ((NFR_{price}^{NI} \otimes NFR_{reputation}^{I}) \vee (NFR_{price}^{NI} \otimes NFR_{duration}^{I}))$
5	Charlotte Douglas Intl (CLT)	Los Angeles Intl (LAX)	$((NFR_{price}^{SI} \otimes NFR_{number\ of\ stops}^{EI}) \vee (NFR_{price}^{SI} \otimes NFR_{reputation}^{EI})) \vee (NFR_{price}^{NI} \otimes NFR_{duration}^{I})$

www.manaraa.com

Table 2.12. Membership Function

Non-functional attribute	Linguistic term	Satisfactory value	Dissatisfactory value
Reputation (AQR)	Good	> = 3rd	< = 14th
Duration (Hrs)	Long	< = 1.4	> = 5
Price ($)	Affordable	< = 170	> = 1520
Number of Stops	Few	< = 1	> = 4

Table 2.13. Top-5 out of 3498 Services Based on User 1's Personalized Preference and Trade-off Strategy

Service (by rank)		Duration	Price	Reputation	Number	Score
Route	Airline(s)	(hr:min)	($)	(ranking)	Of stops	
DEN→MSN	UA	2:08	165.71	10th	0	0.87
DEN → MSN	F9	2:08	322.75	4th	0	0.86
DEN→ MSN	US	2:08	476.46	9th	0	0.75
DEN→CAK→ATL→MSN	F9→FL→DL	6:23	634.37	3rd	2	0.63
DEN → ATL → MSN	FL → DL	4:47	666.2	3rd	1	0.61

Table 2.14. Top-5 out of 7468 Services Based on User 2's Personalized Preference and Trade-off Strategy

Service (by rank)		Duration	Price	Reputation	Number	Score
Route	Airline(s)	(hr:min)	($)	(ranking)	Of stops	
DFW → ATL → STL	DL → FL	3:24	428.84	3rd	1	0.64
DFW→ATL→DEN→ STL	DL→FL → F9	6:52	641.45	3rd	2	0.63
DFW →CVG → ATL→STL	DL→DL→ FL	4:48	638.33	3rd	2	0.62
DFW → DEN → STL	F9 → F9	3:48	628.72	4th	1	0.60
DFW→DEN→ ATL → STL	US →F9→ FL	6:06	663.17	5th	2	0.59

www.manaraa.com

Table 2.15. Top-5 out of 8409 Services Based on User 3's Personalized Preference and Trade-off Strategy

Service(by rank)		Duration (hr:min)	Price ($)	Reputation (ranking)	Number of stops	Score
Route	Airline(s)					
ATL → DTW	FL	1:41	330.17	1st	0	0.64
ATL → DTW	DL	1:41	172.94	6th	0	0.59
ATL → DAY → DTW	FL → DL	2:10	368.15	3rd	1	0.55
ATL → AVL → DTW	DL → DL	2:15	435.4	6th	1	0.53
ATL → CLE → DTW	DL → DL	2:17	636.22	6th	1	0.53

Table 2.16. Top-5 out of 7927 Services Based on User 4's Personalized Preference and Trade-off Strategy

Service(by rank)		Duration (hr:min)	Price ($)	Reputation (ranking)	Number of stops	Score
Route	Airline(s)					
AUS → DEN → JFK	F9 → DL	5:47	340.01	5th	1	0.43
AUS → ATL →DAY →JFK	DL→FL→DL	5:04	515.23	4th	2	0.42
AUS → ATL →DEN →JFK	DL→FL→ DL	8:44	545.6	4th	2	0.42
AUS → JFK	DL	3:32	406.54	6th	0	0.42
AUS → BWI → JFK	WN → DL	4:01	332.04	7th	1	0.42

www.manaraa.com

Table 2.17. Top-5 out of 15441 Services Based on User 5's Personalized Preference and Trade-off Strategy

Service (by rank)		Duration (hr:min)	Price ($)	Reputation (ranking)	Number of stops	Score
Route	Airline(s)					
CLT → LAX	UA	4:44	493.48	11th	0	0.63
CLT → BWI → BOS →LAX	FL→FL→AS	8:09	734.93	2nd	2	0.61
CLT → CVG → LAX	DL → DL	5:27	468.36	6th	1	0.60
CLT → BWI → ATL →LAX	FL→DL→FL	7:14	753.39	2nd	2	0.60
CLT → BNA → LAX	US → AS	5:14	273.08	7th	1	0.60

2.7.2.3 Evaluation. The service selection system proposed in this work was evaluated. The focus of this evaluation is to compare the results from both requests with preferences and requests without preferences. In the evaluation, a service search with an input similar to user 1's input in Table 2.11 except that there is no personalization of the non-functional attributes was performed. (Input: airline service from Denver Intl (DEN) to Madison-Dane Co (MSN) with a trade-off strategy ($NFR_{price}^{SI} \wedge NFRreputationSI) \wedge (NFR_{price}^{SI} \wedge NFR_{number\ of\ stops}^{SI})$). The result is shown in Table 2.18.

From Table 2.18, it can be seen that the overall satisfaction of all the top-5 services is very low; 0.29 for the first service and 0.17 for the next four services. It shows that even the best services available in the repository have a low overall satisfaction with respect to the trade-off strategy. The low satisfactions arise from the aggregation operators used. For instance, based on the published data of the airline services, *price* and *number of stops* non-functional attributes have a conflicting relationship and therefore it will be appropriate to use the COMPROMISE operator instead of AND.

Comparing Table 2.18 to the results in Table 1.13 (personalized preference results), none of the top-5 services in Table 2.13 appeared in Table 2.18. Actually, the top-5 services in Table 2.13, appeared as services 96, 632, 291, 647 and 656 respectively in the complete results of (Input: *airline service from Denver Intl (DEN) to Madison-*

www.manaraa.com

Dane Co (MSN) with a trade-off strategy ($NFR_{price}^{SI} \wedge NFR_{reputation}^{SI}$) $\wedge$ ($NFR_{price}^{SI} \wedge NFR_{number\ of\ stops}^{SI}$)).

Table 2.18. Top-5 out of 3498 Services Based on the above Trade-off Strategy

Service (by rank)		Duration (hr:min)	Price ($)	Reputation (ranking)	Number Of stops	Score
Route	Airline(s)					
DEN → DFW → MSN	NK → AA	3:54	1127.19	14th	1	0.29
DEN→BOS→DTW →MSN	B6 → DL → DL	6:51	1480.94	14th	2	0.17
DEN →BOS→EWR→MSN	B6 → B6 → UA	6:58	1744.42	14th	2	0.17
DEN →BOS→EWR→MSN	UA → B6 → UA	6:58	1247.79	14th	2	0.17
DEN →BOS→DCA→ MSN	US → B6 → F9	7:10	1106.75	14th	2	0.17

2.7.3. Experimental Evaluation. The experimental evaluation was performed to see the scalability of the prototype with respect to the number of non-functional attributes and the number of available services. The prototype was executed with a varied number of non-functional attributes (2, 4, 6, 8, and 10) against different number of services (20K, 40K, 60K, 80K, and 100K) as seen in Figure 2.10. Each number of non-functional attributes is run against all of the different number of services. For example, 2 non-functional attributes, is run against 20K, 40K, 60K, 80K, and 100K number of services and the execution time at each run is recorded. Our expectations were that the system scales linearly with increasing number of services and non-functional attributes. Figure 2.10 shows the execution time vs. the number of non-functional attributes with respect to the different number of services.

www.manaraa.com

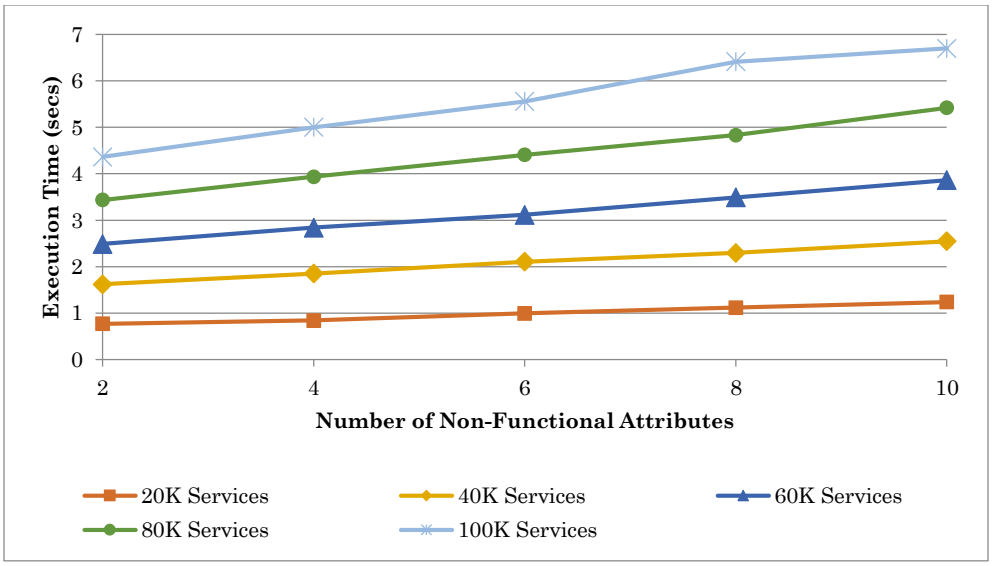

Figure 2.10. Execution time of the services selection system

Figure 2.10 shows that the implementation is expensive with respect to the time complexity. Therefore, a re-implemented the service selection system in a parallel fashion was performed. Once the parallel implementation was complete, a similar scalability experiment was conducted. Figure 2.11 shows the execution time of the parallel implementation. The number of cores used for the experiment was eight (8).

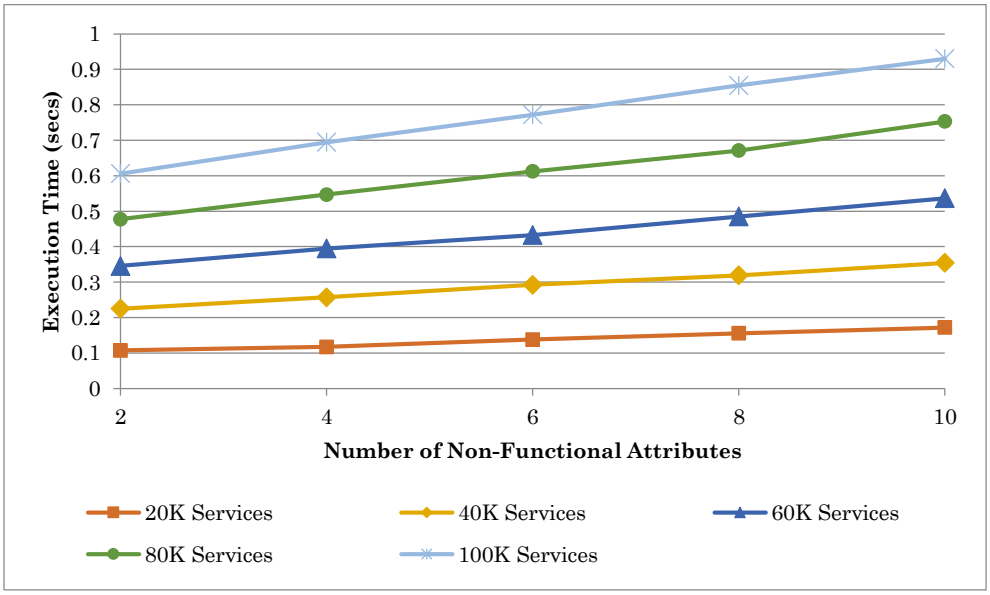

Figure 2.11. Execution time of the services selection system parallel implementation

www.manaraa.com

2.7.4. Impact of Membership Functions and Weights on the Evaluation. An analysis is carried out to investigate the robustness of the services selection framework discussed in this work. This is performed to determine the impact on the actual results if the membership function or the weight is tweaked. In this section, four (4) scenarios are presented. Two (2) of such scenarios determine how changing (i.e. strengthening or relaxing) the membership function of individual non-functional attribute will impact the recommended services. The other two (2) scenarios focus on the impact of changing (increasing or decreasing) the weight of individual non-functional attributes will have on recommended services.

2.7.4.1 Strengthening/relaxing membership function of individual non-functional attributes. Membership functions can be strengthened or relaxed. Strengthening a non-functional attribute's membership function means increasing its lowest level of service satisfaction. For instance, the reputation non-functional attribute in Figure 2.4 can be strengthened by increasing the lowest service satisfaction level from 14th to 12th (see Figure 2.12).

Similarly, relaxing a non-functional attribute's membership function implies reducing its lowest level of service satisfaction. An example will be to reduce the lowest service satisfaction level of the response time non-functional attribute in Figure 2.3 from 10 secs to 12 secs as depicted in Figure 2.13.

The membership function in Table 2.12 was strengthened (see Table 2.19). Based on Table 2.19, user 2's service request is executed by the prototype. The recommended services are shown in Table 2.20.

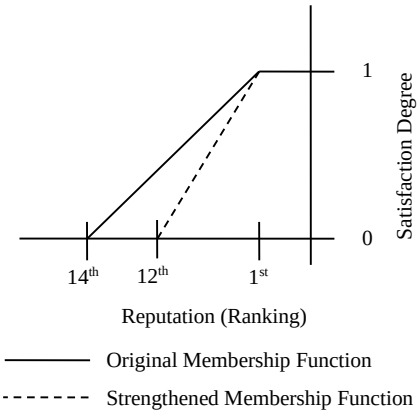

Figure 2.12. Strengthening the *Reputation* non-functional attribute

www.manaraa.com

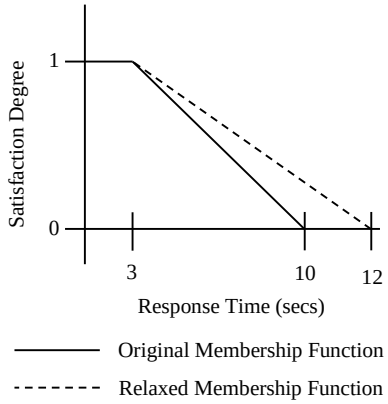

Figure 2.13. Relaxing the *Response Time* non-functional attribute.

Table 2.19. Strengthened Membership Functions

Non-functional attribute	Linguistic term	Satisfactory value	Dissatisfactory value
Reputation (AQR)	Good	> = 3rd	< = 9th
Duration (Hrs)	Long	< = 1.4	> = 4
Price ($)	Affordable	< = 170	> = 950
Number of Stops	Few	< = 1	> = 3

Table 2.20. Top-5 out of 7468 Services Based on User 2's Personalized Preference and Trade-off Strategy with Strengthened Non-Functional Attribute

Service (by rank)		Duration	Price	Reputation	Number	Score
Route	Airline(s)	(hr:min)	($)	(ranking)	Of stops	
DFW → ATL → STL	DL → FL	3:24	428.84	3rd	1	0.61
DFW → ATL → STL	DL → DL	3:24	315.83	6th	1	0.53
DFW → CVG → ATL → STL	DL → DL→ DL	4:48	525.32	6th	2	0.52
DFW → ELP → ATL → STL	AA → DL →FL	6:05	526.79	6th	2	0.51
DFW → CVG → DCA →STL	DL→ DL→ WN	5:21	480.56	6th	2	0.50

www.manaraa.com

2.7.4.2 Increasing/decreasing weights of individual non-functional attributes. In order to show the effect of weights users place on individual non-functional attributes of recommended services, the weight values are tweaked. The weights user 2 placed on his/her individual non-functional attributes were decreased to the following: the weights for *price*, *reputation*, *duration* and *number of stops* non-functional attributes were decreased to extremely low, fair, low and low respectively. The recommended services generated by the prototype are tabulated in Table 2.21.

Table 2.21. Top-5 out of 7468 Services Based on User 2's Personalized Preference and Trade-off Strategy with Decreased Weights

Service (by rank)		Duration	Price	Reputation	Number	Score
Route	Airline(s)	(hr:min)	($)	(ranking)	Of stops	
DFW → ATL → STL	DL → DL	3:24	315.83	6th	1	0.25
DFW → ATL → STL	DL → FL	3:24	428.84	3rd	1	0.24
DFW → DEN → STL	US → F9	3:48	412.49	7th	1	0.23
DFW → ATL → STL	AA → FL	3:24	477.43	6th	1	0.23
DFW → CVG → DCA →STL	DL→DL→ WN	5:21	480.56	6th	2	0.23

From the results in Tables 2.20 and 2.21, it can be concluded that, tweaking the membership functions and the weights influences the selected services. For instance, the top service for user 2 after strengthening the non-functional attributes are flights operated by Delta (DL) and Airtran (FL) that routes from Dallas Fort Worth Intl Airport (DFW) through Atlanta-Hartsfield Jackson Airport (ATL), and finally to St Louis Lambert Airport (STL).

2.8. CONCLUSION

In service markets, *reliability* and other non-functional attributes generally play crucial roles in service selection. Due to the cooperative, conflicting, or exclusive relationships among non-functional attributes, users are likely to specify trade-offs when requesting services. This section presents a novel service selection method that allows

www.manaraa.com

users to represent their elastic non-functional attributes using linguistic terms. At the same time, they are able to explicitly specify their personalized trade-offs among non-functional attributes for service selection. Also, the method permits users to specify the weights, in linguistic terms, of each non-functional attribute. First the satisfaction degree of individual non-functional requirements is computed for each service satisfying user's functionality. Then the overall satisfaction degree for that service, based on a user's personalized trade-off strategy, is computed using fuzzy connective operators. Services are then ranked using the overall satisfaction degrees and top-ranked services are selected for the user accordingly. To illustrate how the proposed method works, an illustrative example was given. Results from the case study presented show the effectiveness of the method and that the system can select services to meet users' individual service needs. In addition, an evaluation of the proposed method was performed and was concluded that the service selection method scales well with the number of non-functional attributes and the number of available services. Compared with existing service selection methods, the proposed method in this paper is more efficient in incorporating users' personal preferences and trade-offs.

www.manaraa.com

3. AGGREGATING RANKED SERVICES FOR SELECTION (ARSS)

3.1. MOTIVATION

As a motivating example, consider a user whose search intent is to find a *cheap* nonstop flight from New York to London that gets to London in the *shortest possible time*. Based on this user's search intent, he/she concerned with three non-functional attributes, *price*, *stops*, and *duration* and is not willing to trade-off any of these non-functional attributes. The user's request functionality and trade-off strategy are as follows:

Functionality: *Flight from New York to London*
Trade-off Strategy: *Price AND Stops AND Duration* [1]

A "cheap" flight for this user is any flight under $1,150 and the phrase "shortest possible time" to describe the user's preference on duration non-functional attribute refers to any flight that gets to London in under 7 hours. Preferences for each non-functional attribute for his/her initial request is shown in Table 3.1.

