2013 IEEE 6th International Conference on Service-Oriented Computing and Applications

A Framework for Automated Service Negotiation

Khayyam Hashmi, Erfan Najmi and Zaki Malik
Department of Computer Science
Wayne State University
Detroit, Michigan, USA
Email: {khayyam, erfan, zaki}@wayne.edu

Abdelmounaam Rezgui
Department of Computer Science
Engineering, New Mexico Tech
NM, USA
Email: rezgui@cs.nmt.edu

***Abstract*—**

Web services composition enables the business to dynamically and seamlessly integrate business applications on the web. The performance of an overall composition is a function of its individual component services. Hence, both functional and non-functional properties of Web services are important when negotiating component service for a system. Automated negotiation helps speed up this process and often improves the overall quality of the composite system.

In this paper, we present a negotiation Web service that would be used by both the businesses searching for a Web service as well as the providers of Web services for conducting negotiations for dependent QoS parameters. We present the detailed architecture and system engineering view of our framework and use a genetic algorithm (GA) based approach for finding acceptable solutions in multi-party and multi-objective scenarios. Experimental results indicate the applicability and improved performance of our approach in facilitating the negotiations involved in a Web service composition process.

I. Introduction

With the extensive adaptation of Web service based applications in dynamic businesses applications including on-demand computing, highly configureable virtual solutions and cloud computing based systems demand automated tools for composing and managing these services in composite systems. With an increasing number of functionally equivalent services, it is now possible to combine several services to formulate a composite solution, where the businesses have the option of selecting the most "suitable" service for their solution. Service selection (thereby composition) is a multi-stage process that ranges from finding functional equivalence, negotiating customer and provider preferences, to finally creating an agreement. Currently, this 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 [4] for service description, SOAP [40] for communication, etc.), and approaches being proposed to facilitate the on-demand composition of component services for formulating highly focused solutions, the research interest is shifting towards the non-functional aspects of Web services [29].

Negotiations is an important form of decision making process of different aspects of human life, such as business, scientific, social and political interactions etc [15] [25]. It is the process of back and forth communication in order to reach a mutually agreeable solution. In SOA the negotiation process can be defined as a decision problem with multiple decision makers, and multiple (probably conflicting) objectives. The aim is to simultaneously optimize multiple objectives, considering the constraints of both the service providers and consumers. In existing literature, this has usually been accomplished in an ad-hoc manner [14] [31], which is of minimal interest in SOAs due to the high developmental costs of such solutions, lack of ubiquity, and dynamic participants. An automated negotiation mechanism consists of three main components, namely, a high-level protocol, negotiation objectives, and decision strategies; while the negotiation context dictates the selection and integration of these components [14]. A comprehensive negotiation system should allow its user to specify multiple attributes for negotiations. It should consider the fact that multiple services would potentially be using heterogeneous protocols which may or may not be compatible. Similarly not every participant would be using the same decision model and hence, may require different decision making capabilities. It should also be able to simultaneously negotiate with multiple services, taking into account the dynamic nature of its participants requirements.

Solving multi-attribute optimization problems is an evolving effort in computer science, statistics, economics, and mathematics circles. To date, many powerful deterministic and stochastic techniques for addressing these large-dimensional optimization problems have been introduced. "Evolutionary algorithms are one such generic stochastic approach that has proven to be successful and widely applicable in solving both single-attribute and multi-attribute problems" [5]. In negotiation scenarios where the goal is to generate co-allocation offers as optimal as possible so that interaction between the requester and provider is minimized, resource utilization and provider profit is maximized, are NP-complete [36] [34] [6] [16]. Moreover, a number of natural variations of this problem are NP-hard, and determining whether a particular allocation is Pareto Optimal is co-NP-complete[8] . This is closer to the "winner determination problem" class of algorithms (that are also NP-complete), where approximation mechanisms such as evolutionary algorithms are shown to be highly effective [36] [26] [33] [32].

