SNRNeg: A social network enabled negotiation service

Khayyam Hashmi a,*, Zaki Malik a, Erfan Najmi a, Abdelmounaam Rezgui b

a Department of Computer Science, Wayne State University, Detroit, MI 48202, United States
b Department of Computer Science and Engineering, New Mexico Tech, NM, United States

ARTICLE INFO

Article history:
Received 3 February 2015
Revised 2 February 2016
Accepted 8 February 2016
Available online 13 February 2016

Keywords:
Social networks
Negotiation

ABSTRACT

In recent years, the number of services and applications on the World Wide Web has increased exponentially. Consequently, a plethora of services providing similar functionalities are now available, which often poses a challenge for the users in terms of quality-based selection of their required services. Interaction with unknown services raises the issue of trust. Moreover, the service deliverables have to be negotiated and agreed upon, before any exchange can start. In this paper, we present a social network based trust framework (SNRNeg) that facilitates the negotiation of quality of service components. We extract recommendations from the social network (using the trust relationships between different nodes), and incorporate it into a decision model; which is used for negotiating the component Web service(s) of a composite system in an automated manner. Experiment results indicate the applicability and performance of SNRNeg in improving the negotiations for service composition process.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

In recent years, information system design has been influenced by the service oriented paradigm to facilitate easy integration between organizations that publish their services on the internet [2]. The ultimate goal is enabling the use of Web services as independent components in online enterprises that are automatically (i.e., without human intervention) formed as a result of consumer demand and which may dissolve post demand-completion [28].

With an increasing number of functionally equivalent services, it is now possible to combine several services to formulate a composite solution, where the clients have the option of selecting the most “suitable” service for their solution. Service composition is a multi-stage process that ranges from finding candidate services, determining their functional equivalence, negotiating customer and provider preferences, to finally creating an agreement. Therefore, on the service Web, the component services will have to automatically determine to which extent they may trust other services to provide the required functionality, before they interact with them. There could be scenarios where the Web services may fail to provide the promised services or meet their published values for different Quality of Service (QoS) values. This may affect the overall quality of the whole enterprise solution. Since interactions may occur between unknown entities, it is important to consider the trust values of these services [5,41] for the selection and composition process.

Currently, the services’ selection process is very tedious since it involves human intervention for negotiating customer and provider preferences. With the increasing agreement on the functional aspects of Web services (e.g., using WSDL [7] for
service description, SOAP [43] for communication etc., the research interest is shifting towards the non-functional aspects of Web services [33]. Since Web services likely span across multiple enterprise boundaries where different providers exercise control over their proprietary service(s), certain limitations are put on the different Quality of Service (QoS) attributes such as availability, reliability, scalability etc., of otherwise functionally equivalent services. Moreover, most of the quantitative attributes are not directly proportional in their cost/benefit curve (e.g., 99.999% uptime vs. 99.0% uptime). This non-linear curve naturally generates a disparity among the provided values for these QoS attributes and opens them to negotiation.

The negotiation process can be defined as a decision problem with multiple decision makers, and multiple (often conflicting) objectives. Selecting a Web service for automated composition, by generating a dynamic service level agreement (SLA), based on multiple objectives (e.g. QoS parameters) could be modeled as a constrained multi-objective problem. One of the important negotiation parameters is the confidence level of the negotiated service for providing its promised (published) quality. Determining this confidence level could be a tricky process that requires a lot of information and is mostly based on recommendations and trust between the negotiation participants. A primary source for such trust-related information is social media [12,36]. Recently, social media has enjoyed a great deal of success, with millions of users visiting sites like Facebook for social networking, Wordpress for blogging, Twitter for micro-blogging, Flickr for photo sharing, YouTube video sharing respectively, Digg for reading socially contributed news, and Delicious for bookmarking and sharing these bookmarks. These sites mainly operate on the content created and shared by their online users and their online relationships and provide personalized recommendations on this data. For instance, the goal of a personalized recommender system is to adapt the content delivery based on individual characteristics of the users. Since social media introduces new types of public data and metadata, such as tags, ratings, comments, and explicit people relationships, the idea is to use these pieces of information to enhance the quality of recommendations.

An automated negotiation mechanism consists of three main components: a high-level protocol, negotiation objectives, and decision strategies; while the negotiation context dictates the selection and integration of these components[16]. In existing literature, this has usually been accomplished in an ad-hoc manner [16,37], which is of minimal interest in SOAs due to the high developmental costs of such solutions, lack of ubiquity, and dynamic participants. A typical SOA-SLA negotiation involves multiple QoS attributes (e.g. reliability, availability, accessibility, response time etc.) [49], and there may be more than one combination of these attributes that may be suitable under a specified negotiation context. As mentioned above, since a services-based information system would be composed of more than one service(s), we may need to simultaneously negotiate multiple services. However, certain system properties are a composite function of its component services, e.g. the overall throughput of a system is limited by its component services having the least transaction per second. Hence, a negotiation system offering simultaneous negotiation of multiple services should have a mechanism to express these dependency relationships among the component services.

In this paper, we present an end-to-end solution for a negotiation Web service that could be used for negotiating component services and their associated qualities in a composite system. The main idea is to utilize the social network to gather information by using the trust relationships among the social network participants to filter the data (i.e., determine the trustworthiness of the claims published by individual services). The collected information is then leveraged to construct a decision model for conducting multi-objective and multi-agent negotiations of component services. Specifically, SNRNeg uses a social network based approach to first find a list of candidate services for the service selection process based on the recommendation and then ranks them based on the confidence in their ability to meet their proclaimed QoS parameters. It then uses a GA based approach for solving the Web service negotiation problem. The proposed framework takes into account the presence of malicious raters/recommenders that may exhibit oscillating honest and dishonest behaviors.

1.1. Motivation

In the following, we motivate the problem and our proposed approach with the help of a sample scenario. Consider a software engineer named John assigned the task of building a Web site that would enable a small travel scheduling and sales company to provide its services online. A primary objective is to provide the users with the ability to book their complete vacation online (including travel insurance). This potentially increases the chances of getting more business by providing its clients a single portal for their vacation planning. The proposed vacation package includes plane tickets, hotel reservation, car rentals, sight-seeing, etc. plus the travel insurance quote (e.g. services offered by sites such as Orbitz, Expedia, and Travelocity). John is a big fan of reusing off-the-shelf components to minimize the effort and time that is needed to implement a solution. He has been working with Web services for a while and feels comfortable implementing them as the building blocks of his new Web site. He also knows some companies provide a Web interface (e.g. WSDL) for developers so that their services could be used in a composite solution.