The user submits his/her initial request that results in ranked list of flights shown in Table 3.2. From the table, although the user's requirement for nonstop criteria is satisfied, his/her requirement for price and duration were not met. For this reason, the user relaxes his/her nonstop and duration criteria and then performs the search again using modified preferences on the non-functional attributes from the initial request (see Table 3.1). Again, the user's stop criteria was met but neither the requirement for price nor duration was met (see Table 3.3). However, the flights from Table 3.3 have non-functional attribute values that are close to the user's preferences for his/her modified request.

At this point, none of the flights have fully satisfied the user's preferences. Therefore, he/she decides to modify the preferences on the non-functional attributes and try the search one more time (see Table 3.1 for the modified request 2). The ranked list of flights generated from this request, as shown in Table 3.4, satisfies the price and stops non-functional attributes but not duration.

www.manaraa.com

Table 3.1. Three Similar Requests Showing the Differences in Preferences on Non-Functional Attributes

Request	Price ($)	Stops	Duration
Initial Request	Under 1,150	Nonstop	Under 7h
Modified Request1	Under 1,150	1 Stops	Under 12h
Modified Request2	Under 1,150	2 Stops	Under 12h

Note: The 3 requests in this table are similar because they all have the same functionality and trade-off strategy.

Table 3.2. Suggested Ranked List of Flights that Closely Match User's Initial Request

Flight	Price ($)	Stops	Duration
American	1, 731	Nonstop	6h 50m
British Airways	1, 791	Nonstop	6h 55m
Virgin Atlantic	1, 851	Nonstop	6h 40m
Aeroflot	2, 403	Nonstop	6h 50m

Table 3.3. Suggested Ranked List of Flights that Closely Match User's First Modified Request

Flight	Price ($)	Stops	Duration
Virgin Atlantic	1, 369	1 stop	12h 50m
British Airways	1, 469	1 stop	13h 30m
Aeroflot	1, 651	1 stop	15h 15m

After the third search, assume that the user realizes that his/her search intent cannot be completely satisfied. However, he/she needs to choose a flight based on the three ranked lists of flights obtained so far. It becomes necessary to find the optimal flight with respect to the user's requests. This can be achieved by aggregating the three ranked lists of flights (Tables 3.2 to 3.4), to produce an aggregated ranked list (Table 3.5). The aggregated ranked list is a compromise between Tables 3.2 to 3.4 and closely represents the user's search intent.

3.2. BACKGROUND AND RELATED WORK

Rank aggregation is the problem of combining several ranked lists of objects in a robust way to produce a single consensus ranking of the objects [35]. The rank

www.manaraa.com

aggregation problem is not purely a new research area, as it derives from many previous works from many information retrieval subfields [36]. It involves finding a consensus ranking on a set of candidates, based on the preferences of individuals [36, 37, 38, 39, 40]). In computer science, rank aggregation has proven to be a useful and powerful paradigm in several applications such as meta-searching and information retrieval, search engine spam fighting, e-commerce, learning from experts, analysis of population preference sampling, committee decision making and more [37]. However, it has not been employed much in the area of service computing for service selection.

Table 3.4. Suggested Ranked List of Flights that Closely Match User's Second Modified Request

Flight	Price ($)	Stops	Duration
Virgin Atlantic	967	2 Stops	16h 50m
Delta	1, 122	2 Stops	17h 30m
Virgin Atlantic	1, 127	2 Stops	21h 15m
British Airways	1, 143	2 Stops	20h 50m

Table 3.5. Aggregated Ranked List of Flights From Tables 3.2, 3.3, and 3.4

Flight	Price ($)	Stops	Duration
Virgin Atlantic	1, 369	1 stop	12h 50m
Virgin Atlantic	967	2 Stops	16h 50m
British Airways	1, 469	1 stop	13h 30m
American	1, 731	Nonstop	6h 50m
British Airways	1, 791	Nonstop	6h 55m
Virgin Atlantic	1, 851	Nonstop	6h 40m
Virgin Atlantic	1, 127	2 Stops	21h 15m
Aeroflot	2, 403	Nonstop	6h 50m
Delta	1, 122	2 Stops	17h 30m
Aeroflot	1, 651	1 stop	15h 15m
British Airways	1, 143	2 Stops	20h 50m

www.manaraa.com

In voting, the rank aggregation given a list of n candidates $\{c_1, c_2, \ldots, c_n\}$ running for an election and a set of m voters, each voter issues an ordered list, $\prec$, of full or subset of the n candidates. An ordered list from voter j can be seen as a permutation, $\prec_j$, where $\prec_j (i)$indicates the position of candidate c_i in the ordered list of voter j. A candidate c_i is preferred by voter j if $i = 1$. From these m ordered lists, rank aggregation is employed to form one list to select the best candidate that best suits all voters [41]. The Condorcet winner of an election is the candidate who, when compared to every other candidate, is preferred by more voters i.e. a candidate who would beat any opponent in a simple majority in a two-candidate election. The Condorcet voting paradox, however, indicates that such a winner may not always exist [42]. Borda count [43] is one selection algorithm that searches for the best trade-off for this criterion. The Borda count of a candidate c_i is its mean position over all ordered lists. Candidates are subsequently sorted in increasing order of Borda count.

$$BordaCount(c_i) = \frac{1}{m}\sum_{j=1}^{m} \prec_j (i)$$

Another selection algorithm, Reciprocal Rank, finds the geometric mean of a candidate c_i's positions within all ordered lists. Candidates are then sorted in increasing order of reciprocal rank.

$$Reciprocal\ Rank(c_i) = \frac{1}{\sum_{j=1}^{m} \frac{1}{\prec_j (i)}}$$

The Reciprocal Rank Fusion [44], is a simple method for rank aggregation typical in the information retrieval (IR) domain. It simply sorts documents form multiple IR systems according to a naïve scoring formula [44]. Given a set D of documents to be ranked and a set of rankings R, each a permutation on 1..|D|, the reciprocal rank fusion score can be computed as

www.manaraa.com

$$RRFScore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}$$

Cormack et al. [44] found that reciprocal rank fusion, when used to combine the results of IR methods, almost invariably improved on the best of the combined results. They also showed that the reciprocal rank fusion equaled or outperformed some established meta ranking standards.

In their survey, Kopliku et al. [36], proposed a simple analysis framework for rank aggregation as employed in web search. They focused on more recent trends, namely cross vertical aggregated search and relational aggregated search, which are already present in current Web search and an overview of existing work. Hofreiter and Marchand-Maillet [41] modeled the web service selection problem using rank aggregation strategies (full voting strategy). In their model, each web service is seen as a candidate in the selection process and a QoS factor is an abstract voter that will sort the web services according to their values. The web services were then ranked according to their QoS factors independently, leading to a number of ordered lists. Finally, the ordered lists were aggregated into a final ordered list, from which users can select the best web service.

Baltrunas et al. [45] proposed an idea that applied rank aggregation and collaborative filtering to group recommendation. Their premise was that there is sometimes the need to recommend services to satisfy all members of a group. Their method took into consideration the individual preferences of the group's members in order to generate effective group recommendations. The result of their group recommendation process is an ordered list of items. They employed existing rank aggregation methods, taking a set of predicted ranked lists, one for each group member, and producing one combined and ordered recommendations' list.

Qin et al. [46], proposed a distance-based rank aggregation model called the coset-permutation distance based stagewise (CPS) model. The model is stagewise based on probabilistic model on permutations. The model first of all decomposes the generative process of a permutation π into sequential stages and then at the K^{th} stage, an object is selected and assigned to position k with a certain probability [46]. The CPS model then defines the selection probability based on the distance between a location permutation σ and the right coset of π (referred to as coset-permutation distance) at each stage.

www.manaraa.com

Most rank aggregation problems often assume that users' preferences are given by total orders. However, the rankings encountered in many natural situations, as in our case, often come with incomplete orderings of a set of candidates. To aggregate several incomplete rankings into one consensus ranking has additional challenges, since all the distance measures known so far are based on complete orderings of the candidates. Fagin et al. [35] provided a comprehensive view of comparing partial rankings, and proposed several metrics to compare partial rankings. They considered variations of the Kendall τ distance where they varied a certain parameter. The first set of metrics was based on profile vectors and the second set of metrics were based on the Hausdorff distance.

Brandenburg et al. [40] considered the generalization of total and bucket orders to partial orders and compare them by the nearest neighbor and the Hausdorff Kendall's τ distances. Pihur et al. [47] presented two distinct algorithms for rank aggregation: the Cross-Entropy Monte Carlo algorithm and the Genetic algorithm, and discussed rank aggregation as an optimization problem.

Although all the works discussed in this section are able to aggregate several ranked lists, the aggregated ranked list they produce do not best fit with respect to the set of their input ranked lists. In other words, the aggregated ranked list produced by our algorithms closely represent the sets of ranked lists than existing methods. This is shown later in section 3.6. Also, the proposed algorithm in this work to deal with incomplete rank lists, RAIL, recursively extends the partial orderings to complete orderings rather than just assigning arbitrary ranks to missing elements in the input rank lists.

3.3. OVERVIEW OF THE METHOD

This section gives an overview of the method for aggregating ranked services for selection (rank aggregation) in general and also describes the components that make up its framework proposed in this work.

3.3.1. Framework Description. Figure 3.1 shows the framework of the service aggregation method. The main components of the framework are the *service ranking engine* [1, 2] and the *service aggregation engine*. The *service ranking engine* searches for services and rank the results based on user's personal preference(s). It has been discussed in detail in section 2. The ranked lists of services produced by the *service ranking engine*

www.manaraa.com

are subsequently aggregated by the *service aggregation engine*. Aggregating ranked services is considered as an optimization problem in this work. The problem is to find a new ranked list (solution), where the total distance from the solution to the other set of ranked lists of services is minimum. The aggregated ranked list, which reflects the overall rankings together, are subsequently selected for the user.

The aggregation engine consists of four different components as shown in Figure 3.1. First there is the *make complete list* component. This component changes all incomplete ranked lists of services to the aggregation engine to complete ranked lists. This is necessary because as stated in section 1, there are instances where the set of ranked lists of services to be aggregated come with incomplete orderings. As such it becomes challenging to determine the overall rank of those services. For instance, considering Tables 3.2-3.4 in section 1, it is hard to correctly determine the overall rank of American airlines since it only ranks 1st in one list and does not appear in the other two ranked lists, i.e. its rank in the other ranked lists is unknown. In this example, the *make complete list* component will be used to determine the rank of American airlines in the other two ranked lists.

The next component in the services aggregation engine, *define optimization problem*, defines the optimization problem, given the set of complete ranked lists of services from the *make complete list* component. The *compute minimum cost* component, thereafter solves the optimization problem defined by the *define problem* component. The results from the *compute minimum cost* component is then decoded by the *decode results* component to obtain the aggregated ranked list. In section 3.4, a detailed discussion of how all of these components work together to select the top-k aggregated ranked services for the user is given.

3.4. SERVICE AGGREGATION ENGINE

Given *m* lists of top-k ranked services from the *service ranking engine*, it is necessary to determine a consensus of the top-k ranked lists that reflects all rankings together. Rank aggregation is considered as an optimization problem, which is formally defined as follows.

www.manaraa.com

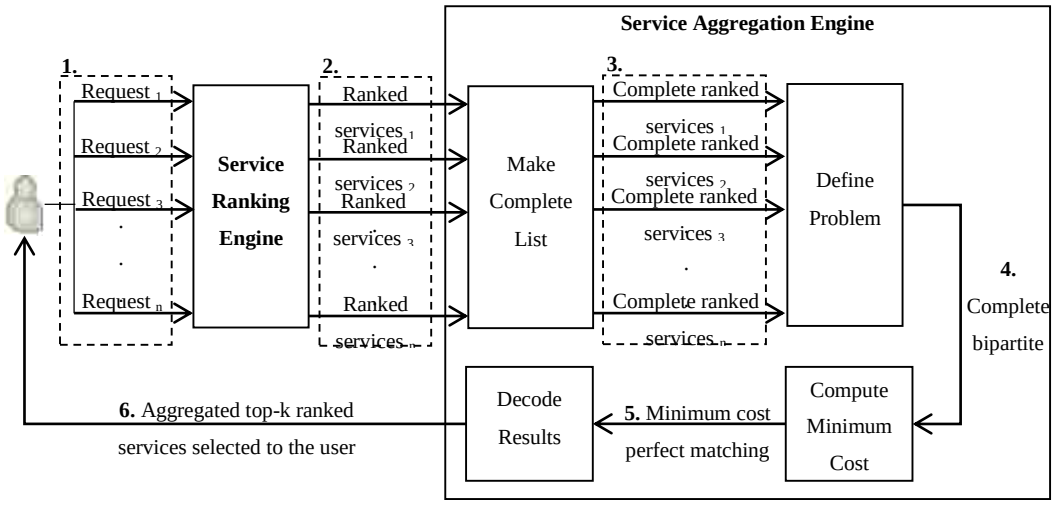

Figure 3.1. Framework of the proposed services aggregation method

Definition (Rank Aggregation Optimization Problem). Given m ordered lists, $\prec_1, \prec_2, \dots, \prec_m$, a distance measure D, find a new ordered list ρ^* such that the total distance from ρ^* to all the input lists is the minimum, i.e.,

$$\rho^* = \arg\min_{\rho} \left(\sum_{i=1}^{m} D(\rho, \prec_i) \right) \tag{12}$$

There are many choices for distance measure D. In this paper the Spearman's Footrule distance [47] and Kendall's τ distance [47], are considered which will be discussed in detail next.

3.4.1. Distance Measures. Both the Spearman's Footrule distance and the Kendall's τ distance are used to measure the difference or disagreement, between two input lists. Each ordered list is a full permutation of a set.

The Spearman's Footrule considers the position difference of an element in two orderings, and the summation of the absolute values of the differences is called Spearman's Footrule distance.

www.manaraa.com

Definition (Spearman's Footrule Distance). Let $\prec_1$, $\prec_2$ be two complete orderings. Let $\prec_1^a$ be the position of element *a* in ordering $\prec_1$, and $\prec_2^a$ be the position of element *a* in ordering $\prec_2$, then the Spearman's Footrule distance is defined as:

$$F(\prec_1, \prec_2) = \sum_a |\prec_1^a - \prec_2^a| \tag{13}$$

Unlike the Spearman's Footrule distance, the Kendall's τ distance uses a different approach in measuring the "closeness" between two ordered lists. It counts the number of pairwise inversions/disagreements between the two input lists.

Definition (Kendall's τ Distance). Let $\prec_1$, $\prec_2$ be two complete orderings. Let $\prec_1^a$ be the position of element *a* in ordering $\prec_1$ and $\prec_1^b$ be the position of element *b* in ordering $\prec_1$. Then the pairwise inversion $\psi_{a,b}(\prec_1, \prec_2)$ is defined as:

$$\psi_{a,b}(\prec_1, \prec_2) = \begin{cases} 1, & if \ \prec_1^a > \prec_1^b \ and \ \prec_2^b > \prec_2^a \\ 0, & otherwise \end{cases} \tag{14}$$

The Kendall's τ distance is subsequently defined as the summation of all pairwise inversions:

$$\tau(\prec_1, \prec_2) = \sum_{a,b} \psi_{a,b}(\prec_1, \prec_2) \tag{15}$$

The Kemeny measure [48], shows how fit the new ordered list, ρ^*, is with respect to the set of *m* ordered lists. A Kemeny meaure of 0 indicates a good fit, 1 indicates a revert fit and 0.5 is the random level fit [48].

Definition (Kemeny measure). Let ρ^* be a new ordered list with respect to a set of ordered lists, $\eta = \{\prec_1, \prec_2, \dots, \prec_m\}$, then the Kemeny measure *K*, is given as:

$$K(\rho^*, \eta) = \frac{1}{m} \sum_{j=1}^{m} \tau(\rho^*, \eta_j) \tag{16}$$

www.manaraa.com

For example, given two complete ranked lists $\prec_1 = \{s_1, s_3, s_5, s_4, s_2\}$ and $\prec_2 = \{s_3, s_1, s_2, s_4, s_5\}$. The Spearman's Footrule distance can be computed as follows:

$$F(\prec_1, \prec_2) = \left|\prec_1^{s_1} - \prec_2^{s_1}\right| + \left|\prec_1^{s_2} - \prec_2^{s_2}\right| + \left|\prec_1^{s_3} - \prec_2^{s_3}\right| + \left|\prec_1^{s_4} - \prec_2^{s_4}\right| + \left|\prec_1^{s_5} - \prec_2^{s_5}\right|$$
$$= |1-2| + |5-3| + |2-1| + |4-4| + |3-5|$$
$$= 6$$

Whereas, the Kendall's τ distance can be computed as:

$$\tau(\prec_1, \prec_2) = \psi_{s_1,s_2} + \psi_{s_1,s_3} + \psi_{s_1,s_4} + \psi_{s_1,s_5} + \psi_{s_2,s_3} + \psi_{s_2,s_4} + \psi_{s_2,s_5} + \psi_{s_3,s_4} + \psi_{s_3,s_5} + \psi_{s_4,s_5}.$$
$$= 0 + 1 + 0 + 0 + 0 + 1 + 1 + 0 + 0 + 1 = 4$$

Although given the two ordered lists, both metrics can be computed in polynomial time, to compute the rank aggregation based on the two metrics impose different degrees of challenges: while to solve the rank aggregation problem defined in equation (12) using the Spearman's Footrule distance is solvable in polynomial time, the same problem becomes NP-hard to solve when Kendall's τ distance is used [35]. For this reason, Spearman's Footrule distance was adopted to compute rank aggregation in the proposed algorithm. For cross-validation, Kemeny measure was used to show how fit the solution is with respect to the input lists.