Genetic Algorithms (GA) have been used to enhance automated negotiation. GAs are used to evolve the best strategies [22], generate proposals at every round [37] [38], track

978-1-4799-2701-2/13 $31.00 © 2013 IEEE
DOI 10.1109/SOCA.2013.27

shifting tactics and changing behaviors [22][25], and learn effective rules for supporting negotiation [23]. GAs have been successful in solving such optimization problems as GAs are capable of finding solutions to complex problems that are hard to solve using more traditional approaches [24][39][1]. Similarly, constraints on these problems have been handled by penalty functions [9][19], repair algorithms [10], [35], special genetic operators to keep solutions in the feasible region [30][45]. There have also been some efforts for 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 [12]. Similarly, in [17] an agent-based GA negotiation system for B2B eCommerce 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. A GA based approach that uses Bayesian learning to predict the preferences of the negotiation participants based on their offers and counter offers have been presented in [11]. In [44], 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.

In this paper we present a framework for a negotiation service that is designed for a business environment with simultaneous negotiations among multiple service provider businesses, where each communication instance (among the consumer and service provider business) is private. We enhance the traditional Genetic Algorithm to make it possible enable all the negotiating agents to adapt quickly, and significantly reduces the search space by guiding the negotiation process towards a mutually agreeable and beneficial solution.

The paper is organized as follows: in Section 2 we present our framework for the negotiation service. In Section 3, we discuss our proposed automated negotiation approach. In Section 4, we present the experiment details, while Section 5 concludes the paper and outlines the directions for future work.

II. A FRAMEWORK FOR WEB SERVICE NEGOTIATION

A business looking for a Web service can benefit from a service that publishes (e.g. as UDDI $tModels$ [7])both functional and non-functional requirements(e.g. availability 99.99%, reliability 99.9%) and then simultaneously negotiate these requirements with multiple providers to obtain an optimal solution.

Figure 1 depicts the state diagram of an automated negotiation process. Typically, negotiation starts when the customer requests for proposals for a component service. After receiving the initial offers from service providers, it would then select some providers to engage in bi-lateral negotiations. This would start a round of offers and counter offers among the customer and selected service providers. Once both the provider and

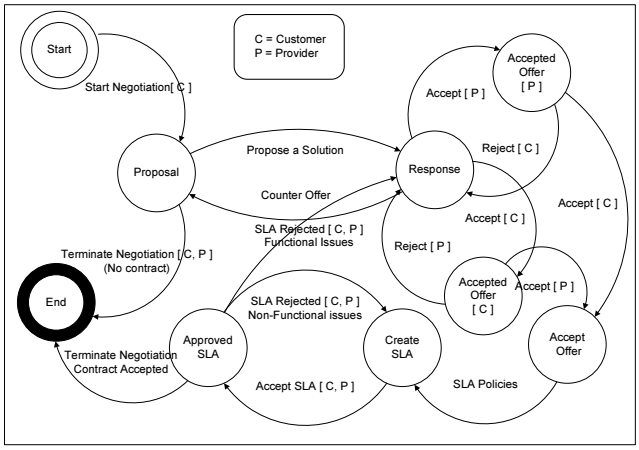

Fig. 1. Negotiation State Chart

the customer agree on certain attributes (e.g. price) for the services to be provided, they enter into the formal Service Level Agreement (SLA) formation phase. At this point, both parties agree on the terms and conditions of the agreement. These usually include both the functional (service to be provided, cost, etc) and non-functional (QoS parameters, violation terms, penalties, etc). This process could also be modeled as the exchange of offers and counter offers (for SLA terms) among the customer and the selected provider. Once agreed, the parties create/contract an agreement and the services are rendered. If both the parties could not agree on the terms of an SLA the current negotiation session is terminated and a new round of negotiation is started.