1.1.1. Current practice

John starts searching for potential approaches to discover Web services that he may be able to use in his composition. For example, he may look through online Web services registries using a key word based approach i.e. airline, hotel, etc. Assume that he locates few Web services that match his key-word based search criteria. Now he tries to go through these search results to see what these services do, and which of these could be composed to make a final system that meets his functional requirements. In this process, he may run into some unexpected issues. For instance, some of the services could be so old that either their implementation does not exist, or is not able to produce any output. Some of results
returned by the services may be outdated, or plain incorrect. Some of the services may take unexpectedly long time to return any meaningful results. Similarly, different services may return different results for the same query (same flight with different cost for the ticket). This means that John will have to go over these services, and then find out suitable services for his composition using a trial-and-error approach (hoping that he finds a good combination of services that provide cost effective results).

1.1.2. Proposed approach

John spends some time at the trial-and-error method and finally gives up on it (too laborious). Being a software engineer, he starts looking at the problem from different angles to see if he could do a better job in less time, and come up with a better solution. He starts thinking on how this could be achieved in a real-world scenario. His usual plan of action is to search online, read reviews on what he wants to buy, and then probably ask some of his friends that may have the knowledge about the product, or have bought similar products; gather all the information, and then make a decision on what to buy and what not to buy. Similarly, he can look around his social network (Facebook, Twitter, etc.) and figure out what types of services his friends are using e.g. hotwire for travel, hotels.com for lodging etc [14,30]. There are several methods of doing so based on his friends’ feedbacks he can get an idea on what services are currently being used by customers and how they rank in their value to the end consumer. He can then prioritize this information based on which of his friends have shared this information and can use factors like how close that friend is to him, what are his preferences, what is John’s experience with the friend’s previous recommendations, and incorporate these factors into his search criteria.

Since John is building a commercial system, he needs to work on other aspects of the solution such as Quality of Service (QoS) components; availability, reliability, throughput, etc. so that his customers can have a good experience using these services, and in turn be more inclined to buy his company’s travel insurance packages. Since most the services are paid, he would have to negotiate with the service providers on the cost of using them. The major negotiation factors could thus be the non-functional QoS components as shown in Fig. 1. All these components would thus reflect how his own solution is perceived by the end users. Some of these components among services are independent of each other (e.g. availability, and reliability), while some are dependent (e.g. throughput). Therefore, he will have to be careful in his service selection, to make sure that not only do they work well individually, but also be a good option for the composite solution.

The main contributions of the paper are as follows:

- We present a social network based recommendation approach that uses trust information among peers to construct a list of recommended Web services for negotiation.
- We present an end to end Genetic Algorithm based Web service negotiation approach for modeling dependency relationships among the QoS component of Web services.
- We compare our approach with similar techniques to show its applicability.

The rest of the paper is organized as follows. In Section 2, we discuss the recommendation system based on the social network. In Section 3, we present our negotiation methodology. Section 4 presents a brief literature review. In Section 5, we present the results of our experiments for the proposed approach and compare the performance of different components of the approach with similar works. Section 6 concludes the paper and outlines the directions for future work.
2. Social network recommendation

We can define a social network as a combination of nodes where each node represents an individual. When seeking the recommendation of a social network to buy a particular item (a Web service in our case), the node would query its neighbors for recommendation on that particular item. If the neighbors do not have any information about that particular item, they will pass the query to their neighbors. Hence the social network replies to the query of an individual node by offering a set of recommendations. There are multiple ways of using the newly acquired information. The easiest of all would be to use the most frequently recommended item. However, this may not be the best solution considering the heterogeneous preferences of each node and its interaction with that particular item. Table 1 lists the definition of symbols used henceforth.

Let us consider a set \( S_0 \) of \( N_0 \) nodes \( a_1, a_2, a_3, \ldots, a_n \). The idea is that nodes are connected to each other in a social network, such as, friend in a network on sites like Facebook, etc. that share their opinions and recommendations. Hence, each node will have a set of links to other nodes. Moreover, since networks evolve over time, i.e. people make new friends and breakups do happen, we assume that we work on a snapshot of the network, and the snapshot window is so small that these graphs could be treated as static network graphs. In this paper we model these networks on the random graphs. Although random graphs may not the best approximation of special structure of graphs i.e. for Facebook or Twitter, but using them makes our approach network independent. Let us have a set \( S_0 \) of \( N_0 \) object as \( o_1, o_2, o_3, \ldots, o_n \). These objects represent anything (products, service, etc.) that could have a rating. In our case, the focus is on Web services. We further assume that these objects are classified under one or more \( N_s \) categories from \( S_s \), denoted by \( c_1, c_2, c_3, \ldots, c_s \). In our scenario, it would be that Web services are categorized as “Travel Service” or “Hotel Reservation Services”. We denote the fact that an object \( o_i \) is in the category \( c_j \) by stating that \( o_i \in c_j \). Each node \( a_j \) is associated with one certain preference profile which is one of \( N_s \) preference profiles in the system, where \( S_p = p_1, p_2, p_3, \ldots, p_s \). Such a profile \( p_i \) is a mapping which associates each object \( o_j \in S_0 \) a particular corresponding rating \( r_{ij} \in [-1, 1] \).

Each node \( a_j \) keeps track of a trust value \( T_{a_j} \in [0, 1] \) to each of its neighboring node \( a_j \). The trust value between two nodes \( T_{a_j, a_k} \) is initialized to \( T_0 \), where this could either be as simple as 0.5 or we could come up with a complete bootstraping method (e.g. as defined in our previous work [25]). It is important to mention that direct trust relationships only exist between neighbors in the social network. However, two such nodes may indirectly be connected to each other through a path in the network. For example, node \( a_i \) could be connected to node \( a_j \) through node \( a_k \) assuming \( a_i \) and \( a_k \) are neighbors and nodes \( a_k \) and \( a_j \) are neighbors. Hence we can compute the trust path in the graph as shown in Eq. (1).

\[
T_{a_i, \ldots, a_j} = \prod_{(a_i, a_k) \in \text{path}(a_i, a_j)} T_{a_k, a_{k+1}} \tag{1}
\]

where the trust value along a path is the product of the trust values of the links on that path. There may be more than one path between two nodes; in such cases, each path has its own trust value. Fig. 2 illustrates a part of such a social network of nodes and a chain of trust relationships between two nodes \((a_i \text{ and } a_k) \text{ are connected with } T_{a_i, a_j}, T_{a_j, a_k}, \text{ and } T_{a_k, a_m})\).