3.4.2. Rank Aggregation for Complete Lists (RACoL) Algorithm. First, a demonstration of how the Spearman's Footrule distance can be used as the distance measure to compute rank aggregation when the input rank lists are all complete lists, i.e., each input list is a full permutation of a set of *n* items is given. Second, how to deal with incomplete input lists will be discussed later in section 3.4.3.

When using the Spearman's Footrule distance, the rank aggregation problem defined in equation (12) becomes: to compute a permutation of *n* items ρ^* such that the Footrule distance from ρ^* to all input lists is minimized,

$$\rho^* = \arg\min_{\rho} \left(\sum_{i=1}^{m} F(\rho, \prec_i) \right) \tag{17}$$

www.manaraa.com

To solve problem (17), the Rank Aggregation for Complete Lists (RACoL) algorithm (see algorithm 1) is proposed. The idea behind the algorithm is to consider a ranked list as a matching between n elements, $e_1, e_2, \ldots, e_n$, and n positions, $1,2,\ldots,n$. The algorithm takes as input the m complete ranked lists, C. The output is the aggregated ranked list, called *SuperList*. The three (3) main steps in the algorithm are discussed.

ALGORITHM 1. RANK AGGREGATION FOR COMPLETE LISTS (RACOL)

Input: m complete list (C).

Output: The aggregated ranked list (SL)

form a complete bipartite graph, $G = (V, E)$, *where* $V = L \cup R$;

compute the edge cost $c(i, j)$ *as weight* $\forall (i, j) \in E$;

$SL \leftarrow$ *Min_Cost_Perfect_Matching (G, n);*

Step 1. Construct a complete, bipartite graph $G = (V, E)$. The left vertex set L represent the n elements, and the right vertex set R represent the n positions, as shown in Figure 3.2. Since it is a complete graph, the edge set E includes edges going from each vertex in L to each vertex in R, i.e., $E = \{(u, v), \forall\, u \in L, \forall\, v \in R\}$.

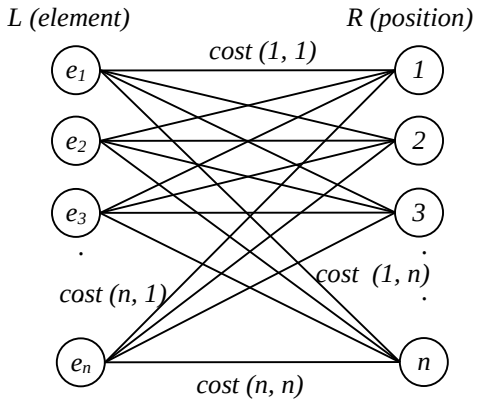

Figure 3.2. A complete bipartite graph with edge cost.

Step 2. Add cost for each edge in the complete bipartite graph. The cost of edge *(i, j)* is defined as the total penalty for placing an item i in position j, given by

www.manaraa.com

$$cost(i,j) = \sum_{l=1}^{m} d(j, \prec_l^i) \tag{18}$$

where $\prec_l^i$ is the position of item *i* in ordering *l* , and $d(j, \prec_l^i)$ is the distance between j and $\prec_l^i$, $d(j, \prec_l^i) = |j - \prec_l^i|$. So *d(j, $\prec_l^i$)*, intuitively, is the cost incurred in list *l* for positioning item *i* at position *j*; summation from all input lists is the total cost for having item *i* at position *j*.

Step 3. Finally, a minimum-cost perfect-matching [49] problem on *G* is solved. A perfect matching *M* in a bipartite graph *G*, is a subset of edges such that each node in *G* is met by exactly one edge in the subset. On a weighted, complete bipartite graph, the minimum-cost perfect-matching problem is to find an optimal matching, i.e., a perfect matching *M* which minimizes the total cost $\sum_{e \in M} w(e)$.

To compute the minimum-cost perfect-matching, first create antiparallel edges of the original edge set *E*, which is to add an edge *(j,i)* for each edge *(i,j)* ∈ *E*, $i \in L$ and $j \in R$, and then extend the cost function to antiparallel edges:

$\forall (i,j) \in E, i \in L, j \in R$, define

$$\begin{cases} w(i,j) = cost(i,j), \text{and} \\ w(j,i) = -cost(i,j) \end{cases} \tag{19}$$

The minimum-cost perfect-matching algorithm to solve the minimum-cost perfect-matching problem [49] is presented in algorithm 2.

In solving the minimum-cost perfect-matching problem, construct a flow network, *G'* as follows (See Figure 3.3):

(1) Add source *s* and sink *t*.

(2) Add edges from *s* to each vertex in *L*, and from each vertex in *R* to *t*.

(3) The new edges all have weight *0*. This will ensure that the additional edges do not contribute to the total weight on any path from *s* to *t*.

(4) Assign capacity 1 to all edges.

www.manaraa.com

ALGORITHM 2. MINIMUM-COST PERFECT-MATCHING

Input: Complete undirected bipartite graph (*G*).

Output: Matching (*M*).

initialize $M = \emptyset$;

build a flow network $G' = (V', E')$;

initialize flow $f(u, v) = 0 \ \forall (u, v) \in E'$;

initialize residual network, $G'_f \leftarrow G'$;

repeat

$P \leftarrow$ *compute shortest path from s to t on* G';

$f \leftarrow f + f_p$;

Compute the residual network G'_f;

until $|\boldsymbol{f}| = \boldsymbol{n}$;
$\boldsymbol{M} = \{(\boldsymbol{u}, \boldsymbol{v}) : \boldsymbol{u} \in \boldsymbol{L}, \boldsymbol{v} \in \boldsymbol{R}, \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{v}) > \boldsymbol{0}\}$;
return $\boldsymbol{M}$;

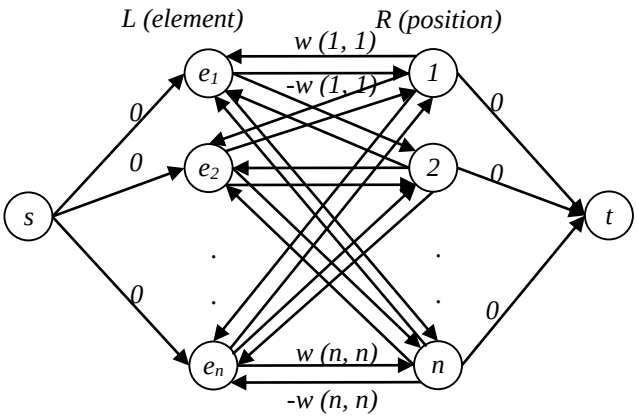

Figure 3.3. A flow network with antiparallel edges

To compute the minimum-cost perfect-matching is equivalent to computing the minimum weight flow with |f|=n, which is an iterative process as follows:

(1) compute the shortest path from *s* to *t* with respect to the weight function *w(i,j)* as defined in (19), then push 1 unit of flow from *s* to *t* along this path. The path is called an augmenting path in flow network *G'*.

(2) compute the residual network after flow augmentation.

(3) repeat step 1 and step 2 until the value of the flow |f|=n. Upon completing, the

www.manaraa.com

edges with *f(u,v)=1* are included in *M*, $\forall u \in L, \forall v \in R$. Figure 3.4 shows an example of the process.

The edges in *M* indicate the solution of the minimum-cost perfect-matching problem. To retrieve the solution for rank aggregation, if edge *(i,j)* $\in$ *M*, then item *i* should be positioned at position *j*, and so on. Since *M* is a perfect matching, it produces a full permutation ρ^* with minimum distance to the input orderings. The optimality of the ordering is guaranteed from the optimality of the minimum-cost perfect-matching solution.

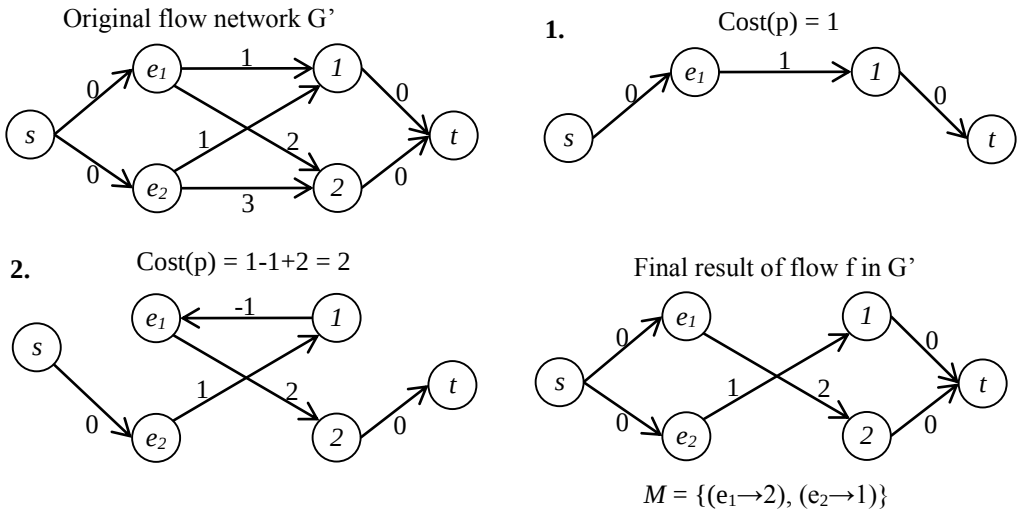

Figure 3.4. Using algorithm 2 to find the minimum-cost perfect matching

3.4.3. Rank Aggregation for Incomplete Lists (RAIL) Algorithm. The Spearman's Footrule distance assumes that the orderings are actually complete. This assumption cannot be made in this work since top-k lists are compared. The top-k lists are incomplete, i.e. all lists do not contain all of the elements. Therefore, there is the need to extend all lists such that all the elements appear in all the lists. The extended lists are termed *complete lists*.

Fagin et al. [39] suggest several techniques for creating complete list by appending the missing elements at the end of each list since they were clearly not considered to be in the top-k by that list.

www.manaraa.com

Definition (Missing Elements). Let $P = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ be a set of incomplete rankings and $\beta = \bigcup_{i=1}^{m} \sigma_i$, be the union set of *P*. The set of missing elements σ_i', for an incomplete ranking, σ_i, is given as:

$$\sigma_i' = \beta \backslash \sigma_i \tag{20}$$

Figure 3.5(a) shows and example of four incomplete lists and their missing elements. When appending the missing elements to the end of each list, the order in which the extra elements should be appended to the list is the challenge. Three solutions have been proposed:

1) Append the missing elements at location *l=k+1*, if element is not within the top-k [38].
2) Append the missing elements at location *l=(3k-z+1)/2*, which corresponds intuitively to placing the missing elements at an average location of the appended part. *z* is number of items in the intersection of two sets. *3k-z+1* is the average of *k+1* (beginning position) and *2k-z* (ending position). *2k-z* is also the size of the union of the two sets. In both cases, *l>k*, is a location parameter [38].
3) Append the elements in a random order, and define the distance as the average [38].

While the aforementioned solutions give a complete list, they do not necessarily agree with the rankings given by other lists, and therefore they do not produce a consistent ranking after aggregation. Considering σ_3 in Figure 3.5(a) with missing elements $\{S_1, S_3, S_7\}$. These missing elements will have the same rank, i.e. 5, when solution 1 is employed. However, S_1 has a rank of 1 in the other three lists and should therefore have a higher rank than S_3.

Due to this issue, a new recursive algorithm, Rank Aggregation for Incomplete Lists (RAIL), is proposed. It takes into account the relative position of missing elements in other top-k lists. In this algorithm, if two elements are missing in one ordering, but are ranked in other orderings, then the available rank from other lists will be used to determine their relative order in the ordering where they are missing. For this purpose,

www.manaraa.com

$\sigma'_{i,j}$ is used to denote the ranking of the missing elements in σ'_i whose order is consistent with input list σ_j.

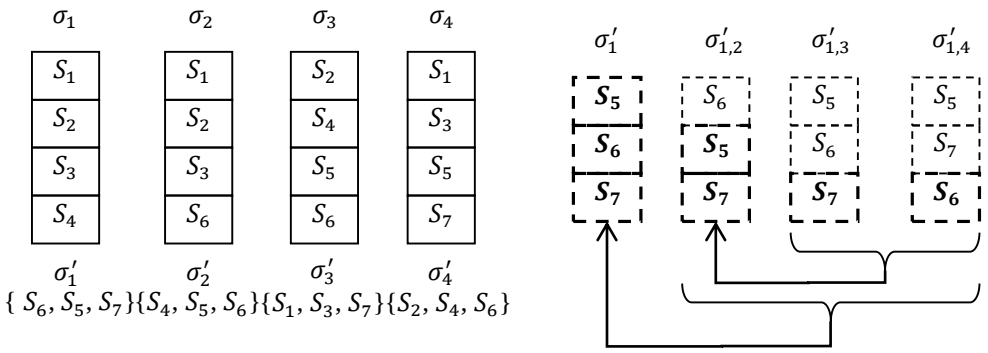

Figure 3.5. (a) Incomplete rankings and their missing elements. (b) Computing the ranks of missing elements of σ_1 from other lists

Figure 3.5(b), demonstrates how the RAIL algorithm is used to rank the missing elements in σ'_1. The missing elements in σ'_1 are identified to be $\{S_6, S_5, S_7\}$, their relative positions in other orderings are put in lists $\sigma'_{1,2}$, $\sigma'_{1,3}$, and $\sigma'_{1,4}$. In the figure, S_5, S_7, are already ranked in σ_4, so S_6 is appended after S_5 and S_7. Similarly, S_5, S_6 are already ranked in σ_3, so S_7 is subsequently appended. Now, the aggregate ranking of $\sigma'_{1,3}$ and $\sigma'_{1,4}$ are used to rank the list $\sigma'_{1,2}$. Finally, the aggregate ranking of $\sigma'_{1,2}$, $\sigma'_{1,3}$ and $\sigma'_{1,4}$ are used to rank the missing elements in σ'_1.

Once complete lists are obtained, the RACoL algorithm is used to compute its aggregate rank. Algorithm 3, is developed for this purpose.

3.5. EVALUATION

In this section, the two algorithms, RACoL and RAIL proposed in this work are evaluated. This is done by applying each algorithm to ranked lists of real-world airline services, the Openflights Dataset [34]. The ranked lists of real-world airline services, obtained from the service ranking engine, are based on multiple similar user requests. These ranked lists of airline services are aggregated using both RACoL and RAIL, and then discuss the results from each algorithm.

The Openflights Dataset [34] contains 61,199 routes between 3341 airports on 565 airlines spanning the globe. Each record in the dataset corresponds to an existing

www.manaraa.com

airline service as of February 2013. Each record contains the source and destination airports, airline, flight duration, flight distance and the number of stops.

ALGORITHM 3. RANK AGGREGATION FOR INCOMPLETE LISTS (RAIL)

Input: Incomplete ranked lists, $P = \{\sigma_1, \sigma_2 \dots \sigma_m\}$, $K = \{size(\sigma_1), size(\sigma_2) \dots size(\sigma_m)\}$
Output: The super list, SL.
initialize complete list, C ← P;
get the union list, $\beta = \{\sigma_1 \cup \sigma_2 \cup \dots \cup \sigma_m\}$ *and assign* $n \leftarrow |\beta|$*;*
for *i* from *1* to *m* **do**
    if *n-K*$_i$ *== 0* **then continue**;
    initialize missing elements list, $M = \emptyset$;
    compute the complement of σ_i : $\sigma_i' = \beta \backslash \sigma_i$;
    if *n-K*$_i$ *== 1* **then**
        append $\sigma_{i,0}'$ *at the end of Ci*;
        return;
    else if *n-K*$_i$ *> 1* **then**
        for *j* from *1* to *m* **do**
            if $j \neq i$ **then**
                get the intersection set $I_j = \sigma_i' \cap \sigma_j$ *while keeping the order in* σ_j;
                $S_j \leftarrow |I_j|$;
            end
        end
    end
    $R = \bigcup_{\substack{j=1 \\ j \neq i}}^{m} I_j$, and S= $\bigcup_{\substack{j=1 \\ j \neq i}}^{m} s_j$
    T ← RAIL(R, S)
    append T at the end of C;
end
SL ← RACoL(C)*;*
output *SL*;

3.5.1. RACoL Evaluation. RACoL algorithm is used to aggregate the four (4) ranked lists of services shown in Tables 3.6 to 3.9. These ranked lists were obtained from the *service ranking engine* based on four (4) similar requests that were submitted. The similar requests, each with different preferences on the non-functional attributes, were based on the following:

www.manaraa.com

Functionality: *Airline service from Atlanta Intl (ATL) to Detroit (DTW)*.

Trade-off Strategy: *(Price OR Stops) AND (Duration OR Reputation)* [1].

Table 3.6. Ranked Services Based on Request 1

Service	Route	Airline(s)
S_8	ATL→DTW	FL
S_{10}	ATL→DAY→DTW	FL→DL
S_7	ATL→FNT→DTW	FL→DL
S_2	ATL→DTW	DL
S_9	ATL→AVL→DTW	DL→DL
S_6	ATL→CAK→DTW	DL→DL
S_3	ATL→CLE→DTW	DL→DL
S_1	ATL→BNA→DTW	DL→DL
S_4	ATL→BNA→DTW	DL→WN
S_5	ATL→CLT→DTW	DL→US

Table 3.7. Ranked Services Based on Request 2

Service	Route	Airline(s)
S_{10}	ATL→DAY→DTW	FL→DL
S_2	ATL→DTW	DL
S_8	ATL→DTW	FL
S_9	ATL→AVL→DTW	DL→DL
S_7	ATL→FNT→DTW	FL→DL
S_6	ATL→CAK→DTW	DL→DL
S_1	ATL→BNA→DTW	DL→DL
S_3	ATL→CLE→DTW	DL→DL
S_5	ATL→CLT→DTW	DL→US
S_4	ATL→BNA→DTW	DL→WN

Table 3.8. Ranked Services Based on Request 3.