In this paper we present the high level architecture of our proposed negotiation service i.e. *NegF* in Figure 2 that is based on the negotiation flow presented above . The proposed model is very flexible in terms of its functionality and the services provided. It is primarily targeted to be invoked by a business searching among a list of service providers providing similar functionalities. The client does not need to implement any negotiation specific component to use the proposed service. The architecture presented in Figure 2 is compatible with both the negotiation scenarios i.e., the negotiating participants could either provide their own negotiation component or send all the necessary information to the service, that would handle all the negotiation process. A brief overview of the major modules of the proposed negotiation architecture is as follows:

A. Negotiation Service

The negotiation service acts as the interface of the whole system. This component is responsible for all external system communication. The internal components use this service to communicate with both the customer and provider business as well as with other peers within the community (to be discussed later). A business invokes the service providing its negotiation attributes (e.g. availability, reliability), negotiation policy as

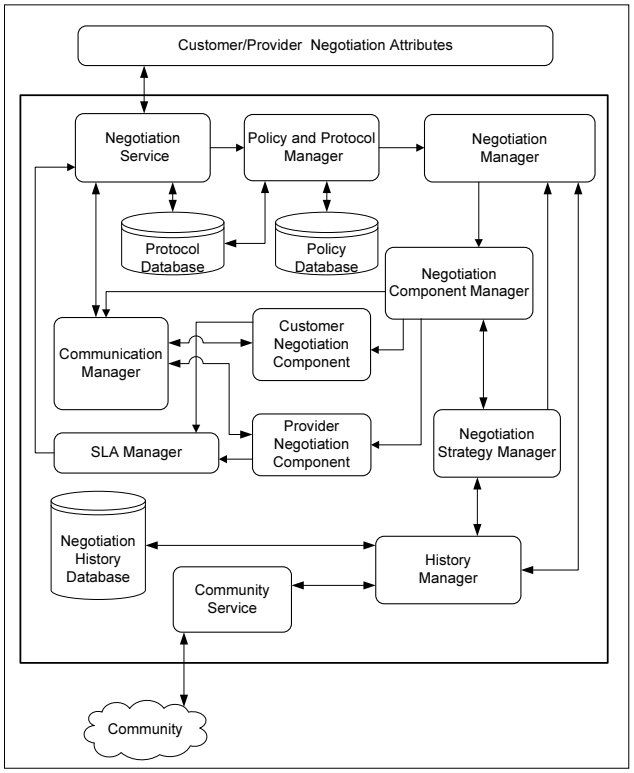

Fig. 2. NegF System Architecture

well as its decision model. The negotiation service then communicates with potential providers and requests their decision model and policy attributes for the negotiation process.

B. Policy and Protocol Manager

This component is responsible for standardizing the inputs from the communicating participants. Different participants may use different protocols for describing their decision models and policy attributes. A generic component would ensure that these heterogeneous participants could communicate with the negotiation service. After receiving this data from the negotiation service this component then translates it into a standard form which is used for the internal information exchange among different components of the system. It then stores these participant communication preferences in the Policy and Protocol database. Negotiation service would then use this information for any future communication with the participants. This generic module would mean that the system is extendable to incorporate future communication protocol and allow the customer and providers using different communication protocols to still be able to negotiate service attributes and form service level agreements (SLA).

C. Negotiation Manager

Figure 3 show the architecture of the Negotiation Manager. Once the service receives a request for negotiation from the customer along with all the necessary data, it then proceeds to the negotiation step. Negotiation manager would then query the Web service directory e.g. UDDI to search for the matching service providers. The customer also has the option of providing its own list of possible providers. Once it has the list of service providers providing similar services, it ranks these providers based on trust ratings.

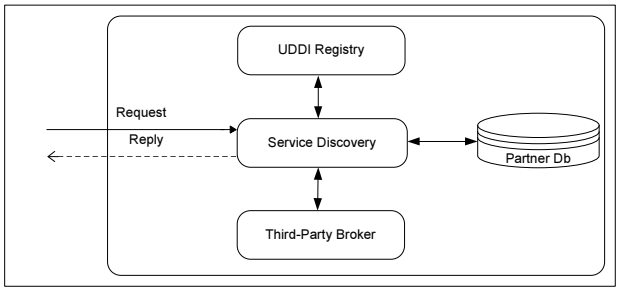

Fig. 3. Negotiation Manager

It uses the trust model based on the concept of community [20] where reputation represents the perception of the users in the community regarding a service provider. For a newly starting service that does not have any history, it uses the reputation bootstrapping mechanism defined in [21]. Community service is a knowledge base that is responsible for information regarding different providers, including reputation, trust and past negotiations. The community ensures that no private information is released to its users but could publish non identifiable data e.g. It does not give out any information about systems that are using lets say *ServiceA*, but could tell the total number of the systems currently running *ServiceA*. These pieces of information combined with the above mentioned methods of trust and reputation assessment, help the negotiation manger in selecting appropriate services, from a number of services providing similar functionalities. We assume that all these services are functionally equivalent.