There are two possible methods of searching for a recommendation. Ranking within a category (RWC): where a node queries for a particular category and then searches for several objects within that category to recommend a response, e.g. Travel Services in our scenario. Second Specific rating of an object (SRO): where a node would query the network for the recommendation about a specific object to determine the recommendation against it i.e. Expedia (a travel service). Both these variants are possible in our model. However RWC is best suited for our running example. At each time \( t \) each node \( a_i \) prepares the query for the selected category \( c_j \) and searches for recommendations. We can limit how many nodes it will traverse before returning the search using the concept of Time to Live (TTL). Specifically, the algorithm is as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_0 )</td>
<td>Set of nodes ( a_1, a_2, a_3, \ldots, a_n ), i.e. users in the network.</td>
</tr>
<tr>
<td>( S_o )</td>
<td>Set of objects ( o_1, o_2, o_3, \ldots, o_n ), i.e. objects that can have ratings.</td>
</tr>
<tr>
<td>( S_s )</td>
<td>Set of categories ( c_1, c_2, c_3, \ldots, c_s ).</td>
</tr>
<tr>
<td>( S_p )</td>
<td>Set of profiles ( p_1, p_2, p_3, \ldots, p_s ).</td>
</tr>
<tr>
<td>( r_{ij} )</td>
<td>Rating value of ( j )th component.</td>
</tr>
<tr>
<td>( T_{a_i, a_j} )</td>
<td>Trust value of the path between node ( a_i ) and ( a_j ).</td>
</tr>
<tr>
<td>( T_o )</td>
<td>Initial trust value of a node.</td>
</tr>
<tr>
<td>( R )</td>
<td>Set of all responses.</td>
</tr>
<tr>
<td>( \Gamma )</td>
<td>Parameter that controls the exploratory behavior of the node. Provider’s vector.</td>
</tr>
<tr>
<td>( \alpha_{a_i, a_j} )</td>
<td>Trust value for the path ((a_i, a_j)).</td>
</tr>
<tr>
<td>( \xi )</td>
<td>Similarity of two nodes.</td>
</tr>
</tbody>
</table>
Algorithm

1. Node \( a_i \) prepares a query\((a_i, c_j)\) for category \( c_j \) and then transmits it to its neighbors
2. Each neighbor \( a_k \) receives the query\((a_i, c_j)\) and EITHER
3. returns a response \((a_k, a_i, (o_j, r_j), T_{a_k}...a_k)\), it knows the rating \( r_j \) for a particular object \( o_j \) in \( c_j \)
4. \( p_k(o_j) = r_j > 0 \) if it was a positive rating
5. \( p_k(o_j) = r_j < 0 \) if it was a negative rating
6. OR passes the query\((a_i, c_j)\) to its own neighbor, if it does not know the rating \( r_j \) for the particular category \( c_j \)

End Algorithm

It is assumed that nodes keep track of the queries they have seen. There are two strategies to guarantee that the algorithm terminates: nodes do not process queries that they have already seen (incomplete search, IS); or, nodes pass on queries only once, but, if they have an appropriate recommendation, can return responses more than once (complete search, CS). In essence, both are a form of breadth-first search on the social network of nodes, but with different properties: the former returns, for each possible recommendation, only one possible path in the network from the querying to the responding node. The latter, however returns for each possible recommendation, each of the possible paths in the network from the querying to the responding node. The IS returns a recommendation along one of these paths, while the CS returns a set of recommendations along all possible paths. Some paths between two nodes have high trust, some have low trust. The IS may return a recommendation along a low-trust path even though there exists a high-trust path, thus providing a node with insufficient information for proper decision-making. Of course, there is also a pitfall with the CS is that it is computationally much more expensive.

As a result of a query, each node \( a_i \) possesses a set of responses from other nodes \( a_k \). It now faces the issue of making a decision for a particular object. The node needs to decide, based on the response of different nodes, what would be the appropriate choice of objects to recommend. We denote the query\((a_i, o_j)\) = \( Q \) and a response \((a_k, a_i, (o_j, r_j), T_{a_k...a_k}) \in R \) where \( R \) is the set of all responses. The values of trust along the path provide a ranking of the recommendations. There are many ways of choosing based on such rankings; one would be to sum all the negative and positive recommendations for the objects in the category and then use the one that is most recommended. However we would use an exploratory behavior of nodes and an established way of doing so consists in choosing randomly among all recommendations with probabilities assigned by a logarithmic function. First, we convert the recommendation into an intermediate variable \( \Gamma \) such that it lies in the range \([-\infty, \infty]\)

\[
\Gamma_{a_i...a_j} = \frac{1}{2} \ln \left( \frac{1 + 2(T_{a_i...a_j} - T_0)}{1 - 2(T_{a_i...a_j} - T_0)} \right) \in [-\infty, \infty]
\]  

This means that if the recommendation is 0 i.e. \( T_{a_i...a_j} = 0 \) then \( \Gamma_{a_i...a_j} = -\infty \) and similarly if the recommendation is 1 i.e. \( T_{a_i...a_j} = 1 \) then \( \Gamma_{a_i...a_j} = \infty \). We do need to take care of the negative recommendations separately in this part. That would
mean that
\[ \gamma = \exp(\Gamma_{a_{i-1}a_1}) \quad \text{and} \quad L(\text{response } (a_k, a_i, (a_j, r_j), T(a_{i-1}a_1))) = \frac{\gamma}{\sum_{k \in K} \gamma} \]
(3)
where \( \Gamma \) is the parameter that controls the exploratory behavior of the node. This makes it possible to have the trust value lie between \([0,1]\). For \( \Gamma = 0 \) each response will have equal probability i.e. random choice and for \( \Gamma > 0 \) responses with higher trust values will be selected. Now, suppose that a node receives a recommendation from another node, with multiple paths. For example, \( a_i \) may be linked to \( a_k \) through \( a_j \), but also through \( a_k \). Then, each of the two responses would be assigned a probability according to Eq. (2). Since recommendations coming along paths of high trust will have a higher probability of being chosen, this implies that recommendations coming along paths of low trust are still part of the decision making process, but with much lower probability [17].