Service	Route	Airline(s)
S_8	ATL→DTW	FL
S_{10}	ATL→DAY→DTW	FL→DL
S_2	ATL→DTW	DL
S_9	ATL→AVL→DTW	DL→DL
S_7	ATL→FNT→DTW	FL→DL
S_6	ATL→CAK→DTW	DL→DL
S_3	ATL→CLE→DTW	DL→DL
S_1	ATL→BNA→DTW	DL→DL
S_4	ATL→BNA→DTW	DL→WN
S_5	ATL→CLT→DTW	DL→US

Table 3.9. Ranked Services Based on Request 4

Service	Route	Airline(s)
S_9	ATL→AVL→DTW	DL→DL
S_{10}	ATL→DAY→DTW	FL→DL
S_6	ATL→CAK→DTW	DL→DL
S_8	ATL→DTW	FL
S_7	ATL→FNT→DTW	FL→DL
S_3	ATL→CLE→DTW	DL→DL
S_2	ATL→DTW	DL
S_4	ATL→BNA→DTW	DL→WN
S_5	ATL→CLT→DTW	DL→US
S_1	ATL→BNA→DTW	DL→DL

www.manaraa.com

This evaluation shows the results of each step for the RACoL algorithm as described in section 3.4.2.

Step 1: Generate a complete bipartite graph with the services in one vertex set and positions in the other vertex set as shown in Figure 3.6.

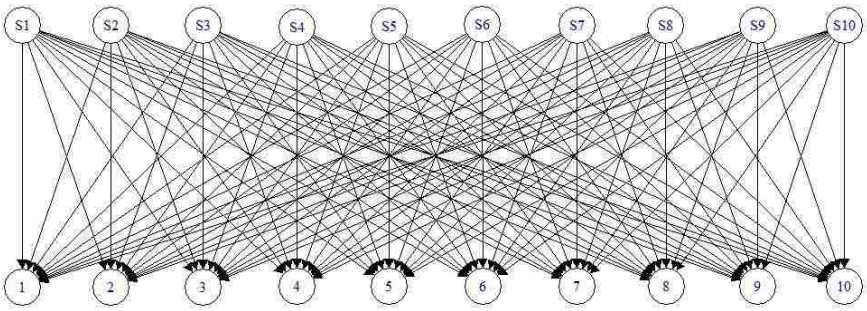

Figure 3.6. A complete bipartite graph

Step 2: Compute edge cost for each edge in Figure 3.6. Table 3.10 shows the edge costs for each edge in Figure 3.6.

Table 3.10. Edge Costs for all Edges in Figure 3.6

Edge	Cost	Edge	Cost	Edge	Cost	Edge	Cost	Edge	Cost
$S_1 \rightarrow 1$	29	$S_3 \rightarrow 1$	24	$S_5 \rightarrow 1$	34	$S_7 \rightarrow 1$	14	$S_9 \rightarrow 1$	10
$S_1 \rightarrow 2$	25	$S_3 \rightarrow 2$	20	$S_5 \rightarrow 2$	30	$S_7 \rightarrow 2$	10	$S_9 \rightarrow 2$	8
$S_1 \rightarrow 3$	21	$S_3 \rightarrow 3$	16	$S_5 \rightarrow 3$	26	$S_7 \rightarrow 3$	6	$S_9 \rightarrow 3$	6
$S_1 \rightarrow 4$	17	$S_3 \rightarrow 4$	12	$S_5 \rightarrow 4$	22	$S_7 \rightarrow 4$	4	$S_9 \rightarrow 4$	4
$S_1 \rightarrow 5$	13	$S_3 \rightarrow 5$	8	$S_5 \rightarrow 5$	18	$S_7 \rightarrow 5$	2	$S_9 \rightarrow 5$	6
$S_1 \rightarrow 6$	9	$S_3 \rightarrow 6$	4	$S_5 \rightarrow 6$	14	$S_7 \rightarrow 6$	6	$S_9 \rightarrow 6$	10
$S_1 \rightarrow 7$	5	$S_3 \rightarrow 7$	2	$S_5 \rightarrow 7$	10	$S_7 \rightarrow 7$	10	$S_9 \rightarrow 7$	14
$S_1 \rightarrow 8$	3	$S_3 \rightarrow 8$	4	$S_5 \rightarrow 8$	6	$S_7 \rightarrow 8$	14	$S_9 \rightarrow 8$	18
$S_1 \rightarrow 9$	5	$S_3 \rightarrow 9$	8	$S_5 \rightarrow 9$	2	$S_7 \rightarrow 9$	18	$S_9 \rightarrow 9$	22

www.manaraa.com

Table 3.10. Edge Costs for all Edges in Figure 3.6 (cont.)

Edge	Cost	Edge	Cost	Edge	Cost	Edge	Cost	Edge	Cost
$S_1 \rightarrow 10$	7	$S_3 \rightarrow 10$	12	$S_5 \rightarrow 10$	2	$S_7 \rightarrow 10$	22	$S_9 \rightarrow 10$	26
$S_2 \rightarrow 1$	12	$S_4 \rightarrow 1$	32	$S_6 \rightarrow 5$	17	$S_8 \rightarrow 1$	5	$S_{10} \rightarrow 1$	3
$S_2 \rightarrow 2$	8	$S_4 \rightarrow 2$	28	$S_6 \rightarrow 6$	13	$S_8 \rightarrow 2$	5	$S_{10} \rightarrow 2$	1
$S_2 \rightarrow 3$	6	$S_4 \rightarrow 3$	24	$S_6 \rightarrow 7$	9	$S_8 \rightarrow 3$	5	$S_{10} \rightarrow 3$	5
$S_2 \rightarrow 4$	6	$S_4 \rightarrow 4$	20	$S_6 \rightarrow 8$	7	$S_8 \rightarrow 4$	7	$S_{10} \rightarrow 4$	9
$S_2 \rightarrow 5$	8	$S_4 \rightarrow 5$	16	$S_6 \rightarrow 1$	5	$S_8 \rightarrow 5$	11	$S_{10} \rightarrow 5$	13
$S_2 \rightarrow 6$	10	$S_4 \rightarrow 6$	12	$S_6 \rightarrow 2$	3	$S_8 \rightarrow 6$	15	$S_{10} \rightarrow 6$	17
$S_2 \rightarrow 7$	12	$S_4 \rightarrow 7$	8	$S_6 \rightarrow 3$	7	$S_8 \rightarrow 7$	19	$S_{10} \rightarrow 7$	21
$S_2 \rightarrow 8$	16	$S_4 \rightarrow 8$	4	$S_6 \rightarrow 4$	11	$S_8 \rightarrow 8$	23	$S_{10} \rightarrow 8$	25
$S_2 \rightarrow 9$	20	$S_4 \rightarrow 9$	2	$S_6 \rightarrow 5$	15	$S_8 \rightarrow 9$	27	$S_{10} \rightarrow 9$	29
$S_2 \rightarrow 10$	24	$S_4 \rightarrow 10$	4	$S_6 \rightarrow 6$	19	$S_8 \rightarrow 10$	31	$S_{10} \rightarrow 10$	33

Step 3: Compute the minimum cost perfect matching algorithm on the bipartite graph. The solution of the minimum-cost perfect-matching is shown in Figure 3.7. This solution is then decoded and the ranked aggregated result is shown in Table 3.11.

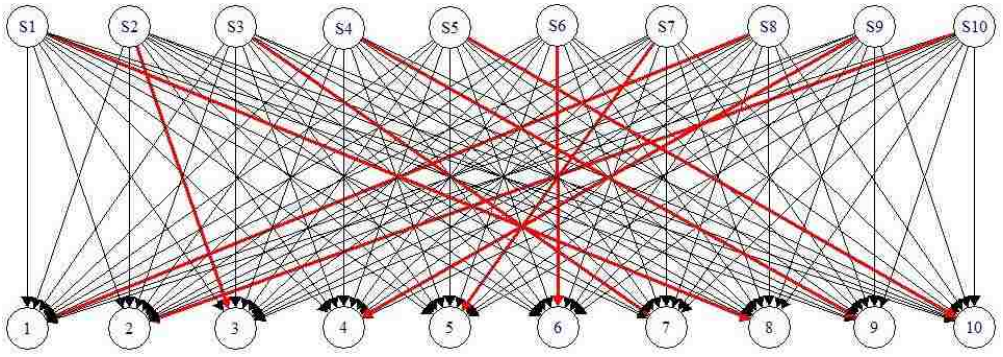

Figure 3.7. Solution of the minimum perfect matching algorithm

www.manaraa.com

Table 3.11. Aggregated Results

Service	Route	Airline(s)
S_8	ATL → DTW	FL
S_{10}	ATL → DAY → DTW	FL → DL
S_2	ATL → DTW	DL
S_9	ATL → AVL → DTW	DL → DL
S_7	ATL → FNT → DTW	FL → DL
S_6	ATL → CAK → DTW	DL → DL
S_3	ATL → CLE → DTW	DL → DL
S_1	ATL → BNA → DTW	DL → DL
S_4	ATL → BNA → DTW	DL → WN
S_5	ATL → CLT → DTW	DL → US

3.5.2. RAIL Evaluation. Similar to RACoL evaluation, RAIL algorithm was also evaluated using the four (4) ranked lists of services in Tables 3.12 to 3.15 obtained from the *service ranking engine*. The ranked lists were each from similar requests, with different preferences on the non-functional attributes. It can be observed that the results obtained are incomplete ranked services. These similar requests were based on the following:

Functionality: *Airline service from Denver Intl (DEN) to Madison-Dane(MSN).*

Trade-off Strategy: *(Price AND Stops) OR (Duration AND Reputation)* [1].

Table 3.12. Top-5 out of 3498 Services Based on Request 1

Service	Route	Airline(s)
S_1	DEN→MSN	UA
S_5	DEN→MSN	F9
S_6	DEN→ATL→MSN	DL→DL
S_7	DEN→DFW→ATL→MSN	F9→DL→DL
S_{10}	DEN→DCA→ATL→MSN	US→DL→DL

Table 3.13. Top-5 out of 3498 Services Based on Request 2

Service	Route	Airline(s)
S_1	DEN→MSN	UA
S_5	DEN→MSN	F9
S_{11}	DEN→EWR→MSN	UA→UA
S_6	DEN→ATL→MSN	DL→DL
S_{12}	DEN→DTW→MSN	UA→DL

www.manaraa.com

Table 3.14. Top-5 out of 3498 Services Based on Request 3

Service	Route	Airline(s)
S_1	DEN→MSN	UA
S_5	DEN→MSN	F9
S_{13}	DEN→DFW→ EWR →MSN	F9→AA→ UA
S_6	DEN→ATL→MSN	DL→DL
S_{12}	DEN→DTW→ MSN	UA→DL

Table 3.15. Top-5 out of 3498 Services Based on Request 4

Service	Route	Airline(s)
S_1	DEN→MSN	UA
S_5	DEN→MSN	F9
S_{11}	DEN→EWR→ MSN	UA→UA
S_{13}	DEN→DFW→EWR →MSN	F9→AA→UA
S_6	DEN→ATL→MSN	DL→DL

This evaluation shows the results of each step for the RAIL algorithm as described in section 3.4.3.

Step 1: Generate the complete bipartite graph with the services in one vertex set and positions in the other vertex set as shown in Figure 3.8. The services' vertex set consists of the unique services found in Tables 3.12 to 3.15.

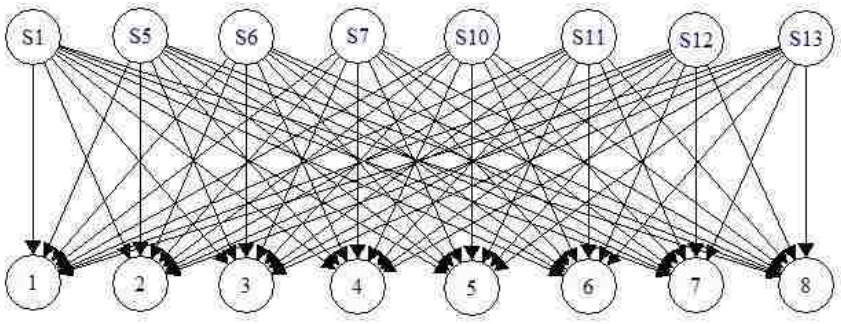

Figure 3.8. A complete bipartite graph

Step 2: Compute edge cost for each edge in Figure 3.8 using the four (4) different methods (including RAIL algorithm) discussed earlier in section 3.4.3. Each of these

www.manaraa.com

methods used the same initial complete bipartite graph shown in Figure 3.8. However, they produce different edge costs for the complete bipartite graph. The edge costs for the respective methods are tabulated in Tables 3.16 to 3.19.

Table 3.16. Edge Costs Using the l=k+1 Method

Edge	Cost	Edge	Cost	Edge	Cost
$S_1 \rightarrow 1$	0	$S_6 \rightarrow 7$	12	$S_{11} \rightarrow 5$	6
$S_1 \rightarrow 2$	4	$S_6 \rightarrow 8$	16	$S_{11} \rightarrow 6$	6
$S_1 \rightarrow 3$	8	$S_7 \rightarrow 1$	18	$S_{11} \rightarrow 7$	10
$S_1 \rightarrow 4$	12	$S_7 \rightarrow 2$	14	$S_{11} \rightarrow 8$	14
$S_1 \rightarrow 5$	16	$S_7 \rightarrow 3$	10	$S_{12} \rightarrow 1$	18
$S_1 \rightarrow 6$	20	$S_7 \rightarrow 4$	6	$S_{12} \rightarrow 2$	14
$S_1 \rightarrow 7$	24	$S_7 \rightarrow 5$	4	$S_{12} \rightarrow 3$	10
$S_1 \rightarrow 8$	28	$S_7 \rightarrow 6$	2	$S_{12} \rightarrow 4$	6
$S_5 \rightarrow 1$	4	$S_7 \rightarrow 7$	6	$S_{12} \rightarrow 5$	2
$S_5 \rightarrow 2$	0	$S_7 \rightarrow 8$	10	$S_{12} \rightarrow 6$	2
$S_5 \rightarrow 3$	4	$S_{10} \rightarrow 1$	19	$S_{12} \rightarrow 7$	6
$S_5 \rightarrow 4$	8	$S_{10} \rightarrow 2$	15	$S_{12} \rightarrow 8$	10
$S_5 \rightarrow 5$	12	$S_{10} \rightarrow 3$	11	$S_{13} \rightarrow 1$	15
$S_5 \rightarrow 6$	16	$S_{10} \rightarrow 4$	7	$S_{13} \rightarrow 2$	11
$S_5 \rightarrow 7$	20	$S_{10} \rightarrow 5$	3	$S_{13} \rightarrow 3$	7
$S_5 \rightarrow 8$	24	$S_{10} \rightarrow 6$	1	$S_{13} \rightarrow 4$	5
$S_6 \rightarrow 1$	12	$S_{10} \rightarrow 7$	5	$S_{13} \rightarrow 5$	5
$S_6 \rightarrow 2$	8	$S_{10} \rightarrow 8$	9	$S_{13} \rightarrow 6$	5
$S_6 \rightarrow 3$	4	$S_{11} \rightarrow 1$	14	$S_{13} \rightarrow 7$	9
$S_6 \rightarrow 4$	2	$S_{11} \rightarrow 2$	10	$S_{13} \rightarrow 8$	13
$S_6 \rightarrow 5$	4	$S_{11} \rightarrow 3$	6		
$S_6 \rightarrow 6$	8	$S_{11} \rightarrow 4$	6		

Table 3.17. Edge Costs Using the l=(3k-2z+1)/2 Method

Edge	Cost	Edge	Cost	Edge	Cost
$S_1 \rightarrow 1$	0	$S_6 \rightarrow 7$	12	$S_{11} \rightarrow 5$	5
$S_1 \rightarrow 2$	4	$S_6 \rightarrow 8$	16	$S_{11} \rightarrow 6$	7
$S_1 \rightarrow 3$	8	$S_7 \rightarrow 1$	15	$S_{11} \rightarrow 7$	11
$S_1 \rightarrow 4$	12	$S_7 \rightarrow 2$	11	$S_{11} \rightarrow 8$	15
$S_1 \rightarrow 5$	16	$S_7 \rightarrow 3$	7	$S_{12} \rightarrow 1$	17
$S_1 \rightarrow 6$	20	$S_7 \rightarrow 4$	3	$S_{12} \rightarrow 2$	13
$S_1 \rightarrow 7$	24	$S_7 \rightarrow 5$	1	$S_{12} \rightarrow 3$	9
$S_1 \rightarrow 8$	28	$S_7 \rightarrow 6$	5	$S_{12} \rightarrow 4$	5
$S_5 \rightarrow 1$	4	$S_7 \rightarrow 7$	9	$S_{12} \rightarrow 5$	1
$S_5 \rightarrow 2$	0	$S_7 \rightarrow 8$	13	$S_{12} \rightarrow 6$	3
$S_5 \rightarrow 3$	4	$S_{10} \rightarrow 1$	16	$S_{12} \rightarrow 7$	7
$S_5 \rightarrow 4$	8	$S_{10} \rightarrow 2$	12	$S_{12} \rightarrow 8$	11
$S_5 \rightarrow 5$	12	$S_{10} \rightarrow 3$	8	$S_{13} \rightarrow 1$	14
$S_5 \rightarrow 6$	16	$S_{10} \rightarrow 4$	4	$S_{13} \rightarrow 2$	10
$S_5 \rightarrow 7$	20	$S_{10} \rightarrow 5$	0	$S_{13} \rightarrow 3$	6
$S_5 \rightarrow 8$	24	$S_{10} \rightarrow 6$	4	$S_{13} \rightarrow 4$	4
$S_6 \rightarrow 1$	12	$S_{10} \rightarrow 7$	8	$S_{13} \rightarrow 5$	4
$S_6 \rightarrow 2$	8	$S_{10} \rightarrow 8$	12	$S_{13} \rightarrow 6$	6
$S_6 \rightarrow 3$	4	$S_{11} \rightarrow 1$	13	$S_{13} \rightarrow 7$	10
$S_6 \rightarrow 4$	2	$S_{11} \rightarrow 2$	9	$S_{13} \rightarrow 8$	14
$S_6 \rightarrow 5$	4	$S_{11} \rightarrow 3$	5		
$S_6 \rightarrow 6$	8	$S_{11} \rightarrow 4$	5		