D. Negotiation Component Manager

Since negotiation is a multi-party mechanism, the *NegF* system needs to spawn separate components for each customer and provider. In the most basic scenario at any given instance, the systems would have one customer, and multiple provider components. These components operate in their separate context and communicate with their original service through the communication manager. The negotiation manager is responsible for creating and managing these components. The architecture is flexible enough to let the participants provide their own negotiation components. We may have this scenario when the participants believe that they do not want to share any of their private information with even a trusted third party broker. This allows the flexibility of implementing a distributed negotiation system.

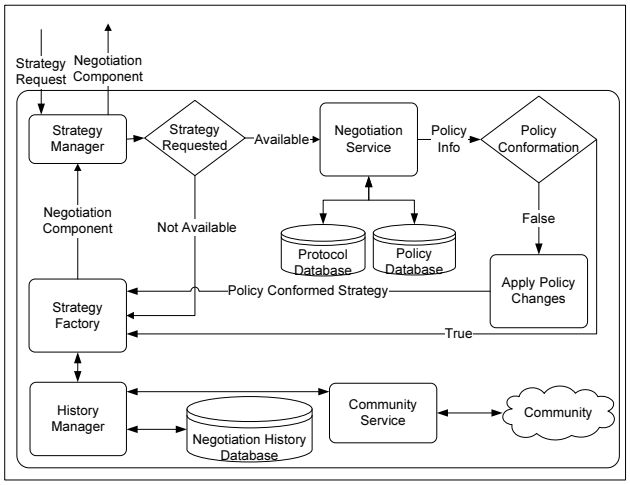

Fig. 4. Negotiation Strategy Manager

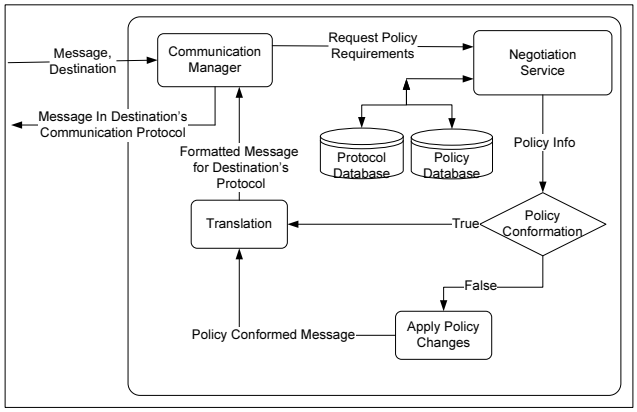

Fig. 5. Communication Manager

E. Negotiation Strategy Manager

There are multiple strategies available for conducting efficient negotiations. The *NegF* system architecture does not restrict the components to any one negotiation strategy. It has multiple strategies for the components to choose from. Participants could opt for using any strategy and pass this information as a policy to the system. Figure 4 depicts the architecture of the strategy manager.

If none is chosen, the system selects one or a combination of strategies for the negotiation process. The negotiation strategy manager selects and binds each component with the appropriate strategy and is responsible for implementing the component policies and decision model in the context of the selected strategy as well as monitoring and storing any transient data related to the negotiation process.

F. Communication Manager

All the external pre-contract communications are handled by this manager. The component may communicate with their respective services for any decision model or policy/guidance queries. Communication manager ensures that all the communication is related to the current negotiation and adheres to the negotiation service's policies. Figure 5 shows the architecture of communication manager. On receiving a communication message the components request policy info from policy and protocol databases and checks the conformity of the message with the retrieved info. If the message does not conform to the policies, it goes through a policy conformation process. After conforming with the policies the message goes through any translations if needed before being delivered to its destination.