In order for the nodes to learn from their experiences, it is necessary to incorporate the feedback of the recommendations. After an interaction, node \( a_i \) who has acted on a rating through its neighbor, node \( a_j \) (assuming we are using the probabilistic approach), updates the value of trust to this neighbor, based on the experience that he made. Let \( a_k \) be the chosen object. Then, assuming node \( a_i \) having profile \( p_i, p_j(a_k) = r_k \) is the experience that \( a_i \) has made by following the recommendation transmitted through \( a_j \). Now if \( r_k \geq 0 \) then
\[ \alpha_{a_i,a_j}(t + 1) = \beta \alpha_{a_i,a_j}(t) + (1 - \beta) r_k \]
(4)
and for \( r_k < 0 \) then
\[ \alpha_{a_i,a_j}(t + 1) = (1 - \beta) \alpha_{a_i,a_j}(t) + \beta r_k \]
(5)
where \( \alpha_{a_i,a_j}(0) = 0 \) and \( \beta \in [0,1] \), because \( \alpha_{a_i,a_j} \in [-1,1] \) we have to map it back to interval \([0,1]\) as follows:
\[ T_{a_i,a_j}(t + 1) = \frac{(1 - \alpha_{a_i,a_j}(t + 1))}{2} \in [0,1] \]
(6)
We prefer a slow positive and a fast negative trust propagation mechanism, i.e. it is hard to gain someone’s trust but is very easy to lose it. It is depicted by \( r_k \) for the values of \( \beta > 0.5 \) Hence we can use this to get a list of recommendations based on our social network. For our scenario this list denotes the Web services that our social network recommends. Since we considered all types of social networks to be used with this approach that means the recommendations received are the perceptions of users that have used this service with some bias already added to them. This in turn narrows down the search for the most useful Web services.

2.1. Evaluation

In order to analyze the performance of our approach we look at the utility gain of its participant nodes. When a node \( a_i \) follows a recommendation of node \( a_j \) at time \( t \) for object \( o_k \) then its utility can be defined as follows where node \( a_i \) determines \( p_i(o_k) = r_i \). We consider the performance of the system as the average utility gain of its nodes.
\[ u(a_i, t) = r_i \quad \text{and} \quad S_{\text{perf}} = \frac{1}{N_o} \sum_{a_i \in S_o} u(a_i, t) \]
(7)

Now we introduce the concept of similarity of profiles. In the limit of mean-field approximation we consider that the fix points of trust are an approximation of similarity, \( \xi \) of their profiles. We then calculated the critical threshold of social network density above which the node can receive meaningful recommendation on its queries. The similarity of two profiles is calculate as
\[ \xi(i,j) = \frac{1}{N_o} \sum_{o_k \in S_o} 1 - | p_i(o_k) - p_j(o_k) | \]
(8)

Now if two nodes recommend the same object then they are similar i.e. \( 0 < \xi < 1 \) and \( -1 < \xi < 0 \) if they disagree on recommendations. If we have only two nodes in the system \( a_1, a_2 \) and they are evenly distributed i.e. \( n_i = 1/2 \) then we can say that \( \xi = 0 \) and if we only have one node \( a_i = 1 \) then trivially \( \xi = 1 \). Consider the trust update of one node towards the other, let us assume that we have two trust profiles \( p_m, p_n \) (profiles are different than nodes). Consider the recommendation made by \( a_j \) to \( a_i \) of a random object then in mean-field approximation we can replace \( r_k \) with its average, which be definition is \( \xi_m,n \). By the time node \( a_i \) develops high trust value towards \( a_j \) then \( a_j \) would have done the same for its neighbors. So if \( \xi_m,n \) is high then only the recommendations from neighbors \( a_w \) of \( a_i \) are associated with the recommendation along the path \( a_w, a_i, a_j \). Hence by induction we can extend this mean-Field approximation to general case of recommendation to \( a_i \) indirectly from \( a_j \).

At each time \( t \) the nodes receive results of queries and their trust ratings from its neighbors. Let \( R \) be the set of all the response \( k \) that a node receives over time. Hence the expected value of rating \( r \), hence the utility \( u \) is given by
\[ u = \sum_{k \in R} r_k P_k = \frac{\sum_{k \in R} r_k \exp(\beta \Gamma_k)}{\sum_{k \in R} \exp(\beta \Gamma_k)} \]
(9)
Since there are many ratings received by various profiles of different nodes so we can replace the node $q_j$ with the profile $p_q$, through following recommendations of other node with profile $p_i$. Similarly we can approximate the trust value with the $\xi$. From prior discussion we know that $\exp(\beta^t)$ can be approximated with the value $(1 + \xi)/ (1 - \xi)\beta/2$. This approximation is well justified for the immediate neighbors and although less accurate but still holds true for others since we assume that the social network is dense and well connected. So the expected utility of the node coincides with system performance and could be calculated as

$$S_{\text{perf}} = \frac{\sum_{i \in S_y} \xi^2 \frac{(1 + \xi S_y)}{1 - \xi S_y} \beta/2}{\sum_{i \in S_y} \left(\frac{(1 + \xi S_y)}{1 - \xi S_y}\right) \beta/2}$$ (10)

If we group the set $S_y$ by the similarity values between profiles of query and recommendation node, since there are only finite number of such pairs hence their probability depends on the relative frequency of these profiles. Hence, the probability of occupation of each similarity value $\xi, \nu(\xi)$ is known by construction. Hence we can rewrite the system utility as

$$S_{\text{perf}} = \frac{\sum_{i} \xi \left(\frac{1 + \xi}{1 - \xi}\right)^{\beta/2} \nu(\xi)}{\sum_{i} \xi \left(\frac{1 + \xi}{1 - \xi}\right)^{\beta/2} \nu(\xi)}$$ (11)

The above equation holds true in the social network setting where the nodes are well connected and cover the categories for recommendation. For instance if we consider two profiles $p_1$ and $p_2$ with the frequency $n_1$ and $1 - n_1$, then the probability that a pair of nodes consists of both $p_1$ or both $p_2$ or mixed is $(n_1)^2, (n_2)^2 = (1 - n_1)^2$ and $2(n_1)(1 - n_1)$ respectively and the corresponding $\xi$ values are 1,1,-1. In the absence of trust the $S_{\text{perf}} = 4(n_1)^2 - 4(2n_1) + 1$ and in the presence of trust the term with $\xi$ close to 1 dominates and hence $S_{\text{perf}}$ reaches 1.

3. SNRNeg negotiation methodology

SNRNeg uses a weighted sum genetic algorithm to support multi-party multi-objective negotiation. Once we have the trust values for the Web services, our next step is to setup the automated negotiation process to formulate a composite solution. We use the trust values to enhance the negotiation process to formulate the composite solution. All the Web services provide their respective QoS parameters to be negotiated. These are called the component vector of a Web service. Each vector is accompanied by a decision model, i.e. ranges of all the QoS parameters as well as their respective priorities also known as the weights. We assume that all the participating Web services are able to articulate their objectives and prioritize them [1]. Table 2 lists the definition of symbols used henceforth.