www.manaraa.com

Table 3.18. Edge Costs Using the Random Position Method

Edge	Cost	Edge	Cost	Edge	Cost
$S_1 \rightarrow 1$	0	$S_6 \rightarrow 7$	12	$S_{11} \rightarrow 5$	8
$S_1 \rightarrow 2$	4	$S_6 \rightarrow 8$	16	$S_{11} \rightarrow 6$	8
$S_1 \rightarrow 3$	8	$S_7 \rightarrow 1$	18	$S_{11} \rightarrow 7$	10
$S_1 \rightarrow 4$	12	$S_7 \rightarrow 2$	14	$S_{11} \rightarrow 8$	12
$S_1 \rightarrow 5$	16	$S_7 \rightarrow 3$	10	$S_{12} \rightarrow 1$	21
$S_1 \rightarrow 6$	20	$S_7 \rightarrow 4$	6	$S_{12} \rightarrow 2$	17
$S_1 \rightarrow 7$	24	$S_7 \rightarrow 5$	4	$S_{12} \rightarrow 3$	13
$S_1 \rightarrow 8$	28	$S_7 \rightarrow 6$	2	$S_{12} \rightarrow 4$	9
$S_5 \rightarrow 1$	4	$S_7 \rightarrow 7$	6	$S_{12} \rightarrow 5$	5
$S_5 \rightarrow 2$	0	$S_7 \rightarrow 8$	10	$S_{12} \rightarrow 6$	5
$S_5 \rightarrow 3$	4	$S_{10} \rightarrow 1$	22	$S_{12} \rightarrow 7$	5
$S_5 \rightarrow 4$	8	$S_{10} \rightarrow 2$	18	$S_{12} \rightarrow 8$	7
$S_5 \rightarrow 5$	12	$S_{10} \rightarrow 3$	14	$S_{13} \rightarrow 1$	17
$S_5 \rightarrow 6$	16	$S_{10} \rightarrow 4$	10	$S_{13} \rightarrow 2$	13
$S_5 \rightarrow 7$	20	$S_{10} \rightarrow 5$	6	$S_{13} \rightarrow 3$	9
$S_5 \rightarrow 8$	24	$S_{10} \rightarrow 6$	4	$S_{13} \rightarrow 4$	7
$S_6 \rightarrow 1$	12	$S_{10} \rightarrow 7$	2	$S_{13} \rightarrow 5$	7
$S_6 \rightarrow 2$	8	$S_{10} \rightarrow 8$	6	$S_{13} \rightarrow 6$	7
$S_6 \rightarrow 3$	4	$S_{11} \rightarrow 1$	16	$S_{13} \rightarrow 7$	9
$S_6 \rightarrow 4$	2	$S_{11} \rightarrow 2$	12	$S_{13} \rightarrow 8$	11
$S_6 \rightarrow 5$	4	$S_{11} \rightarrow 3$	8		
$S_6 \rightarrow 6$	8	$S_{11} \rightarrow 4$	8		

Table 3.19. Edge Costs Using RAIL Algorithm

Edge	Cost	Edge	Cost	Edge	Cost
$S_1 \rightarrow 1$	0	$S_6 \rightarrow 7$	12	$S_{11} \rightarrow 5$	6
$S_1 \rightarrow 2$	4	$S_6 \rightarrow 8$	16	$S_{11} \rightarrow 6$	6
$S_1 \rightarrow 3$	8	$S_7 \rightarrow 1$	21	$S_{11} \rightarrow 7$	10
$S_1 \rightarrow 4$	12	$S_7 \rightarrow 2$	17	$S_{11} \rightarrow 8$	14
$S_1 \rightarrow 5$	16	$S_7 \rightarrow 3$	13	$S_{12} \rightarrow 1$	16
$S_1 \rightarrow 6$	20	$S_7 \rightarrow 4$	9	$S_{12} \rightarrow 2$	12
$S_1 \rightarrow 7$	24	$S_7 \rightarrow 5$	7	$S_{12} \rightarrow 3$	8
$S_1 \rightarrow 8$	28	$S_7 \rightarrow 6$	5	$S_{12} \rightarrow 4$	6
$S_5 \rightarrow 1$	4	$S_7 \rightarrow 7$	3	$S_{12} \rightarrow 5$	6
$S_5 \rightarrow 2$	0	$S_7 \rightarrow 8$	7	$S_{12} \rightarrow 6$	6
$S_5 \rightarrow 3$	4	$S_{10} \rightarrow 1$	25	$S_{12} \rightarrow 7$	8
$S_5 \rightarrow 4$	8	$S_{10} \rightarrow 2$	21	$S_{12} \rightarrow 8$	12
$S_5 \rightarrow 5$	12	$S_{10} \rightarrow 3$	17	$S_{13} \rightarrow 1$	20
$S_5 \rightarrow 6$	16	$S_{10} \rightarrow 4$	13	$S_{13} \rightarrow 2$	16
$S_5 \rightarrow 7$	20	$S_{10} \rightarrow 5$	9	$S_{13} \rightarrow 3$	12
$S_5 \rightarrow 8$	24	$S_{10} \rightarrow 6$	7	$S_{13} \rightarrow 4$	8
$S_6 \rightarrow 1$	12	$S_{10} \rightarrow 7$	5	$S_{13} \rightarrow 5$	4
$S_6 \rightarrow 2$	8	$S_{10} \rightarrow 8$	3	$S_{13} \rightarrow 6$	4
$S_6 \rightarrow 3$	4	$S_{11} \rightarrow 1$	14	$S_{13} \rightarrow 7$	6
$S_6 \rightarrow 4$	2	$S_{11} \rightarrow 2$	10	$S_{13} \rightarrow 8$	8
$S_6 \rightarrow 5$	4	$S_{11} \rightarrow 3$	6		
$S_6 \rightarrow 6$	8	$S_{11} \rightarrow 4$	6		

Step 3: Finally the minimum-cost perfect-matching problem is solved using the edge costs computed. Solutions to the minimum-cost perfect-matching problems are shown in Figures 3.9 to 3.12. The figures show the complete bipartite graphs together with the edges that constitute the solution of the perfect matching. It is obvious from the table that the different methods produced different *superlist*. (see Table 3.20).

www.manaraa.com

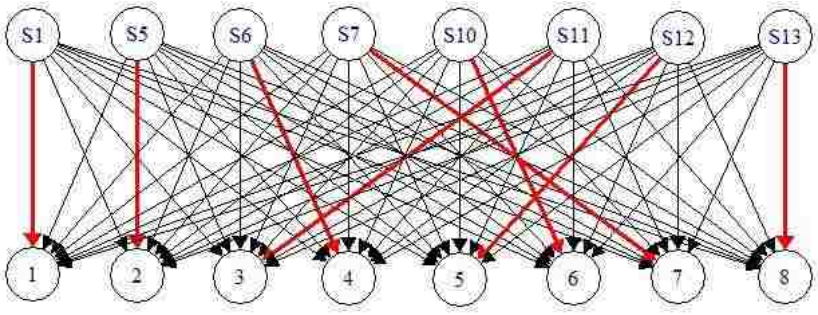

Figure 3.9. Solution to the minimum-cost perfect matching problem using l=k+1 method to calculate the edge costs

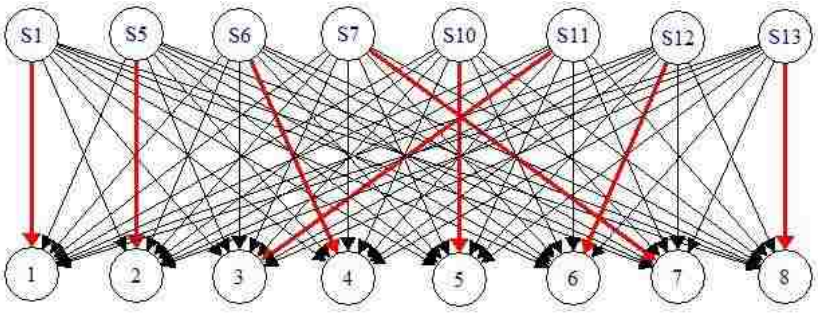

Figure 3.10. Solution to the minimum-cost perfect matching problem using l=(3k-2z+1)/2 method to calculate the edge costs

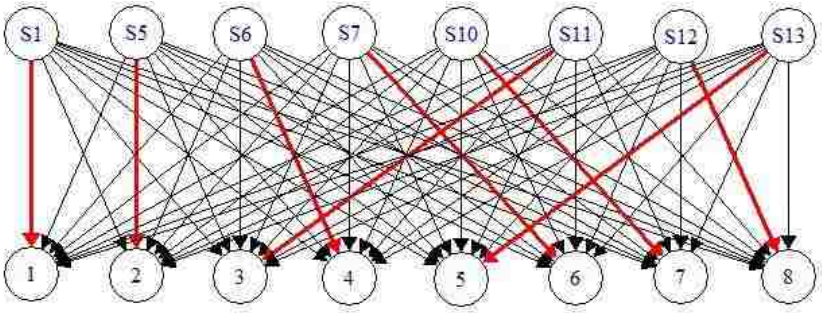

Figure 3.11. Solution to the minimum-cost-perfect matching problem using the random position method to calculate the edge costs

www.manaraa.com

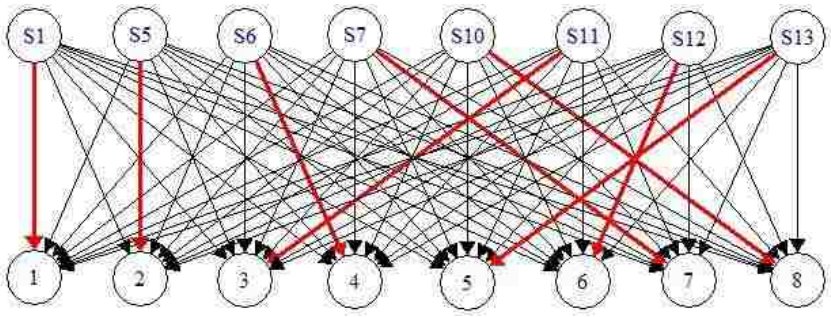

Figure 3.12. Solution to the minimum-cost-perfect matching problem using RAIL algorithm to calculate the edge costs

Table 3.20. Super Lists

Rank	*l=k+1* Method	*l=(3k-2z+1)/2* Method	Random Method	RAIL Algorithm
1	S_1	S_1	S_1	S_1
2	S_5	S_5	S_5	S_5
3	S_{11}	S_{11}	S_{11}	S_{11}
4	S_6	S_6	S_6	S_6
5	S_{12}	S_{10}	S_{13}	S_{12}
6	S_{10}	S_{12}	S_7	S_{13}
7	S_7	S_7	S_{10}	S_7
8	S_{13}	S_{13}	S_{12}	S_{10}

3.6. VALIDATION

In this section, experiments are performed to validate the results from RACoL and RAIL. For each proposed algorithm, and evaluation is performed using five (5) different sets of ranked lists (RL). Then a comparison of the results from each algorithm is made with existing methods. The results show that the proposed algorithms in this work perform better than those existing methods.

3.6.1. Validating Results from RACoL Algorithm. To validate results from the RACoL algorithm, its aggregated ranked list is compared with the aggregated ranked lists from two other methods, Borda Count and Reciprocal Rank, based on the Kemeny

www.manaraa.com

measure [48]. Figure 3.13 shows a graph of the comparison. The Kemeny measure shows how fit the aggregated ranked list is with respect to the set of its input ranked lists. A Kemeny meaure of 0 indicates a good fit, 1 indicates a revert fit and 0.5 is the random level fit [48]. From the graph, it can be seen that RACoL algorithm gives the best (smallest) Kemeny measure compared to the other methods.

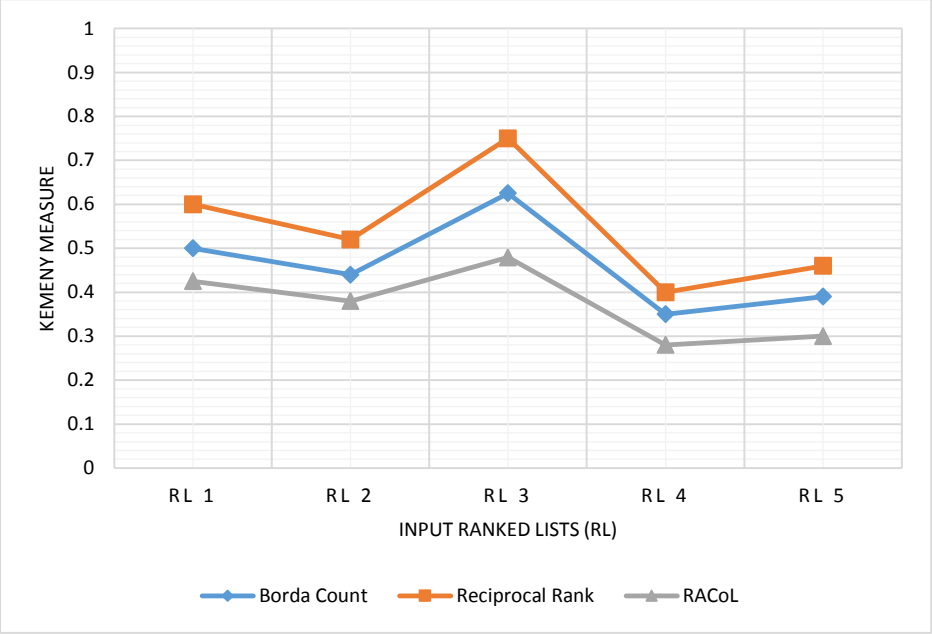

Figure 3.13. A graph showing a comparison of RACoL with Borda Count and Reciprocal Rank based on the Kemeny Measure of each solution to their original 5 input ranked lists

3.6.2. Validating Results from RAIL Algorithm. Results from the RAIL algorithm were validated in two ways. First, results from the RAIL algorithm were compared to the three (3) other methods based on the total minimum cost that produced the aggregated ranked list. Figure 3.14 shows a graph of this comparison. The minimum cost values in the graph have been normalized to the maximum possible cost a matching can have with respect to the number of lists and its elements. It is clear that when RAIL algorithm is used to compute the rank of missing elements, the total minimum cost is the lowest. One the other hand, when the $l = (3k-2z+1)/2$ method is used, the total minimum

www.manaraa.com

cost is the highest. Generally, input ranked list 4 has the lowest minimum costs compared to the other input ranked lists. This implies that most of services in input ranked list 4 have similar ranks.

RAIL was also validated by comparing its aggregated ranked list with the aggregated ranked lists from the other method based on the Kemeny measure [48]. Figure 3.15 shows a graph of the comparison. Here also, RAIL algorithm gives the best (smallest) Kemeny measure compared to the other methods considered in this work.

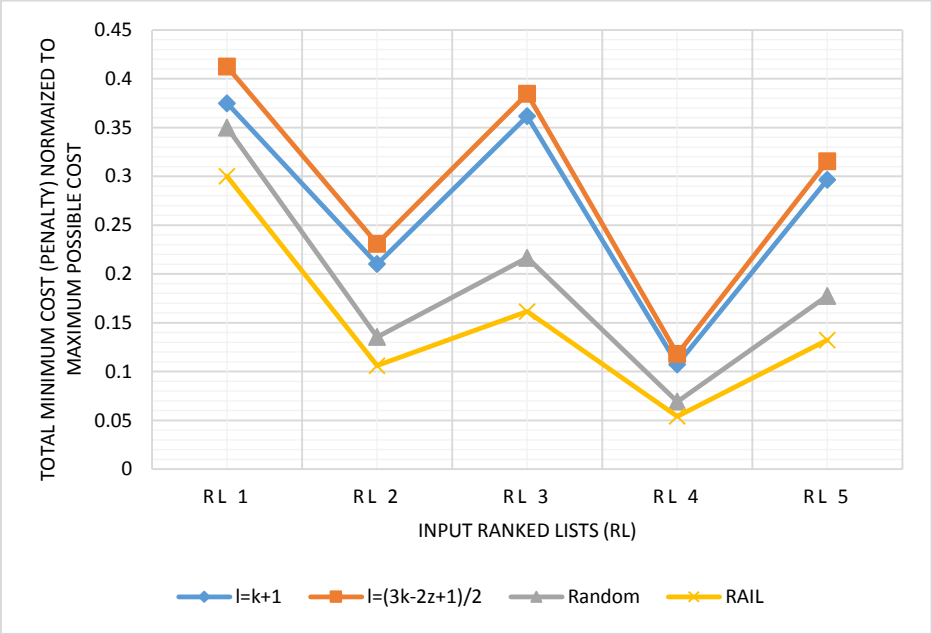

Figure 3.14. A graph showing the total minimum cost, normalized to maximum possible total cost, for each of the solution on the 5 ranked lists. A lower score indicates better selection. The total minimum cost is the total penalty for placing an item in a position as defined in (19)

3.7. CONCLUSIONS

In everyday life, service users are usually faced with the task of choosing a service from several sets of service search results (ranked services). This is typical in situations where several search results do not completely meet the user's preferences. It is impractical for users to choose an optimal service, based on their preference, from the multiple ranked lists just by inspection. This is because, each ranked list may contain huge

www.manaraa.com

number of services that makes it time consuming for users to compare them against services in other ranked lists. In order for users to choose a an optimal service from a set