G. Contract Manager

Once the system identifies a probable negotiation solution(s), it is presented to the respective services, and if they agree, the contract manager then handles the formal SLA creation process. The architecture is shown in Figure 6. Similar to the other modules, any SLA generation must conform to the system policies. Upon receiving a proposed solution, them module performs SLA specific policy conformance checks and prepares a formal SLA. This SLA is then presented to the negotiation participants. If the current selected provider does not agree on the proposed SLA, the system would then try the next best available solution, until either an agreement is achieved or the system has ran out of options. If the system could not find a mutually agreeable solution among all the available providers, then the process would be termed as a failure and the customer would be asked to revise its negotiation model.

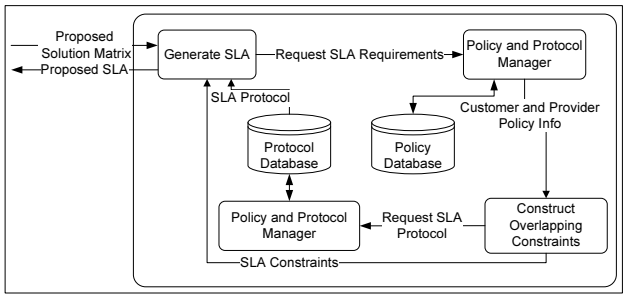

Fig. 6. Contract Manager

H. System Engineering View

The development view of the architecture focuses on the organization of different software modules. A part of this view is presented as the architecture of the software framework that serves as a guideline for the developers. Figure 7 provides the data model used in the implementation of *NegF*. This data model specifies the underlying concepts used by the systems such as service description, service attributes, communication

protocol, policy, status messages and negotiation strategy used for guiding the negotiation process towards a mutually agreeable solution. Figure 8 shows different interfaces used by the *NegF* system to communicate with different components within the systems as well as with other systems. Negotiation messages are composed of URL that identify the sender, a negotiationInfo that specifies the proposal and the state of the sender, which is governed by the INegotiationContext that maintains the negotiation context and the transient information. The interface IServiceSelection is used to get the initial set of services that are further contacted for the negotiation process. ISystemHandler will take care of the communication protocol conversion and maintains the overall state of the system including the SLA generation.

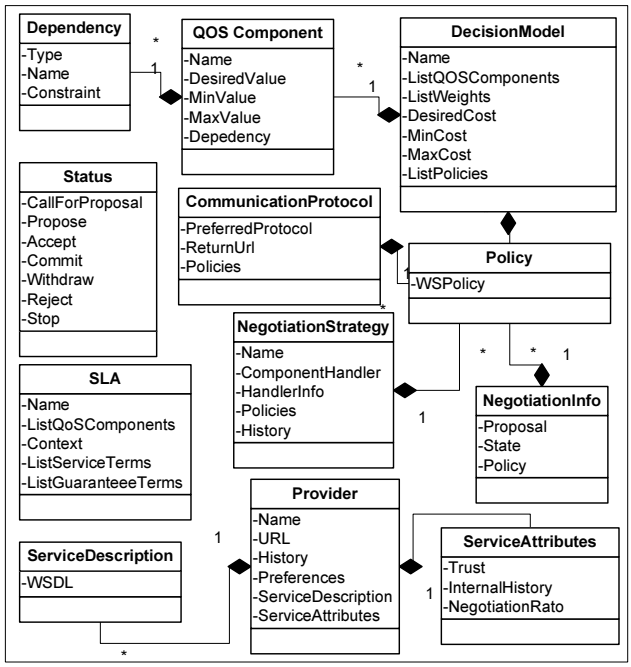

Fig. 7. *NegF* Data Model

III. Negotiation Methodology

In this paper we focus on the implementation of a negotiation algorithm that uses our proposed framework to construct a composite solution. We use a NSGA-2 based approach to support multi-party multi-objective negotiation. All the Web services provide their respective QoS parameters to be negotiated. These are called the component vectors 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. G represents the total number of generations. We present a brief overview of proposed approach below.