Since all the Web services (participants) start negotiation from a different position, they have different preferences for those objectives, and are described by how far their current position is from the customer’s objective. All the Web services conform to some constraints in the solution. For instance, any QoS vector cannot have a negative value (as shown by Eq. (12)). The QoS values lie between the maximum and minimum allowable values set by the Web service (as shown by

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_i$</td>
<td>Fitness of the solution for participant $j$.</td>
</tr>
<tr>
<td>$F_s$</td>
<td>Fitness of the solution $s$ (for all participants).</td>
</tr>
<tr>
<td>$c_j$</td>
<td>The value of $j$th component of Consumer’s vector.</td>
</tr>
<tr>
<td>$\zeta_{\text{min}}$</td>
<td>The minimum allowed value of $j$th component of Consumer’s vector.</td>
</tr>
<tr>
<td>$\zeta_{\text{max}}$</td>
<td>The maximum allowed value of $j$th component of Consumer’s vector.</td>
</tr>
<tr>
<td>$W_{ij}$</td>
<td>The weight of $j$th component of Consumer’s vector.</td>
</tr>
<tr>
<td>$p_j$</td>
<td>The value of $j$th component of $i$th Provider’s vector.</td>
</tr>
<tr>
<td>$\nu_{\text{inj}}$</td>
<td>The minimum allowed value of $j$th component of $i$th Provider’s vector.</td>
</tr>
<tr>
<td>$\nu_{\text{inj}}$</td>
<td>The maximum allowed value of $j$th component of $i$th Provider’s vector.</td>
</tr>
<tr>
<td>$W_{Pj}$</td>
<td>The weight of $j$th component of $i$th Provider.</td>
</tr>
<tr>
<td>$r_j$</td>
<td>Rank for solution $j$ in the system.</td>
</tr>
<tr>
<td>$n_i$</td>
<td>Value of Norm $i$ in the system.</td>
</tr>
<tr>
<td>$\nu_i$</td>
<td>The willingness of participant $j$ to exchange objective $i$.</td>
</tr>
<tr>
<td>$\nu_i$</td>
<td>Amount of resource $i$ exchanged by Web service $j$.</td>
</tr>
<tr>
<td>$G$</td>
<td>Total number of generations.</td>
</tr>
<tr>
<td>$\nu_{\text{inj}}$</td>
<td>The value of generation when interval window is calculated.</td>
</tr>
<tr>
<td>$\nu_{\text{inj}}$</td>
<td>The value of sliding window.</td>
</tr>
<tr>
<td>$\nu_{\text{inj}}$</td>
<td>The initial service selection vector.</td>
</tr>
<tr>
<td>$P_{\text{prev}}$</td>
<td>Fitness values of the $j$th participant in the previous generation.</td>
</tr>
<tr>
<td>$D_j$</td>
<td>Difference among the $i$th QoS attribute value and $i$th Norm value for $j$th participant.</td>
</tr>
<tr>
<td>$\nu_{\text{inj}}$</td>
<td>Value of dependent $i$th Norm of the component $C$ of the system.</td>
</tr>
</tbody>
</table>
Eq. (13)). A repair algorithm is applied to GA after each operator, to ensure all these constraints are met.

\[
C_j \in \mathbb{Q}_{\geq 0}, \quad P_j \in \mathbb{Q}_{\geq 0},
\]

\[
C_{j\text{(min)}} \leq C_j \leq C_{j\text{(max)}} \text{ and } P_{j\text{(max)}} \leq P_j \leq P_{j\text{(min)}}
\]  

Each chromosome is a combination of customer and provider genes. If we have \( n \) QoS parameters to be negotiated then each chromosome will have \( 2n \) genes. The fitness function is a multi-step calculation that evaluates the level of disagreement between the negotiating Web services. A weighted sum approach is used to combine these multiple QoS parameters. The weight of the service depends upon its reputation. We assign a higher weight to more reputable service. Hence the higher reputed service will have a greater chance of being selected since the customer will be confident that the published QoS values would be met. We use a distance function to measure the difference among the proposed solutions of both the customer and provider Web services. Thus, lower fitness values are desired as they translate to lesser disagreement among the participants. Similarly, lower values translate to higher ranks for the solutions among the solution space. Ranks are then used for selection of subsequent steps of the GA [4,47]. Each solution represents a probable distribution of values that may be agreed upon by the other Web service in the negotiation.

The fitness value of a solution is calculated as follows.

\[
f_i = \left( \sum_{k=0}^{n} (WC_k \ast \Delta_{ik} + WP_{ik} \ast \Delta_{ik}) \right) \ast \text{Reputation}(s_j) \text{ and } \Delta_{ij} = \left| \frac{C_j - P_j}{C_j} \right|
\]

\[
F_s = \min_{0 \leq i \leq G} (f_i)
\]

Where \( f_i \) is the fitness of the solution \( s \) for participant \( j \) and \( F_s \) is the fitness of the solution \( s \) (for all participants). \( WC_j \) represents the weight of \( j \)th component of customer’s vector as provided by the customer and \( WP_{ij} \) shows the weight of \( j \)th component of \( i \)th provider’s vector as provided by the provider. \( G \) represents the total number of generations.

3.1. Norm operator

We use the Norm operator for improving the performance of GA. Assume we have \( n \) norms (information sources) in the society and \( k \) population subsets. Set 1 may follow Norm 1, Set 2 may follow Norm n and Set \( m \) may choose to follow Norm 2 while others may not choose to follow any Norm. The selection of subsets and Norm selections are random. Population (Web services) in Set 1 is affected by the values of Norm 1 and they in turn affect the values of the Norm. This cycle makes sure that beneficial values are prevailed in the Norms. Each QoS negotiation criteria is represented as a norm and certain members of the population follow a certain norm. After each generation, the followers update the impact factor of their respective norm. If increasing the value of the norm resulted in a better overall fitness value for the member of population, it would influence the norm into increasing its value. The increase is dependent on the difference of current and previous values of that objective of the reporting individual and the current absolute value of that objective. Both customers and providers share the same influence values of norms. This is an indirect information source for the customer about providers decision model and vice versa. Ideally, we will have one customer and \( n \) providers, hence sharing these impact factors does not reveal any trade secrets. These values have the bias of \( n+1 \) agents and are averaged out.