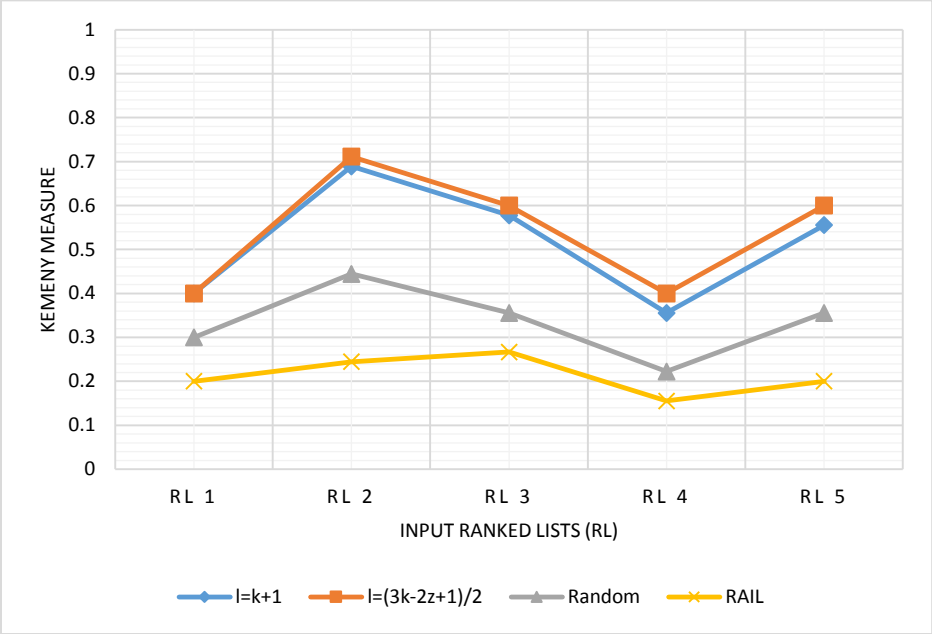

Figure 3.15. A graph showing the Kemeny Measure of each solution to their original input lists on the 5 input ranked lists

of ranked lists, a method that aggregates multiple ranked lists of services into a single aggregated ranked list is presented in this work. Top ranked services are subsequently selected for the user to choose from. The top ranked services represent the optimal services among the available ranked lists. two algorithms; 1) Rank Aggregation for Complete Lists (RACoL), that aggregates complete ranked lists and 2) Rank Aggregation for Incomplete Lists (RAIL) to aggregate incomplete ranked lists were also presented in detail. Both algorithms were evaluated by presenting examples using real-world flight services, open flights dataset. Finally, results from each algorithm were validated against other methods and have concluded that rank aggregation results from both algorithms closely represent the sets of ranked lists than using existing alternative approaches.

www.manaraa.com

4. A METHOD FOR PERSONALIZED PREFERENCE-BASED SERVICE RECOMMENDATION VIA COLLABORATIVE FILTERING

4.1. MOTIVATION AND SUMMARY OF CONTRIBUTIONS

In this section, a motivating example to show the research problem this section aims to address is presented. In this example, five service items (shown in Table 4.1) and five service users (shown in Table 4.2) were considered. Table 4.1 shows the five services together with the non-functional attribute values that describe them and Table 4.2 shows the five users and their respective overall personalized preferences. For the sake of simplicity, assume that all users have the same lowest and highest satisfaction values for each non-functional attribute, which indicates their individual personalized preference on those attributes. For instance, in this motivating example, the lowest and highest satisfaction for each user on the response time non-functional attribute is 10 and 3 secs respectively. This signifies that if a service responds is on average 3 secs or lower, the non-functional attribute is understood to be met. If the response time is increased from 3 secs to anything below 10 secs, the service satisfaction reduces accordingly. However, if the response time is greater than the threshold of 10 secs, the satisfaction degree is 0 and the service is completely unacceptable [1, 2]. In addition, the service invocation history of all users showing their satisfaction for each service is shown in Table 4.3.

Table 4.1. The List of Services and Their Non-Functional Attribute Values with Accounting Functionality

SERVICE	RELIABILITY (MONTHS)	AVAILABILITY (%)	THROUGHPUT (MBPS)	RESPONSE TIME (SECONDS)	PRICE ($)
$Service_1$	6	90	18.13	5	41
$Service_2$	10	97	28.25	7	21
$Service_3$	8	92	25.34	2	45
$Service_4$	6	98	8.29	1	27
$Service_5$	10	96	18.65	4	30

www.manaraa.com

Table 4.2. List of Service Requests from 5 Different Users

USER	PERSONALIZED PREFERENCE
1	$(NFR_{price}^{EI} \wedge NFR_{reliability}^{I}) \vee (NFR_{price}^{EI} \wedge NFR_{response\ time}^{VI})$
2	$(NFR_{price}^{EI} \wedge NFR_{reliability}^{I}) \vee ((NFR_{price}^{EI} \wedge NFR_{throughput}^{SI}) \vee (NFR_{price}^{EI} \wedge NFR_{availability}^{SI}))$
3	$(NFR_{price}^{VI} \wedge NFR_{response\ time}^{VI}) \wedge (NFR_{throughput}^{I} \wedge NFR_{availability}^{I})$
4	$(NFR_{price}^{I} \wedge NFR_{reliability}^{I}) \vee (NFR_{price}^{EI} \wedge NFR_{response\ time}^{I})$
5	$(NFR_{price}^{EI} \wedge NFR_{reliability}^{I}) \otimes ((NFR_{price}^{EI} \wedge NFR_{throughput}^{SI})$

Table 4.3. User-Service Matrix Indicating Invoked Services and Their Satisfaction

	$Service_1$	$Service_2$	$Service_3$	$Service_4$	$Service_5$
User1	35%	70%	19%	?	
User2		70%	19%		70%
User3	0%				
User4			?	87%	
User5		50%			36%

Let's assume user 1 to be the active user. Let's also assume the task is to determine whether or not service 4 should be recommend to this active user. To do this, current recommendation systems employ weighted average with mean offset [3, 4, 11] or its extension to compute the missing values of an active user. It is typically done by computing the weighted average of the neighboring users' non-functional attribute values using similarity as the weights. This makes the choice of similarity function a critical decision in recommendation systems. The Pearson correlation coefficient (PCC) [3, 4, 10, 11] or its extension are the widely used similarity functions to compute the similarity between any two users in memory-based CF. The similarity function finds the similar neighbors of user 1 based. These will be users who have reported satisfaction value for service 4 and share some commonly invoked services with user 1. In this scenario, using PCC as similarity function, user 1 will have no neighbors since the only user that has

www.manaraa.com

invoked service 4, i.e. user 4 share no commonly invoked services with user 1. Therefore, using classical collaborative filtering methods will not recommend service 4 to user 1.

However, although user 4 share no common invoked service(s) with user 1, it can be argued that user 4 must be a neighbor of user 1. This is because, users 4 and 1 have the same preferences and must be considered as similar users. In fact, it can be inferred that due to the highly positive similarity in their preferences, service 4 should be recommended to user 1. Therefore, a similarity function that considers the preferences of users provides accurate similarity values.

This scenario shows that having an inaccurate similarity values will adversely impact the prediction accuracy of missing non-functional attribute values and hence the recommended services. It is therefore necessary to incorporate user's personalized preference on non-functional attribute when computing similarity between users or service items for personalized service recommendation.

The main contributions of this work are summarized as follows:

(1) To accurately compute the similarity between users or service items, users' personalized preferences on non-functional attributes must be incorporated in the proposed similarity function. This is done as follows:
   (a) For users who do not share any past experience on service item(s), instead of assuming that such users are not similar, their personalized preferences on non-functional attributes are used to find the similarity between them.
   (b) For users who share some past experience on service item(s), their similarity is obtained by including the satisfaction of their personalized preferences on non-functional attributes by extending the Pearson Correlation Coefficient.

(2) For the proposed prediction function, the weighted average with mean offset is extended to also include the satisfaction of users' non-functional attribute based on their personalized preferences. This will predict the satisfaction of the active user's non-functional attribute.

(3) Finally, comprehensive experiments were conducted to evaluate the proposed method by employing real-world web service non-functional attribute data set [11]. The method is validated by comparing it to well-known service recommendation systems, WSRec [11] and PHCF [4].

www.manaraa.com

4.2. BACKGROUND AND RELATED WORK

In this section, related work regarding collaborative filtering (CF) recommendation method is discussed in general, specifically, memory based CF. Related work focusing on personalized service recommendation methods is also presented.

4.2.1. Service Recommendation Based on Collaborative Filtering. Collaborative filtering (CF) is a popular and solid recommendation algorithm that bases its predictions and recommendations on the ratings or behavior of other users in the system [3]. It assumes that, if users agree about the quality or relevance of some items, then they are likely to agree about other items as well. There are two main categories of CF – memory-based and model-based CF methods [10]. The most analyzed examples of memory-based CF methods include user-based approaches and item-based approaches [10]. The proposed method in this work employs the memory-based CF method.

The user-based CF, also known as the k-NN CF, aims at finding other users whose past rating behavior is similar to that of the current user. It then uses their ratings on other items to predict what the current user will like. It achieves this by using some similarity function to compute the similarity between users. The user similarity value is in the interval of [-1, 1], with a larger value indicating that the two users are more similar [3]. Using the identified similar users, a rating value is usually predicted for all missing items in the target user's profile. Item-based CF methods use a similar idea to user-based CF methods except that they compute similarity between items as opposed to users as is the case of user-based CF. A rating value is also predicted for all missing items in the target user's profile using the similar items identified.

There are limited research works that have employed memory-based CF methods to service recommendation. While some of these works used either user-based CF or item-based CF, others focus on hybrid memory based CF methods (a combination of the user-based and item-based CF). Shao et al. [51] proposed a user-based CF algorithm to predict QoS values. Zheng et al. [11] used a hybrid CF algorithm to recommend web services. Sreenath and Singh [52] and Rong et al. [53] applied the idea of CF in their systems, and used MovieLens data [54] for experimental analysis. The above mentioned research works neither considered users' personalized preferences on QoS and therefore the prediction accuracy of these methods was unsatisfactory.

www.manaraa.com

4.2.2. Personalized Service Recommendation. Personalized service recommendation has been studied in recommendation systems. Jiang et al. [4] proposed a hybrid personalized CF-based recommendation method that considers the contribution of an object (service item) to the similarity degree between users. Their method was based on the notion that, if two users invoked the same service item in the past, it does not guarantee that those users are similar. In their work, they determined the contribution of a service item by computing the standard deviation of the QoS metrics for the service item. Shao et al. [51] also proposed an approach for personalized QoS prediction for web services via CF that considers the different experiences of users on the quality of the same web service. Their approach predicts QoS for web services, taking the similarity among consumers' experiences into consideration. Their assumption was that consumers, who have similar historical experiences on some services, would have similar experiences on other services. Chen et al. [10] proposed a personalized QoS-aware recommendation method that considers the QoS variance according to users' locations to recommend services. The basic idea of their method was that users closely located with each other are more likely to have similar service experience than those who live far away from each other.

Although the methods discussed above aim at personalizing service recommendation either through location or user experiences, none of these methods consider users personalized preferences on non-functional attributes to recommend services. Due to this, these existing recommendation methods suffer from low prediction accuracy. An effective CF algorithm for service recommendation that considers users' personalized preferences on non-functional attributes is proposed. Comprehensive experiments conducted with real-world data show that the proposed method outperforms others.

4.3. PERSONALIZED PREFERENCE COLLABORATIVE FILTERING METHOD

Figure 4.1 shows the framework of the proposed personalized preference CF method for service recommendation. Prior to recommending services, it is necessary to know the history of an active user with respect to his/her non-functional attribute. This is

www.manaraa.com

important because non-functional attribute information plays part in making accurate service recommendations. Due to this, our method collects historical non-functional attribute record of active users and stores this information in the non-functional attribute values history repository. Besides the non-functional attribute values history, it is also necessary to obtain the active user's personalized preference in order to personalize the services recommended to him/her. The personalized preference component is used to collect this information. The active user can specify his/her personalized preferences using the specification described in section 1.

Using both the non-functional attribute historical data and the personalized preference of the active user, the satisfaction of this user can be computed for each service in the service repository. Based on the satisfaction of the services, the similarity between users can then be computed and subsequently similar users (in case of user-based personalized preference recommendation) or similar items (in case of the item-based personalized preference recommendation) can be identified. Once the similar users and/or similar items are obtained, the respective missing values of the active user are predicted. Finally, the recommender weighs the two predicted values to recommend optimal services to the active user.

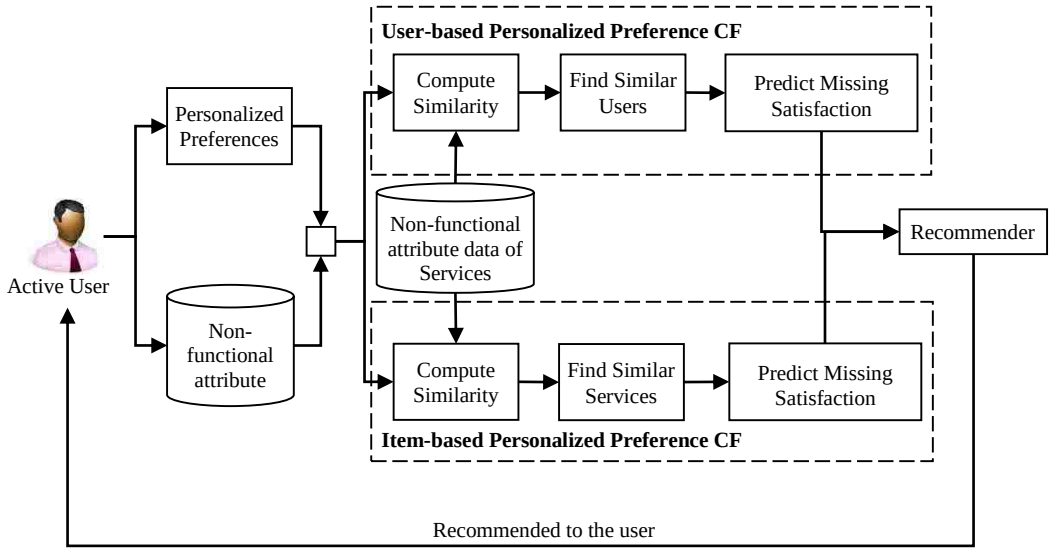

Figure 4.1. Framework of the personalized preference collaborative filtering method for service recommendation

www.manaraa.com

4.3.1. Problem Formulation. Formally, a service recommendation system consists of m service users, $U = \{u_1, u_2, ..., u_m\}$, and n service items, $S = \{s_1, s_2, ..., s_n\}$. The relationship between service users and service items can be denoted by a user-item matrix, $U \times S$. Each entry in this matrix, $r_{m,n}$, represents a vector, of non-functional attribute values, which is obtained by the service user m on the service item n. If user m did not invoke service item n, then $r_{m,n} = 0$.

4.4. PERSONALIZED PREFERENCE RECOMMENDATION ALGORITHM

As indicated in Section 4.3, the proposed personalized preference recommendation algorithm is formulated by using the memory-based CF method. This sections discusses the different aspects of the algorithm.

4.4.1. Similarity Computation. Finding the best choice of similarity function in CF-based service recommendation is a critical decision because the accuracy of the overall service recommendation depends on the accuracy of the similarity function [11]. Several different similarity functions have been proposed and evaluated in literature [3]. These include the Pearson correlation coefficient, constrained Pearson correlation coefficient, spearman rank coefficient, and cosine similarity. In general, Pearson correlation coefficient has been found to provide the best results [3], although results from other research works suggest that the constrained Pearson correlation coefficient may provide some improvement when items are rated on an absolute scale [3]. Due to this the Pearson correlation coefficient is adopted and extended for the proposed personalized preference similarity computation in this work.

Pearson Correlation Coefficient (PCC) [3, 4, 11] has been employed in a number of recommender systems for similarity computation. It computes the statistical correlation between two non-functional attribute values to determine their similarity. In user-based CF, PCC is used to compute the similarity between two service users based on their co-invoked services. PCC lies in the interval [-1, 1]. For any two users, the more positive the PCC, the more similar the two users are.

Formally, let a and u be two service users. The degree of similarity between these two users, $Sim_{PCC}(a,u)$, using PCC, is computed as:

www.manaraa.com

$$Sim_{PCC}(a,u) = \frac{\sum_{i \in I}(r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})}{\sqrt{\sum_{i \in I}(r_{a,i} - \overline{r_a})^2}\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r_u})^2}} \quad (21)$$

where, $I = I_a \cap I_u$ is the set of co-invoked service items by both users a and u, $r_{a,i}$ and $r_{u,i}$ be the respective non-functional attribute values that were observed by users a and u, when they both invoked service item i, and $\overline{r_a}$ and $\overline{r_u}$ represent the mean non-functional attribute value of users a and u respectively.

Similarly, for item-based CF, PCC is used to compute the similarity between two service items based on the common users that invoked the services. For any two service items, the more positive the PCC, the more similar the two service items are.

Formally, let i and j be two service items. The degree of similarity between these two service items, $Sim_{PCC}(i, j)$, using PCC, is computed as:

$$Sim_{PCC}(i, j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_i})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{r_j})^2}} \quad (22)$$

where, $U = U_i \cap U_j$ is the set of users that invoked both service items i and j, $r_{u,i}$ and $r_{u,j}$ be the respective non-functional attribute values that were observed by user u, when he/she invoked service items i and j, and $\overline{r_i}$ and $\overline{r_j}$ represent the mean non-functional attribute value of service items i and j respectively.

Using PCC to compute similarity between service users or items has some limitations. Firstly, as shown in equations (5) and (6), PCC considers the non-functional attribute values itself in its similarity computation without any regard to the personalized preferences of users. For this reason, PCC often overestimates the similarities of service users and/or service items, especially, those with few co-invoked services or common users [11]. One way to address this problem as proposed by Zheng et al. [11], is to employ a similarity weight to reduce the influence of a small number of similar co-invoked items. However, their method does not incorporate users' personalized

www.manaraa.com

preference thereby creating a gap between users' non-functional attribute values and their satisfaction on the services. The non-functional attribute values supplied by users may not necessarily represent their satisfaction based on users' personalized preference on that non-functional attribute. Therefore, in order to bridge the gap between users' non-functional attribute values and their satisfaction, their personalized preference should be considered in similarity computation. Intuitively, if a non-functional attribute value does not satisfy a user's personalized preference it should not be included in the similarity computation that involves that user.

Secondly, PCC strongly relies on the co-invoked services between users for its similarity computation. For this reason, it assumes that if two users or two service items have no co-invoked services (i.e. $I = \varnothing$ or $U = \varnothing$), then those two users or service items are not similar at all (i.e. $Sim_{PCC}(a,u) = Sim_{PCC}(i,j) = 0$). While this might be true for some users, it is not always accurate, especially for users with no service invocation history. For instance, consider the list of users, their respective invoked services and personalized preferences as shown in Table 4.4. PCC will estimate the similarity between users 1 and 3 to be 0 ($Sim_{PCC}(user_1, user_3) = 0$), because there are no co-invoked services between users 1 and 3. However, users 1 and 3 share some similarity based on their personalized preferences. In fact, it can be argued that, based on the personalized preferences of users 1 and 3, they are as similar as users 2 and 3 even though users 1 and 3 have no co-invoked services.