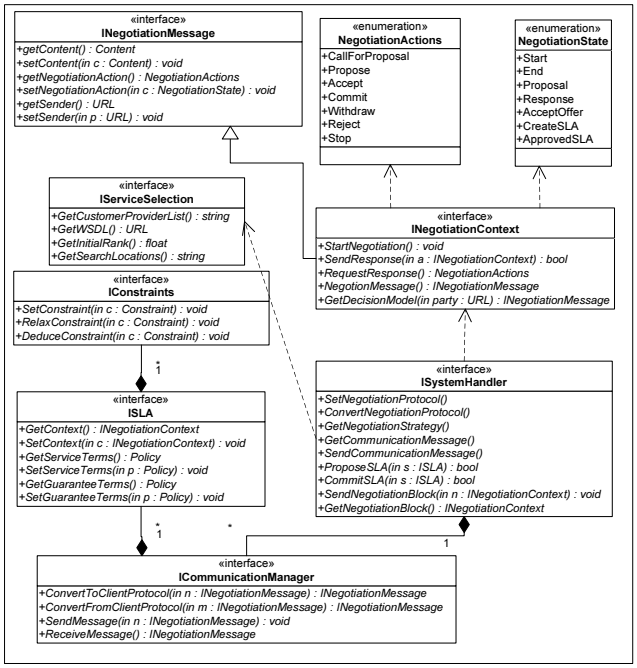

Fig. 8. System Interface Diagram

Algorithm 1 NSGA-2

1: **INPUT:** Initial offer vectors of consumer and producer services
2: **OUTPUT:** Negotiated component vectors of services.
3: Set generation number g equal to zero ($g = 0$)
4: Generate initial population
5: **for** each generation **do**
6: Evaluate objective values
7: Perform fast nondominated sort
8: Generate child population using GA operators (Tournament selection, crossover and mutation)
9: Evaluate objective values of child population
10: Perform Elitism
11: Combine parent and child population
12: Perform fast nondominated sort
13: Create next generation based on rank and crowding distance
14: $g = g + 1$
15: **if** $g == G$ (last generation) **then**
16: Ensure Pareto optimality
17: exit
18: **end if**
19: **end for**

A. NSGA-2

Initially a random parent population g_0 is created. The population space is sorted based on nondomination. Then each solution is assigned a fitness rank based on the its nondomination level. Then the new generation is created using the tournament selection, crossover and mutation. Elitism is introduced by comparing current population with the previously found best nondominated solution. We then combine the parent and child generation and sort them based on nondomination. Once

all solution slots are filled by the lowest ranked dominated solution (i.e. the best solution will have the lowest rank in the sorted list). We repeat this until we reach our desired number of generations.

Each chromosome is a combination of customer and provider genes. If we have n objectives 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. 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. Each solution represents a probable distribution of values that may be agreed upon by the other Web service in the negotiation. Pareto optimality is not enforced after each generation as it is possible for a Web service to accept a less favorable solution for the time being (in the negotiation process) for a better solution in the long run. However, a secondary population of solutions is kept which is updated after each iteration. This secondary population or *Elitism* is a an important concept in genetic searches [47][2]. The probabilistic nature of GA does not guarantee that the best solutions would be preserved in the final generation. Hence a secondary population of best solutions is kept through all generations.