Norm is implemented for the exchange of resources among different participants. Exchange must occur between two distinct objectives, participants can trade some or all of their available objectives and there is at most one exchange per pair per generation. Exchange is implemented probabilistically. Each member of population is reviewed for possible exchange. The participants and objectives involved in the exchange are selected randomly. Then it is decided if an exchange will actually occur based on the willingness of participants. The exchange only occurs if both randomly selected participants are willing to make an exchange. Essentially, willingness to exchange is higher if a participant has more of an objective than he ideally wants and if the information source that he is following is influencing a lower value of that specific objective. If the current Web service is following Norm\(_m\) then the willingness to exchange is \( E_{ij} \) and the amount exchanged \( A_{ij} \) is calculated as

\[
E_{ij} = \left| \frac{C_i}{N_m} \right| \cdot A_{ij} = (1 - WC_j) \left| 1 - \frac{C_i}{N_m} \right|
\]

If the current Web service is not following any Norm then the willingness to exchange and the amount to be exchanged is calculated as

\[
E_{ij} = \left| \frac{C_i}{P_j} \right| \cdot A_{ij} = (1 - WC_j) \left| 1 - \frac{C_i}{P_j} \right|
\]

where \( N_i \) is the value of Norm \( i \) in the system. \( E_{ij} \) shows the willingness of participant \( j \) to exchange objective \( i \) and \( A_{ij} \) is the amount of resource \( i \) exchanged by Web service \( j \). Norm operator holds the cumulative knowledge of the entire system. It is used to share the private knowledge without revealing the personally identifiable information. In the beginning norms
are populated with intelligently guessed values and member of population are randomly assigned to different norms. After each iteration every member of the population assess its performance and provides feedback to the norm being followed. When progressing from generation $i$ to $i+1$, following Norm$_m$ it will be calculated as follows.

$$A_{ij} = (1 - W C_j) \left[ 1 - \frac{C_i}{P_{ij}} \right] \text{ and } c = f_j - \text{Pre} f_j$$  \hspace{1cm} (18)

where Pre$f_j$ is the fitness value of the $j$th participant in the previous generation. If $c > 0$ there is an increase in the overall fitness of that member of the population hence what ever he did is something good and should be shared with other participants.

If the current member of population does not follow any Norm, we do not do anything and move on to the next member. If it follows a certain Norm we have to update the cumulative knowledge value of that Norm. Let us assume that the member $j$ follows Norm $i$ and the $i$ component of the vector was changed. Then since it is a positive change we will add some value to the Norm indicating that a higher value is suited assuming that the current $i$ value is higher than the Norm value. If

$$P_{ij} > N_i \text{ and } D_{ij} = P_{ij} - N_i$$  \hspace{1cm} (19)

where $D$ will give us the difference in the value of the current Norm and the member’s corresponding QoS attribute value.

$$\Delta N = \left( \frac{R_j}{j} \right) \ast \left( \frac{D_{ij}}{P_{ij}} \right) \text{ and } N_i = N_i + k \ast \Delta N$$  \hspace{1cm} (20)

In the above equations $R_j$ is the rank for solution $j$ in the system. Where $k$ is learning factor and represents how much weightage is given to the history vs. to the current value and $D_{ij}$ is the distance among the $i$th QoS attribute value and $i$th Norm value for $j$th participant. In our experiments we found out that one tenth is good value for $k$. Similarly, if a negative change in the value of the any vector results in better fitness value we augment the Norm value accordingly. Moreover, if a member is constantly improving its fitness value by following a NormM it will tend to keep following it. On the other hand a decrease in the fitness value of member over a period of time will increase its chances of leaving that Norm. Where $k$ represents the last $k$ generations of this member.

$$P_{\text{switch}} = 1 - \left( \frac{k}{k} \sum_{h=1}^{k} \left( \frac{P_{ij} - N_i}{P_{ij}} \right) \right)^2$$  \hspace{1cm} (21)

3.2. Norm dependency modeling in SNRNeg

We extend the Norm operator to enable the searching of multiple services in parallel, for them to be part of the same composition. Let us say that Service C provides car insurance. To provide car insurance quotes, C needs to gather two pieces of information i.e. the driving history and the credit score of the applicant. For simplicity, lets assume that C negotiated contracts with Service A for driving history and Service B for credit score. Among other QoS parameters, Transactions Per Seconds (TPS) was one of the negotiated parameters and A’s negotiated contract included the proposition of providing 100 TPS. However, while negotiating with B, C found that the maximum TPS offered by B is 75 TPS (that suits C’s budget). This entails that the maximum throughput of the composition cannot be more than 75 TPS. The scenario would be same if we assume that we negotiate Service B first and then negotiate with Service A (which comes more or less as a chicken and egg problem). One solution would be to create some temporary contract and then re-negotiate a new contract based on all ‘found’ parameters. However, this may not be feasible in all scenarios. Moreover, since we know that we are trying to negotiate with multiple services, if we could articulate the dependencies of QoS parameters to the negotiating service, we may find a better solution.

In SNRNeg, we use a sliding window approach to minimize the discrepancy among the dependent QoS parameters. The idea behind the approach is that since all negotiation participants optimize their QoS parameter values based on the corresponding Norm’s value, we restrict the progress of corresponding Norm value for the dependent QoS parameter among all Web services (TPS for Service A and Service B). This ensures that the difference in the negotiated values is not greater than a predefined threshold of $\Delta V$. In this regards, we introduce the concept of a sliding window as the interval that allows the Norm values for dependent QoS parameters to learn from the population space just like the Norm values for independent QoS parameters. At the end of the sliding window interval the Norm value for the dependent QoS parameters is restricted. The sliding window interval is calculated dynamically (based on the level of discrepancy among the Norm values and participating services’ preferred values) so the disparity among the initial offers provided by different participants is neutralized. To dynamically calculate the sliding window we need to first have an initial point when we can calculate the pace of learning for the Norm values i.e. Interval Check point. Then we calculate the size of the sliding window (in term of number of generations) and at the end calculate the generation number when we restrict the Norm values. Then initial check point is represented by $G_{i\text{Check}}$.

$$G_i = G_{i\text{Check}} \text{ and } \Delta_{\text{Interval}} = \left( \frac{U_1 - U_2}{3} \right) + \left( \frac{C_i + P_j}{3} \right) + \Delta N_i$$  \hspace{1cm} (22)
The sliding window size \( S_{\text{Window}} \) is calculated as follows

\[
S_{\text{Window}} = \frac{G}{\Delta_{\text{Interval}}}
\]

(23)

At every sliding window distance we restrict the corresponding Norm values of the dependent QoS parameters. Where \( V_i \) is the value of dependent ith Norm in the system.

\[
V_i = V_j = \min(V_i, V_j) + \frac{\min(V_i, V_j)}{100 + \Delta_{\text{Interval}}}
\]

(24)

This ensures that the Norm values for the dependent QoS attributes closely follow each other and leads the system to a solution where the negotiated QoS values for the dependent parameters do not have a big disparity among them. This leads to an overall better solution.

4. Literature review

4.1. Social network based recommendation systems

Social networks are portals that allow users to connect with each other and share personal or professional information. The main idea is to create an interaction among different users for sharing information and contents. They could be termed as virtual communities for disseminating information. Recommendation systems combine the ideas from user profiling, information filtering, and machine learning to deliver users a more intelligent and customized information service by making product/service recommendations that match user preferences and needs. Recommendation systems can utilize the information present in social network to deliver a better personalized recommendation experience.