Table 4.4. List of Users, Their Invoked Services and Personalized Preferences

Users	Response Time Values of Services (secs)				Personalized Preference	
	Service1	Service2	Service3	Service4	Lowest	Highest
$User_1$	-	-	-	-	0.62	0.11
$User_2$	0.33	0.25	0.12	0.08	0.62	0.11
$User_3$	0.33	0.25	0.12	0.08	0.62	0.11

Due to the above reasons, in order to accurately compute similarity between any two service users or service items, it is necessary to incorporate users personalized

www.manaraa.com

preference on the non-functional attribute in question. For this purpose, a similarity function based on the satisfaction of a user's personalized preference is defined.

Definition 5.1 (Similarity Function Based on Personalized Preferences). Let a and u be two service users and $I = I_a \cap I_u$ be the set of co-invoked service items by both users a and u. The degree of similarity between these two users based on their personalized preference $Sim_{PPBased}(a,u)$ is defined as:

$$Sim_{PPBased}(a,u) = \delta \times Sim_{PPre}(a,u) + (1-\delta) \times Sim_{PSat}(a,u) \quad (23)$$

where $Sim_{PPre}(a,u)$ is the degree of similarity between the personalized preferences of users a and u if there are no co-invoked services between them, $Sim_{PSat}(a,u)$ is the degree of similarity of users a and u based on the satisfaction of their personalized preferences, if they have co-invoked service items, and δ is a tunable parameter which determines which method to use.

Similarly, if i and j are two service items, and $U = U_i \cap U_j$ is the set of users with invoked service items i and j, then the degree of similarity between these two service items based on the personalized preference of user u that invoked the service items $Sim_{PPBased}(i,j)$ is defined as:

$$Sim_{PPBased}(i,j) = \delta \times Sim_{PPre}(i,j) + (1-\delta) \times Sim_{PSat}(i,j) \quad (24)$$

where $Sim_{PPre}(i,j)$ is the similarity between the personalized preferences of user u when he/she invoked service items i and j, $Sim_{PSat}(i,j)$ is the degree of similarity between service items if i and j are based on the satisfaction of the personalized preference of user u, and δ is a tunable parameter which determines which method to use.

5.1.1. Degree of Similarity between Personalized Preferences. Finding similarity between users personalized preference on non-functional attributes is very necessary in situations where the users do not share any past experience (see equation 7 and 8). As discussed in section 4.3, user's personalized preference on non-functional attribute is captured using a method based on fuzzy logic. Therefore, finding the degree of similarity

www.manaraa.com

between personalized preferences, is basically to find the similarity between membership functions (fuzzy sets). The most obvious way of computing similarity of fuzzy sets is based on their distance. This involves two steps: first, the distance between the two fuzzy sets is obtained by a distance measure and second, one of the relationships between similarity and distance comes into play to reach at the degree of similarity.

Definition 5.2 (Similarity between User Preferences). Given two service users a and u, their personalized preferences P_a^i and P_u^i on some non-functional attribute i, and a distance measure D, the degree of similarity between the overall preference (taking into consideration all the non-functional attributes that describe the service) of users a and u, $Sim_{PPre}(a,u)$, using distance based assessment proposed by Koczy [55] is given by:

$$Sim_{PPre}(a,u) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{1+D(P_a^i, P_u^i)}\right) \tag{25}$$

There are many choices for D. In this work, the normalized Hamming distance [55] is considered. It is one of the most commonly employed distance measures and is constructed for a finite universe [Beg and Ashraf 2009], given as:

$$D_{nH}(P_a^i, P_u^i) = \frac{1}{n}\sum_{j=1}^{n}\left|MF_{x_j}(a,i) - MF_{x_j}(u,i)\right| \tag{26}$$

where $MF_{x_j}(a,i)$ and $MF_{x_j}(u,i)$ are the membership functions that defines the personalized preference of users a and u respectively on the non-functional attribute i at point x_j and n is the number of points in the universe X.

As an example, let us consider the personalized preference of users a and u on response time (rt) non-functional attribute as follows:

www.manaraa.com

$$P_a^{rt} = MF(a, rt) = \begin{cases} 0 & , \text{if } rt \geq 0.5 \\ 1 & , \text{if } rt \leq 0.3 \\ \dfrac{0.5 - rt}{0.2} & , \text{otherwise} \end{cases}$$

$$P_u^{rt} = MF(u, rt) = \begin{cases} 0 & , \text{if } rt \geq 0.7 \\ 1 & , \text{if } rt \leq 0.1 \\ \dfrac{0.7 - rt}{0.6} & , \text{otherwise} \end{cases}$$

If $X = \{0,\ 0.2,\ 0.4,\ 0.6,\ 0.8\}$, then the normalized hamming distance can be computed as:

$$D_{nH}(P_a^{rt}, P_u^{rt}) = \frac{|1-1| + |1-0.83| + |0.5-0.5| + |0-0.166|}{5}$$
$$= 0.0672$$

And the degree of similarity between users a and u based on their personalized preference P_a^{rt} and P_u^{rt} is:

$$Sim_{PPre}(a, u) = \frac{1}{1 + 0.0672}$$
$$= 0.937$$

5.1.2. Degree of Similarity based on Satisfaction of Personalized Preferences. For users who have had some past experiences on some service item(s), the similarity is computed by extending the Pearson Correlation Coefficient to include user's satisfaction of their personalized preferences on non-functional attributes.

Definition 5.3 (Similarity between Satisfaction of Users Personalized Preferences). Let a and u be two service users and P_a and P_u be the personalized preference of users a and u respectively. The degree of similarity between these two users, based on the satisfaction of their personalized preferences, $Sim_{PSat}(a, u)$ is computed as:

www.manaraa.com

$$Sim_{PSat}(a,u) = \frac{\sum_{i \in I} (Sat_{P_a}(S_i) - \overline{Sat_{P_a}})(Sat_{P_u}(S_i) - \overline{Sat_{Pu}})}{\sqrt{\sum_{i \in I} (Sat_{P_a}(S_i) - \overline{Sat_{P_a}})^2} \sqrt{\sum_{i \in I} (Sat_{P_u}(S_i) - \overline{Sat_{Pu}})^2}} \quad (27)$$

where, $I = I_a \cap I_u$ is the set of co-invoked service items by both users a and u, $Sat_{P_a}(S_i)$ and $Sat_{P_u}(S_i)$ are the respective satisfaction degrees of service item S_i based on the personalized preference of users a and u, and $\overline{Sat_{P_a^i}}$ and $\overline{Sat_{P_u}}$ represent the mean satisfaction degrees based on the personalized preference of users a and u respectively.

Similarly, let i and j be two service items and P_u be the personalized preference of a user u. The degree of similarity between these two service items based on the satisfaction of the personalized preference of user u, $Sim_{PSat}(i, j)$ is computed as:

$$Sim_{PSat}(i, j) = \frac{\sum_{u \in U} (Sat_{P_u}(S_i) - \overline{Sat_{P_*}(S_i)})(Sat_{P_u}(S_j) - \overline{Sat_{P_*}(S_j)})}{\sqrt{\sum_{u \in U} (Sat_{P_u}(S_i) - \overline{Sat_{P_*}(S_i)})^2} \sqrt{\sum_{u \in U} (Sat_{P_u}(S_j) - \overline{Sat_{P_*}(S_j)})^2}} \quad (28)$$

where, $U = U_i \cap U_j$ is the set of users with invoked service items i and j, $Sat_{P_u}(S_i)$ and $Sat_{P_u}(S_j)$ be the respective satisfaction degrees of service items S_i and S_j based on the personalized preference P_u, of user u, and $\overline{Sat_{P_*}(S_i)}$ and $\overline{Sat_{P_*}(S_j)}$ represent the mean satisfaction degrees of service items S_i and S_j based on the personalized preference P_*, of all users that have invoked service items S_i and S_j respectively.

4.4.2. Similar Neighbor Determination. After calculating the similarities between different users, a set of similar neighbors, N, can be identified based on the similarity values. The selection of similar neighbors is a very important step for making accurate recommendation, since dissimilar neighbors will lead to inaccurate missing value prediction for an active user [11]. The traditional Top-K algorithm is employed to find the similar neighbors, N, for an active user.

4.4.3. Missing Satisfaction Value Prediction. With similar neighbors, N, of the active user identified, predictions for an active user's non-functional attribute value can

www.manaraa.com

be generated for a service item S_i. This is done by combining the satisfaction values of users in N. This is typically done by computing the weighted average with mean offset [3 , 4, 11] of the neighboring users. This function is extended to compute the weighted mean offset of the satisfaction values of users in N using the computed similarity values as weights. Thus for an active user a, the predicted satisfaction value for a service item S_i, $\widehat{Sat_{P_a}}(S_i)$, using the degree of similarity between users, based on the satisfaction of their personalized preferences, $Sim_{PSat}(a,u)$ can be computed as follows:

$$\widehat{Sat_{P_a}}(S_i) = \overline{Sat_{P_a}} + \frac{\sum_{u \in N} Sim_{PPBased}(a,u)(Sat_{P_u}(S_i) - \overline{Sat_{P_u}})}{\sum_{u \in N} Sim_{PPBased}(a,u)} \tag{29}$$

where $\overline{Sat_{P_a}}$ is the vector of average satisfaction value of different services based on the personalized preference of the active user P_a, and $\overline{Sat_{P_u}}$ is the vector of average satisfaction value of different services based on the personalized preference of the similar service user P_u.

Similarly, the satisfaction value for a service item S_i, $\widehat{Sat_{P_a}}(S_i)$, of an active user a, can be predicted using the degree of similarity between service items, based on the satisfaction of their personalized preferences, $Sim_{PSat}(i, j)$ as follows:

$$\widehat{Sat_{P_a}}(S_i) = \overline{Sat_{P_a}} + \frac{\sum_{u \in N} Sim_{PPBased}(i, j)(Sat_{P_u}(S_i) - \overline{Sat_{P_u}})}{\sum_{u \in N} Sim_{PPBased}(i, j)} \tag{30}$$

where $\overline{Sat_{P_a}}$ is the vector of average satisfaction value of different services based on the personalized preference of the active user P_a, and $\overline{Sat_{P_u}}$ is the vector of average satisfaction value of different services based on the personalized preference of the similar service user P_u.

www.manaraa.com

4.4.4. Service Recommendation. To recommend service(s) to the active user, the two predicted satisfaction values (satisfaction values from user-based and item-based) must be combined in a certain fashion. Since these two predicted values may have different prediction performance, the tunable parameter method [4, 11] was adopted to combine the two values using the equation below:

$$Sat_{overall} = \mu \times Sat_{user-based} + (1-\mu) \times Sat_{item-based} \quad (31)$$

where μ is the tunable parameter which determines which method to use (either the user-based, the item-based or both).

Once the overall predicted satisfaction value is obtained, services are recommended to the active user based on this value.

4.5. EXPERIMENTS

Experiments were conducted to evaluate and validate the proposed personalized preference service recommendation method (PPSR). The experiments were performed on QWS dataset, a real-world web service QoS performance dataset.

4.5.1. Dataset Description and Experimental Setup. The QWS Dataset [7], contains 2,507K records. The majority of services were obtained from public sources on the Web including Universal Description, Discovery, and Integration (UDDI) registries, search engines, and service portals [7]. Each record in the dataset corresponds to an existing service on the web as of September 2008. For each service, eleven (11) different parameters representing non-functional attributes exist. Six (6) of these non-functional attributes are selected for this work. Their values represent averages of the measurements collected during a six-day period [7]. The selected non-functional attributes, their descriptions and their units are shown in Table 4.5. Since the preference of users were not available, preferences were randomly generated for 64 users in the dataset. Below are a few of them.

User 6: (Reliability ⊗ ResponseTime) ∧ (Availability ⊗ Throughput)

User 37: (Reliability ⊗ ResponseTime) ∨ (ResponseTime ∧ Throughput)

www.manaraa.com

User 61: (Reliability ∧ Availability) ⊗ (ResponseTime ∧ Throughput)

To make the simulations more realistic, 90% and 70% of response time and throughput values were randomly removed from the data and generate four sparse matrices with density 10% and 30%, respectively for the training data set. The focus is to have a very sparse dataset matrices in order to see how the proposed personalized preference recommendation system works on users with no or few co-invoked services. Typically, active users have only a small number of invoked services [3, 11]. Due to this, some records of some users were also removed and these users were randomly selected as active users. For each non-functional attribute, the number of values made available to active users were varied from 10, 20, and 30, and name them Given 10, Given 20, and Given 30, respectively. The proposed method in this section was then used to predict the missing satisfaction values of active users and subsequently recommend services to them.

Table 4.5. Non-functional Attributes, Their Descriptions, and Units

Non-functional Attribute	Description	Unit
Response Time	Time taken to send a request and receive a response	Milli second (ms)
Availability	Number of successful invocations/ total invocations	Percent (%)
Throughput	Amount of downloads for a given time period	Downloads/second
Reliability	Mean time to failure	Months
Successability	Number of responses / number of request messages	Percent (%)
Latency	Time taken for the server to process a given request	Milli second (ms)

4.5.2. Performance Comparison. To validate the prediction performance of the proposed PPSR, results obtained from the proposed method were compared with two other well-known hybrid recommendation methods, WSRec [11] and PHCF [4]. For this performance comparison, a single non-functional attribute was considered because the hybrid recommendation methods selected are limited to recommending services using a single non-functional attribute. In addition, to make the comparison unbiased, the

www.manaraa.com

satisfaction values of all the predicted non-functional attribute values from WSRec and PHCF were computed before the comparison. This is necessary because the focus of the proposed work is to show the importance of the satisfaction of user personalized preferences on service recommendation.

The Normalized Mean Absolute Error (NMAE), a well-known statistical accuracy metric, was used to measure the prediction accuracy. NMAE is the normalized average absolute deviation of predictions to the ground truth data. It is defined as:

$$NMAE = \frac{1}{n(r_{high} - r_{low})} \sum_{u,i} \left| p_{u,i} - r_{u,i} \right| \tag{32}$$

where r_{high} and r_{low} are the maximum and minimum satisfaction values in the system, respectively, $r_{u,i}$ and $p_{u,i}$ are the expected and predicted satisfaction values respectively. Smaller NMAE value indicates higher prediction quality.

Table 4.6 shows the comparison of PPSR to WSRec and PHCF on the response time non-functional attribute. The table shows that the method produces a smaller NMAE compared to the other methods for both 10% and 30% densities. Table 4.7 also shows the comparison of PPSR to WSRec and PHCF on the throughput non-functional attribute. The experimental results of Tables 4.6 and 4.7 show that using a δ value of 0.7:

- PPSR method obtains smaller NMAE values consistently, which indicates better prediction accuracy.
- The NMAE values of PPSR are independent from the given number unlike WSRec and PHCF. This shows that the proposed method doesn't suffer from PCC's inherent problem of its dependence on co-invoked services.
    - With the increase of the training matrix density from 10 to 30 percent, the prediction accuracy also achieve some enhancement, since denser training matrix provides more information for the prediction.

4.5.3. Impact of δ Value. Parameter delta (δ) makes the proposed method more feasible and adaptable to different datasets, especially those datasets where the active user has little or no previous experience with the services. It also allows the proposed CF-based personalized preference recommendation system to employ the advantages of the

www.manaraa.com

PCC extension of similarity computation as well as preference based similarity computation. For instance, in a situation where there is no or little co-invoked services, δ is set to 1 to alleviate the issues inherent to PCC based similarity functions.

To study the impact of the parameter δ to the proposed personalized preference collaborative filtering method, the Top-K value was set to 10 and vary the value of δ from 0 to 1 with a step value of 0.1. Figure 4.2 shows the results of given number = 10, 20, and 30 with 30% data matrix density.

Table 4.6. Comparison of PPSR to WSRec and PHCF on the Response Time Non-Functional Attribute

Density	10%			30%		
Given number	10	20	30	10	20	30
WSRec	0.5880	0.5512	0.5232	0.4585	0.4394	0.4001
PHCF	0.4814	0.4675	0.4478	0.3828	0.3642	0.3434
PPSR	**0.2962**	**0.2893**	**0.2911**	**0.2165**	**0.2117**	**0.2141**

Table 4.7. Comparison of PPSR to WSRec and PHCF on the Throughput Non-Functional Attribute

Density	10%			30%		
Given number	10	20	30	10	20	30
WSRec	0.8378	0.8071	0.7705	0.7281	0.7033	0.6620
PHCF	0.7444	0.7079	0.6874	0.6247	0.5882	0.5700
PPSR	**0.4806**	**0.4882**	**0.4827**	**0.4018**	**0.4025**	**0.4023**

Observing from Figure 4.2, it can be concluded that the value of δ impacts the recommendation results significantly, and a suitable δ value will provide better prediction accuracy. Another interesting observation is that, in Figure 4.2, with the given number increasing from 10 to 30, the optimal value of δ which obtains the minimal NMAE values of the curves in the figure, changes significantly. This indicates that the

www.manaraa.com

optimal δ value is influenced by the given number. For the current dataset, it was identified that the optimal value of δ is 0.7.