B. Crowding Distance

The crowding distance variable guides the selection of process towards a uniformly spread out Pareto-optimal front. Assume that every gene i in the chromosome has the following attributes:

- nondominant rank (C_{rank})
- crowding distance ($C_{distance}$)
- bookKeeping distance ($C_{bookdistance}$)

Hence the crowding distance will be calculated as

Algorithm 2 Crowding Distance

```
1: INPUT: Solutions in a non-dominated set I
2: OUTPUT: Crowding distance of the solutions.
3: Set n = number of solutions in the set
4: for each i do
5:    Set Idistance[i] = 0
6: end for
7: for each QoS value m do
8:    I = sort (I,m)
9:    Idistance[1] = Idistance[n] = maxdistance
10:   for i = 2 to (n -1) do
11:      Idistance[i] = Idistance + (I[i+1].m - I[i-1].m) / (f(m,max)
         - f(m,min))
12:   end for
13: end for
```

Here $maxdistance$ refers to the maximum distance between two points and is set to infinity. $sort(I, m)$ refers to sorting the set I of solutions on the basis of objective m. $I[i].m$ refers to the mth objective function value of the ith individual in the set I. The parameters $f(m, max)$ and $f(m, min)$ represent the max and min values of the mth objective function.

Now the partial order for the crowding operator will be calculated as:

$$
\begin{aligned}
i \;&<\; j \\
&\quad if(rank[i] < rank[j]) \;\; or \\
&\quad ((rank[i] = rank[j]) \;\; and \;\; (distance[i] > distance[j]))
\end{aligned}
$$

C. Crossover and Mutation

The crossover operator is invoked after applying the non-dominated sorting. Roulette-wheel selection is used for selecting solution pairs for crossover. Roulette-wheel selection is analogous to a roulette wheel where the probability an individual is selected is proportional to its fitness [3]. Solution rankings are used to implement selection. The population is augmented so that solutions with better ranks are more prevalent in the population. We use both ranks and fitness values for our selection technique because ranking indicates the performance of solutions relative to others in the population and minimizes the effect of large disparities in fitness values within the population [41]. Augmentation of the population for roulette-wheel selection is performed as follows:

$$CrossP_j = 1 - \frac{1}{R_j}(R_j - 1) \tag{1}$$

Where $CrossP_j$ is the Cross over probability for service j and R_j is the rank for solution j in the system. Crossover rate is used to determine if crossover will actually occur or if the selected solution will simply be copied over to the next generation. If it is determined that crossover will occur, uniform crossover is implemented on the pair. It has been proved that custom operators provide superior performance for real-valued problems [43]. Mutation is the last operator to act on the population of solutions and is also applied randomly to the elements of the solution, in accordance with the experimentally predetermined mutation rate. A random number is generated for each member in the population and compared to the mutation rate. If the random number is less than or equal to the mutation rate, mutation will occur in that solution. Mutation here involves arbitrarily changing one element of the negotiation vector and then implementing a repair algorithm to ensure that objective values lies within the valid range for that chromosome.

IV. Results

To determine the efficiency of our approach, we conducted experiments on our prototype *NegF*. The experiment environment consists of a Windows server 2008 (SP2)-based Quad core machine with 8.0 GB of ram. We developed 1 client and 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 client negotiated four QoS components (reliability, availability, throughput and accessibility) with the providers, over 200 iterations consisting

of 150 generations each. WSDream-QoSDataset [46] is used, which contains more than 150 Web services distributed in computer nodes located all over the world (i.e., distributed in 22 different countries). Planet-Lab is employed for monitoring the Web services. We take the published services and their corresponding QoS values and write our own wrappers that incorporate the *NegF*'s negotiation component.

A. Experiment 1.

We use our service to negotiate 3 services (*Service A*, *Service B*, *Service C*) where, the negotiation vector consists of four QoS attributes < Throughput, Availability, Reliability, Response Time >. Note that a larger number of services produces similar results (three services are chosen only for illustration). We assume that *Throughput* is the dependent QoS parameter among all the services.

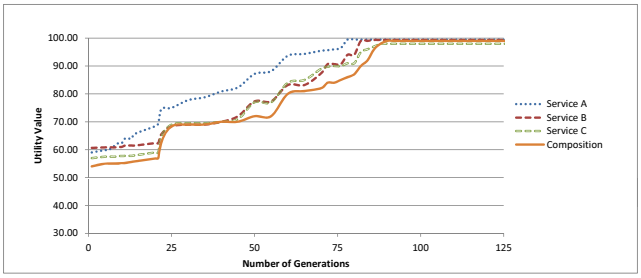

Fig. 9. Experiment results of service negotiation

Figure 9 shows the results of our experiments. Figure 9 shows the utility values of individual service and the composite solution, when the services are negotiated separately from each other. The negotiated vector for *Service A* is <97,98,98.5,99>, *Service B* is <97.9,98,97,98>, *Service C* is <98,97,99,98> with the utility values of 98.83, 99.00 and 98.75 respectively. Hence the system achieves a maximum utility value of 98.75.