In literature there are two most commonly used approaches for building recommendation systems: first content-based approaches (CB) [29] and second collaborative filtering based approaches (CF) [13]. The CB approach is based on recommending items that are similar to those in which the user has shown interest in the past. The CF approach, on the other hand, recommends items to the user based on other individuals who are found to have similar preferences or tastes. Several studies have suggested incorporating direct social relationships in CF systems. ReferralWeb [19] was one of the first systems to suggest the combination of direct social relations and CF to enhance searching for documents and people. Several studies suggest incorporating explicit social network information in CF systems to improve the quality of recommendation in domains such as movies and books (e.g., [6,11]), music [20] and news stories [21]. On the other hand, as tagging has emerged as a popular way to let users annotate social media content, several works propose using tags as content descriptors for CB systems.

Traditional recommender systems purely mine the user-item rating matrix for making recommendations. However, recommendations are not made in rational isolation, which means that they are not evaluated merely by their information value [35]. The social embedding of a recommendation is crucial to understanding the decision making process of an individual; it is determined by factors such as experience, background, knowledge level, beliefs and personal preferences [24]. It has been found [40] that given a choice between recommendations from friends and recommender systems, usually, friends’ recommendations are preferred even though the recommendations given by the recommender systems may be better. People typically trust and act on recommendations from friends more than from the company selling the product. Positive word of mouth recommendations [38] among customers is by far the best predictor of a company’s growth. In general, a user is much more likely to believe statements from a trusted acquaintance than from a stranger. However, current recommendation techniques make recommendations to a target user mainly based on other users’ item preference, these users have similar rating data with the target user, but the trust between users has not been well exploited.

PowerTrust [50] is a distributed version of EigenTrust [18]. It states that the relationship between users and feedbacks on eBay follow a Power-law distribution. In PowerTrust, nodes rate each interaction and compute local trust values. These values are then aggregated to evaluate global trust through random walks in the system. Once power nodes are identified, these are used in a subsequent look-ahead random walk that is based on Markov chain to update the global trust values. Power nodes are used to assess the reputation of providers in a system-wide absolute manner. This is in contrast with our approach where each consumer maintains control over the aggregation of ratings to define a providers reputation. Moreover, PowerTrust requires a structured overlay (for DHT), and the algorithms are dependent on this architecture. In contrast, service-oriented environments or the Web in general do not exhibit such structure.

In [8] authors propose a random walk based algorithm to compute the recommendations based on the trust relationship among the social network. This approach is similar in nature to our work as they also look at the trust relationship as a non-binary relationship and look at the degree of trust. However, the proposed approach has some issues. They use public ratings as the source of recommendation and the random walk algorithm that strictly focuses on the trust paths and hence does not use the complete information present to us in the social network. Due to the sparse nature of recommendations in a big social network it is likely that recommendation information may not be available from the most trust worthy node. Hence our approach is more complete in that sense that it utilizes all the pieces of information to infer a better recommendation. Secondly, authors do not have a feedback mechanism to trust propagation. This is crucial in any recommendation system so that long term interactions would result in much higher accuracy of recommendations.
In [45] authors propose a rank preserving analysis approach in sensor networks. They introduce a rank order information that uses the dimension reduction and utilizes the within-class and between-class information. They introduce a penalized factor distance that takes into account the concentration of measure phenomenon and preserves most of the local patch information formed within-class rank orders. This results in being able to extract local discriminative information while ignoring the rank order information. Although it performs very well in the proposed situation however it still results in a reduced dimension of network and varies from the original distribution. Our proposed problem relies heavily on the trust paths and implicitly on the rank information. A more trusted node will have more influence on the final recommendation result. Since most of the recommendations will come from the neighboring nodes where the distance is relatively small this dimension reduction could potentially loose an important node. Hence, adversely effecting the results. However, this methodology will be faster in case the distance of the recommending nodes is large. Similarly in [44] authors propose a Hessian regularized support vector machine to extract the local geometry of a data fold and successfully apply to label large data sets. However, it requires a large number of training samples and considerable amount of computing and storage resources to achieve high performance. In our proposed solution we approximate the graph structure by using more traditional approaches that require no training data set and very less computing resources. Hence, making it faster and more suitable to our specific problem.

4.2. Negotiation

In SLA negotiations, each participant has multiple objectives, and adheres to multiple constraints, that bind those objectives. The multi-objective, multi-party nature of negotiation suggests that such problems are quite complex, both in terms of formulating a feasible solution and the time it takes to get to that solution. Automated approaches such as machine learning, evolutionary computations etc. have shown promising results for solving complex multi-objective multi-person negotiations. In particular, genetic algorithms (GAs) have had a lot of success in solving such optimization problems. GAs are capable of finding solutions for complex problems that are hard to solve using more traditional approaches [3,42]. There have also been some efforts of employing evolutionary computational techniques for automated negotiations of Web services. For instance, a mediation service based mechanism for generation of automated contracts using some flavor of the GA based operators such as mutation and crossover is proposed in [15]. Similarly, in [22] an agent-based GA negotiation system for B2B eComerce is proposed. Both these models assume bilateral negotiations and keep them independent of any information that may be obtained or effect the decision making process of any agent. However in our approach we use the information gathered during the negotiation process to guide the system towards a mutually agreeable solution. In [48], an approach for bilateral negotiation under uncertainty, where a negotiator is uncertain as to what offer or counteroffer to make, at a particular step in the negotiation is presented. This uncertainty is resolved by making use of the negotiation experience of reputable parties. This approach relies heavily on the negotiation experience of reputable external parties for expressing negotiation offers, which is mostly not the case in Web service composition. A model for bilateral negotiations that consider the uncertain and dynamic outside options is discussed in [23]. Outside options affect the negotiation strategies via their impact on the reservation price. This model is targeted for long running negotiations which span over some amount of time and we may expect new offers during that time, which were not available in the beginning of the negotiation process. Service composition requests do not fall in such a category of negotiations.

5. Performance study

To determine the efficiency of our approach we performed several different classes of experiments: First, we calculate the effectiveness of our social media based recommendation approach using trust paths. Second, we show the effectiveness of our reputation assessment approach by comparing the perceived and actual reputation among services. Third, we show how the GA based negotiation approach performs. Fourth, we show the Norm dependency modeling using a multi-agent and multi-attribute scenario.

The experiment environment consists of a Windows server 2008 (SP2)-based Quad core machine with 8.0 GB of ram. We developed 50 provider Web services running on Microsoft.Net version 3.5 to simulate multi-party negotiations. A large number of similar providers are chosen to show the applicability/scalability of the proposed solution. The Web services were measured on four QoS components (reliability, availability, throughput and accessibility). We use the widely [26,27,34,46] used real world SNAP [30] Facebook data set containing 4000 nodes and 88,000 edges to show the social network with pseudo-random assignment of trust values for each link. We run 1000 transactions where each transaction is performed in one time unit and then the results are fed back into the system. We average out our results for 15 rounds. The GA based negotiation process was tested over 200 iterations consisting of 500 generations each.