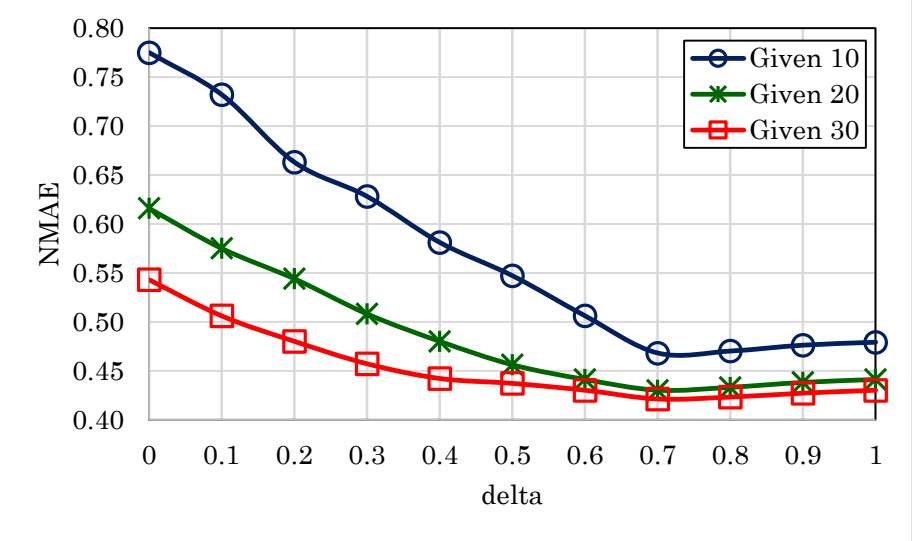

Figure 4.2. Impact of delta (δ)

4.6. CONCLUSIONS

In this section, an innovative method for service recommendation where the personalized preference of users are taken into consideration was presented. To accurately compute the similarity between users or service items, the proposed method extends the widely used Pearson Correlation Coefficient to include satisfaction of users' personalized preferences on non-functional attributes. Based on the similarity values, the top-k algorithm was employed to find similar neighbors. Finally, to predict missing non-functional attribute values, an extension of the weighted average with mean offset was employed to incorporate users' satisfaction on non-functional attributes based on their personalized preferences. Experimental results show that the approach significantly improves the prediction accuracy than the existing methods regardless of the sparseness of the dataset.

www.manaraa.com

BIBLIOGRAPHY

[1] K. K. Fletcher, X. F. Liu, and M. Tang. 2015 Elastic Personalized Non-Functional Attribute Preference and Trade-off based Service Selection. ACM Transactions on the Web. 9, 1, Article No. 1. DOI: http://doi.acm.org/10.1145/2697389.

[2] X. F. Liu, K. K. Fletcher, and M. Tang. 2012. Service Selection Based on Personalized Preference and Trade-Offs among QoS Factors and Price. In *Proceedings of the 1st International Conference on Services Economics (SE'12)*. IEEE Computer Society, Los Alamitos, CA, 32-39. DOI: 10.1109/SE.2012.5.

[3] M.D. Ekstrand, J. T. Riedl, and J. A. Konstan. 2011. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction* 4, 2, 81-173.

[4] Y. Jiang, J. Liu, M. Tang, and X. Liu. 2011. An Effective Web Service Recommendation Method Based on Personalized Collaborative Filtering. In *Proceedings of the 9th International Conference on Web Services (ICWS'11)*. IEEE, Los Alamitos, CA, 211-218. DOI: 10.1109/ICWS.2011.38.

[5] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. 2003. Quality driven web services composition. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*. ACM New York, NY, 411-421. DOI: http://doi.acm.org/10.1145/775152.775211.

[6] Y. Liu, A.H. Ngu, and L. Zeng. 2004. QoS Computation and Policing in Dynamic Web Service Selection. In *Proceedings of the 13th International World Wide Web (WWW) Conference*. ACM, New York, NY, 66-73. DOI: http://doi.acm.org/10.1145/1013367.1013379.

[7] E. Al-Masri and Q. H. Mahmoud. 2007. Discovering the best web service. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*. ACM, New York, NY, 1257-1258. DOI: http://doi.acm.org/10.1145/1242572.1242795.

[8] E. Al-Masri and Q. H. Mahmoud. 2007. QoS-based Discovery and Ranking of Web Services. In *Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN' 07)*. IEEE Computer Society, Los Alamitos, CA, 529-534. DOI:10.1109/ICCCN.2007.4317873.

www.manaraa.com

[9] Y. Zhang, Z. Zheng, and M.R. Lyu. 2010. WSExpress: A QoS-Aware Search Engine for Web Services. In *Proceedings of the 8th International Conference on Web Services (ICWS'10)*. IEEE Computer Society, Los Alamitos, CA, 91-98. DOI:10.1109/ICWS.2010.20.

[10] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun. 2013. Personalized QoS awareWeb service recommendation and visualization. *IEEE Transactions on Service Computing*. 6, 1, 35–47. DOI: 10.1109/TSC.2011.35.

[11] Z. Zheng, H. Ma, M.R. Lyu, and I. King. 2009. WSRec: A Collaborative Filtering Based Web Service Recommender System. In *Proceedings of the 7th International Conference on Web Services (ICWS'09)*. IEEE, Los Alamitos, CA, 437–444. DOI: 10.1109/ICWS.2009.30.

[12] H.-J. Zimmermann. 1991. *Fuzzy Set Theory and Its Applications*. Kluwer Academic, Boston, MA.

[13] S. Ran. 2003. A Model for Web Services Discovery with QoS. *ACM SIGecom Exchanges*. 4.1(2003), 1-10. DOI:http://doi.acm.org/10.1145/844357.844360.

[14] M. Comuzzi and B. Pernici. 2009. A Framework for QoS-Based Web Service Contracting. *ACM Transactions on the Web (TWEB)*. 3.3(2009), Article 10.

[15] K. Benouaret and D. Benslimane. 2012. WS-Sky: An Efficient and Flexible Framework for QoS-Aware Web Service Selection. In *Proceedings of the 19th International Conference on Web Services (ICWS'12)*. IEEE Computer Society, Los Alamitos, CA, 146-153. DOI:10.1109/SCC.2012.83.

[16] K. Benouaret, D. Sacharidis, D. Benslimane, and A. Hadjali. 2012. Majority-Rule-Based web service selection. In *Proceedings of the 13th international conference on Web Information Systems Engineering (WISE'12)*. Springer-Verlag, Berlin, Heidelberg, 689-695. DOI: 10.1007/978-3-642-35063-4_54.

[17] S. S. Yau and Y. Yin. 2011. QoS-based Service Ranking and Selection for Service-based Systems. In *Proceedings of the 8th International Conference on Services Computing (SCC'11)*. IEEE Computer Society, Los Alamitos, CA, 56-63. DOI:10.1109/SCC.2011.114.

www.manaraa.com

[18] H. Sun, Z. Zheng, J. Chen, and M. Lyu. 2011. Nrcf: A novel collaborative filtering method for service recommendation. In *Proceedings of the 9th International Conference on Web Services (ICWS'11)*. IEEE Computer Society, Los Alamitos, CA, 702–703. DOI: http://doi.ieeecomputersociety.org/ 10.1109/ICWS.2011.86.

[19] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun. 2013. Personalized QoS aware web service recommendation and visualization. *IEEE Transactions on Service Computing*. 6.1(2013), 35-47.

[20] H. Sun, Z. Zheng, J. Chen, and M.R. Lyu. 2013. Personalized Web Service Recommendation via Normal Recovery Collaborative Filtering. *IEEE Transactions on Service Computing*. 6.4(2013), 573 - 579. DOI: 10.1109/TSC.2012.31.

[21] K. T. Atanassov. 1986. Intuitionistic Fuzzy Sets. *Fuzzy Sets and Systems*. 20(1986), 87-96.

[22] L. W. Li, L.C. Chun, C.K. Ming, and Y. Muhammad. 2006. Fuzzy Consensus on QoS in Web Services Discovery. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06)*. IEEE Computer Society, Los Alamitos, CA, 791-798. DOI: 10.1109/AINA.2006.186.

[23] P. Wang. 2009. QoS-Aware Web Services Selection with Intuitionistic Fuzzy Set Under Consumer's Vague Perception. *Expert Systems with Applications*. 36.3(2009), Part 1, 4460-4466. DOI: 10.1016/j.eswa.2008.05.007.

[24] P. Wang., K.M. Chao, C.C. Lo. 2010. On Optimal Decision for QoS-Aware Composite Service Selection. *Expert Systems with Applications*. 37.1(2010), 440–449. DOI: 10.1016/j.eswa.2009.05.070.

[25] M. Xiuqin, S. Norrozila, and R. Mamta. 2011. QoS-Aware Web Services Selection with Interval-Valued Intuitionistic Fuzzy Soft Sets. In *Proceedings of the 2nd International Conference on Software Engineering and Computer Systems (ICSECS'11)*. Springer, Berlin, Heidelberg, 259-268. DOI:10.1007/978-3-642-22170-5_23.

[26] M. Almulla, K. Almatori, and H. Yahyaoui. 2011. A QoS-based Fuzzy Model for Ranking Real World Web Services. In *Proceedings of the 9th International Conference on Web Services (ICWS'11)*. IEEE Computer Society, Los Alamitos, CA, 203-210. DOI: 10.1109/ICWS.2011.43.

www.manaraa.com

[27] F. Li, Y. He, W. Hu, L. Wu, and P. Wen. 2011. Web service selection based on fuzzy QoS attributes. *Journal of Computational Information Systems*. 7.1 (2011), 198–205.

[28] X. F. Liu and J. Yen. 1996. An Analytic Framework for Specifying and Analyzing Imprecise Requirements. In *Proceedings of the 18th International conference on Software Engineering (ICSE'96)*. IEEE Computer Society, Washington, DC, 60-69.

[29] X. F. Liu, M. Azmoodeh, and N. Georgalas. 2007. Specification of non-functional Requirements for Contract Specification in the NGOSS Framework for Quality Management and Product Evaluation. In *Proceedings of the 5th International Workshop on Software Quality*. IEEE Computer Society, Los Alamitos, CA, 36–41. DOI:10.1109/WOSQ.2007.12.

[30] A. Sajjanhar, J. Hou, and Y. Zhang. 2004. Algorithm for web services matching. In *AProceedings of the 6th Asia-Pacific Web Conference (APWeb'04)*. Springer, Berlin Heidelberg, 665-670. DOI:10.1007/978-3-540-24655-8_72.

[31] S. Opricovic and G.H. Tzeng. 2003. Defuzzification within a Multicriteria Decision Model. *International Journal of Uncertainty, Fuzziness Knowledge-Based Systems.* 11.05(2003), 635–652. DOI:10.1142/S0218488503002387.

[32] B. D. Bowen and D. E. Headley. 2012. *Airline Quality Rating 2012*. Wichita State University. http://www.airlinequalityrating.com/reports/2012aqr.pdf. (Accessed July 2013).

[33] J. A McCall. 2002. Quality Factors. In *Encyclopedia of Software Engineering*. J. J. Marciniak (Ed.), John Wiley & Sons, New York, Vol. 2, 1083-1092. DOI:10.1002/0471028959.sof265.

[34] OpenFlights. 2013. OpenFlights.org. http:// www openflights.org/. (Accessed January 2013).

[35] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. 2006. Comparing Partial Rankings. *SIAM Journal on Discrete Mathematics*. 20.3(2006), 628-648. DOI: 10.1137/05063088X.

[36] A. Kopliku, K. Pinel-Sauvagnat, and M. Boughanem. 2014. Aggregated Search: A New Information Retrieval Paradigm. ACM Computing Surveys. 46.3(2014), Article 41. DOI: http://dx.doi.org/10.1145/2523817.

www.manaraa.com

[37] N. Ailon. 2010. Aggregation of Partial Rankings, p-Ratings and Top-m Lists. *Algorithmica*. 57.2(2010), 284-300. DOI: 10.1007/s00453-008-9211.

[38] R. Fagin, R. Kumar, and D. Sivakumar. 2003. Comparing Top k Lists. *SIAM Journal on Discrete Mathematics.* 17.1(2003), 134-160. DOI: 10.1137/S0895480102412856.

[39] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. 2004. Comparing and Aggregating Rankings with Ties. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '04)*. ACM, New York, NY, 47-58. DOI: 10.1145/1055558.1055568.

[40] F. J. Brandenburg, A. Gleißner, and A. Hofmeier. 2012. Comparing and Aggregating Partial Orders with Kendall Tau Distances. In *Proceedings of the 6th International Workshop on Algorithms and Computation (WALCOM '12)*. Springer, New York, NY, 88-99. DOI: 10.1007/978-3-642-28076-4_11.

[41] B. Hofreiter and S. Marchand-Maillet. 2013. Rank Aggregation for QoS-Aware Web Service Selection and Composition. In *Proceedings of the 6th International Conference on Service-Oriented Computing and Applications (SOCA'13)*. IEEE Computer Society, Los Alamitos, CA, 252-259. DOI: 10.1109/SOCA.2013.36.

[42] A. Van Deemen. 2014. On the Empirical Relevance of Condorcet's Paradox. *Public Choice*. 158.3.4(2014), 311-330. DOI: 10.1007/s11127-013-0133-3.

[43] C. F. Tsai, Y. H. Hu, and S. W. George Ke. 2014. A Borda Count Approach to Combine Subjective and Objective based MIS Journal Rankings. *Online Information Review*. 38.4 (2014). DOI: http://dx.doi.org/10.1108/OIR-11-2013-0253.

[44] G. V. Cormack, C. L. A. Clarke, and S. Büttcher. 2009. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09).* ACM, New York, NY, 758-759. DOI: http://doi.acm.org/10.1145/1571941.1572114.

[45] L. Baltrunas, T. Makcinskas, and F. Ricci. 2010. Group Recommendations with Rank Aggregation and Collaborative Filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, 119-126. DOI: http://doi.acm.org/10.1145/1864708.1864733.

www.manaraa.com

[46] T. Qin, X. Geng, and T. Y. Liu. 2010. A New Probabilistic Model for Rank Aggregation. *Advances in Neural Information Processing Systems*. 1948-1956.

[47] V. Pihur, S. Datta, and S. Datta. 2009. RankAggreg, an R Package for Weighted Rank Aggregation. *BMC Bioinformatics*.10.62(2009). DOI: 10.1186/1471-2105-10-62.

[48] F. Schalekamp and A. van Zuylen. 2009. Rank Aggregation: Together we're Strong. In *Proceedings of the 11th Workshop on Algorithm Engineering and Experiments (ALENEX'09)*. DOI: http://dx.doi.org/10.1137/1.9781611972894.4.

[49] V. Kolmogorov and V. Blossom. 2009. A New Implementation of a Minimum-Cost Perfect-Matching Algorithm. *Mathematical Programming Computation*. 1.1(2009), 43-67. DOI: 10.1007/s12532-009-0002-8.

[50] X. Chen, Z. Zheng, Q. Yu, and M.R. Lyu. 2014. Web Service Recommendation via Exploiting Location and QoS Information. *IEEE Transactions on Parallel and Distributed Systems*. 25, 7, 1913–1924. DOI: 10.1109/TPDS.2013.308.

[51] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie and H. Mei. 2007. Personalized QoS Prediction for Web Services via Collaborative Filtering. In *Proceedings of the 5th International Conference on Web Services (ICWS'05)*. IEEE, Los Alamitos, CA, 439–446. DOI: 10.1109/ICWS.2007.140.

[52] R.M. Sreenath and M.P. Singh. 2003. Agent-Based Service Selection. Journal of Web Semantics. 1, 3, 261–279. DOI:10.1016/j.websem.2003.11.006.

[53] W. Rong, K. Liu, and L. Liang. 2009. Personalized Web Service Ranking via User Group Combining Association Rule. In Proceedings of the 7th International Conference on Web Services (ICWS'09). IEEE, Los Alamitos, CA, 445–452. DOI: 10.1109/ICWS.2009.113.

[54] B.N. Miller, I. Albert, S.K. Lam, J.A. Konstan, and J. Riedl. 2003. MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. In Proceedings of the ACM International Conference on Intelligent User Interfaces. pp. 263-266, 2003.

[55] Ismat Beg and Samina Ashraf. 2009. Similarity Measures for Fuzzy Sets.

www.manaraa.com

[56] X. Dong, A.Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. 2004. Similarity search for web services. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04)*. Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer (Eds.), Vol. 30. VLDB Endowment, 372-383.

[57] C. L. Huang, K. M. Chao, and C. C. Lo. 2005. A Moderated Fuzzy Matchmaking for Web Services. In *Proceedings of the 5th International Conference on Computer and Information Technology (CIT'05).* IEEE Computer Society, Los Alamitos, CA, 1116- 1122. DOI: 10.1109/CIT.2005.21.

[58] S. Nepal, W. Sherchan, J. Hunklinger, and A. Bouguettaya. 2010. A Fuzzy Trust Management Framework for Service Web. In *Proceedings of the 8th International Conference on Web Services (ICWS'10*). IEEE, Computer Society, Los Alamitos, CA, 321-328. DOI:10.1109/ICWS.2010.52.

[59] QuickGraph. 2013. QuickGraph, Graph Data Structures and Algorithms for .NET. http://www.quickgraph.codeplex.com/. (Accessed January 2013).

[60] L. A. Zadeh. 1986. Test-score semantics as a basis for a computational approach to the representation of meaning. *Literary and Linguistic Computing*. 1.1(1986), 24-35. DOI:10.1093/llc/1.1.24.

[61] K. K. Fletcher, X. F. Liu, and M. Cheng. 2014. Aggregating Ranked Services for Selection. In *Proceedings of the 11th International Conference on Service Computing (SCC'14)*. IEEE Computer Society, Los Alamitos, CA, 331-338. DOI: 10.1109/SCC.2014.51.

[62] R.R. Liu, C.X. Jia, T. Zhou, D. Sun, and B.H. Wang. 2008. Personal recommendation via modified collaborative filtering. *Physica A: Statistical Mechanics and its Applications*. 388, 4, 462–468. DOI: 10.1016/j.physa.2008.10.010.

[63] M.R. McLaughlin and J.L. Herlocker. 2004. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. *In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '04)*. ACM, New York, NY, 329–336. DOI=10.1145/1008992.1009050.

www.manaraa.com

VITA

Kenneth Kofi Fletcher was born on September 7, 1984 in Tema, Ghana. He received his Bachelor of Science degree in Computer Science from the Kwame Nkrumah University of Science and Technology, Kumasi, Ghana in 2006. He was a graduate student in the Computer Science Department at Missouri University of Science and Technology beginning in August 2009. He worked as a Graduate Research Assistant under Dr. Xiaoqing (Frank) Liu from September 2010 to January 2011. He received his MS in Computer Science and a graduate certificate in software design and development at Missouri University of Science and Technology (formerly University of Missouri – Rolla) in December 2010, followed by his Ph.D. in Computer Science in December 2015.

www.manaraa.com