Fig. 3 A shows the comparison between original provider performance and the assessed reputation for providers that perform consistently. It can be seen that due to the high number of honest ratings, the assessed reputations are almost equal to the original provider performance. The small variation in assessed and original reputations is due to the inconsistency brought in by the (honest) differences in opinions of credible raters and malicious attempts of non-credible raters. Similarly we can see that in Fig. 3B our algorithm performs well for services that may have an oscillating performance behavior.

In the next experiment we evaluate the utility of the system. Over time, each node develops a value of trust towards its neighbors which reflects the similarity of their respective profiles. After some time, paths of high trust develop, connecting
As a result, the performance of the system improves over time and reaches a stationary value which approaches the optimum value, as shown in Fig. 4, where the curves correspond to different values of $\beta$. Increasing values of $\beta$ lead to curves approaching the optimum faster. We can see that the number of categories affect the speed of convergence of the system.

Once we have the reputation values for the candidate services we use them for negotiating a SLA based on QoS values. Fig. 5A shows the learning graph of Norm for one sample run of such negotiation scenario. We can see that the Norm values for Throughput, Reliability and Availability stabilize fairly quickly but the Norm value for Response Time stabilizes around the 100th generation. Correspondingly, our technique converges to the solution around the 100th generation in this particular run. Then we compare SNRNeg with similar approaches presented in the literature. We base our comparison on the utility values of these techniques and the time it takes to reach a solution. SBA [31] uses a GA based approach with an offer and counter-offer based protocol for searching a mutually agreeable solution. The degree of overlap among the QoS values requested by the customer and those offered by the provider are taken into account in SBA. We use the results for the maximum overlap (80%) in our comparisons. Similarly, NBA [32] uses a GA based approach with a very similar fitness function as used in our technique, but does not take into consideration any other parameters. SWC [10] also uses a GA based nodes with similar profiles.
approach for the semantic composition of Web services. It uses the semantic equivalence in addition to the QoS values to determine the best offering for the composition. We compare the results of SWC with 20 services (SNRNeg had 50 providers). BLGAN [39] uses a Bayesian learning based approach with GA and incomplete information model to learn the reserve price of its opponent. GTFSN [9] presents a game theoretical model of signaling games for Service Level Agreement negotiation. The results are presented in Fig. 5B.

We can see that SNRNeg is the quickest in improving on the initial solution. This can be attributed to the use of Norm, and the solution improves exponentially as Norm values stabilize (high jumps around generation number 27 and 60). We can also see that SNRNeg finds a solution within the 99% utility range in about 66 generations, while SWC (the second best) takes about 3 times the time. The other two techniques fail to generate such a solution and plateau around the 95% range. The results suggest that our approach outperforms other compared methods both in terms of finding the optimal solution and the amount of time it takes to find that solution.

Finally we show the results of SNRNeg’s dependency modeling approach. We use SNRNeg to negotiate two services (Service A and Service B) where, the negotiation vector consists of four QoS attributes < Throughput, Availability, Reliability, Response...
Time -, we assume that Throughput is the dependent QoS parameter among Service A and Service B. Fig. 6A and C show the values for customer offers when Service A and Service B are negotiated separately. The negotiated vector for Service A is (95,95,95,90) and Service B is (98,95,95,90). We can see that the dependent attribute of Throughput has a value of 95 TPS in Service A reached around generation number 83. In Service B the value of 98 TPS is reached around generation number 110. The overall output of the composed system will thus have a Throughput value of 95 TPS i.e. minimum of the TPS values for the component service. This could only be observed when we have all TPS values from every component service in the system.

When we run the same service negotiation scenario with dependency modeling, simultaneously negotiating for Service A and Service B we can see (in Fig. 6B and D) that the negotiated vector for Service A is (95,95,95,90) and Service B is (95,96,96,91). We get the TPS of 95 for Service A around generation number 200 and the same TPS value of 95 for Service B around generation number 260. So the composite system will have the overall TPS value of 95. We can clearly see that though the TPS value is the same but we get a better overall solution for the system i.e. Service B now has higher negotiated values for Availability, Reliability and Response Time. Since we restrict the value of the dependent QoS attribute to a lower value, we are able to get higher value for the remaining attributes for Service B. This increases the overall utility of the system and shows the practicality of our proposed technique.

6. Conclusion and future directions

We have presented SNRNeg, a framework for Web service negotiation using social networks to enable customers and providers establish SLAs (using the trust relationships established on the social Web). It utilizes the reputation of a Web service to enhance the effectiveness of the negotiation process in a multi-party and multi-attribute negotiation scenario. It exploits the fact that we tend to trust our friends, rather than strangers when it comes to a positive or negative recommendation about an object. Moreover, the strength of a social connection dictates the degree of influence of the recommendation, on the decision making process. We leverage this information to narrow down the candidate search for service composition. Then we use a GA based approach to incorporate the reputation of a Web service into the decision making process for service negotiation. We have enhanced the traditional GA with a new operator called Norm, that presents the cumulative knowledge of the community over a period of time. This accumulated knowledge influences the decision making process of negotiating participants. Experiment results show that our proposed approach facilitates the negotiation process and improves the performance in comparison with similar techniques. We further extended our approach to incorporate dependency modeling for different QoS parameters among multiple services to formulate an optimized solution.

A limitation of our technique lies in the fact that it treats all the social networks as a random graph. Though this assumption makes the technique generic, it also means that we are not utilizing the sparse nature of the social network graph, and are not exploiting the most common patterns. Second, our system assumes that these graphs are static in nature (we work over a temporal snapshot). In reality these graphs are very dynamic and hence would require a much more agile approach of dealing with them. Third, the negotiation service assumes a static environment, where the Web service procurement time window is so small that user preferences do not change during the course of negotiations.

We are currently working on identifying the patterns found on the social Web to make the recommendation process faster and more dynamic. We are investigating on enhancing the effectiveness of private information sharing by exploring the possibilities of having consumers follow multiple information sources (Norms) rather than following just one source. This is motivated by the fact that composite solutions often have dependent objectives. We need to be able to use the information sources of the Norm operator to share such information, and be able to pass on all dependency constraints and decision models to SNRNeg. We are looking into enhancing the recommendation process by considering the different behavioral factors of the social Web. Moreover, we are also looking into how to distribute (propagate) the rating of a multi-node path recommendation.

References