B. Experiment 2.

In this experiment we compare *NegF* 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 [27] 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 [28] 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 [18] also uses a GA based 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. The results are presented in Figure 10. We can see that *NegF* is the quickest in improving on the initial solution.We can also see that *NegF* finds a solution within the 99% utility range in about 100 generations, while SWC (the second best) take about 1.5 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.

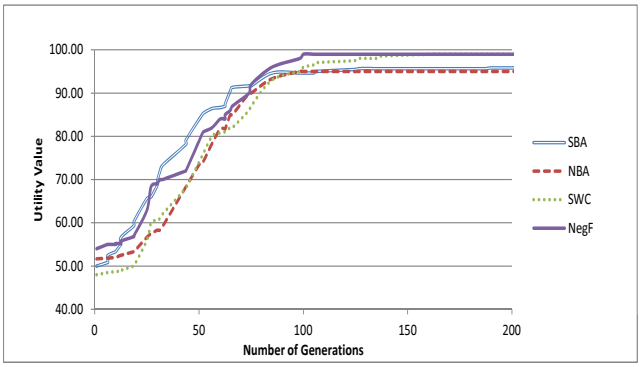

Fig. 10. Utility value comparison of NegF with similar service negotiation techniques

V. Conclusion and Future Direction

We have presented a framework (*NegF*) for Web services negotiation to enable consumers and providers in negotiating QoS parameters for SLA. We show the applicability of our approach using a GA based approach to conduct multi-party multi-objective negotiations. It integrates the concepts of multiple decision making preferences of the participants and allows the participants to either provide their own negotiation component or use a-priori decision model articulation and let the system handle negotiations on their behalf. We are working on extending our approach to incorporate dependency modeling for different QoS parameters among multiple services to formulate optimized solutions. We are also exploring the options of extending WS-Negotiation [13] and WS-AgreementNegotiation [42] to be used with our framework for negotiation. We are also working on a solution that moves away from the centralized approach in the favor of a more adaptive distributed model.

References

[1] O. Andrzej and K. Stanislaw. A new constraint tournament selection method for multicriteria optimization using genetic algorithm. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 501 –508, 2000.

[2] J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 101–111, 1985.

[3] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, 1987.

[4] D. Booth and C. K. Liu. Web services description language (wsdl). http://w3.org/TR/2006/CR-wsdl20-primer-20060327/, 2006.

[5] C. A. C. Coello, B. G. Lamont, and D. A. V.Veldhuisen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation Series. Springer, 2007.

[6] J .Csirik and G. J. Woeginger. Shelf algorithms for on-line strip packing. *Inf. Process. Lett.*, 63(4):171–175, 1997.

[7] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *Internet Computing, IEEE*, 6(2):86 –93, 2002.

[8] P. E. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation. *Artif. Intell.*, 164(1-2):23–46, May 2005.

[9] Z. Ezziane. Solving the 01 knapsack problem using an adaptive genetic algorithm. *Artif. Intell. Eng. Des. Anal. Manuf.*, 16:23–30, January 2002.

[10] M. Gen and R. Cheng. *Genetic algorithms and engineering design*. Wiley series in engineering design and automation. Wiley, 1997.

[11] K. Hindriks and D. Tykhonov. Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 331–338, 2008.

[12] H.Tung and R.J. Lin. Automated contract negotiation using a mediation service. In *Seventh IEEE International Conference on E-Commerce Technology*, pages 374 – 377, 2005.

[13] P. C. K. Hung, H. Li, and J. Jeng. Ws-negotiation: An overview of research issues. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[14] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[15] P. R. Kleindorfer, H. C. Kunreuther, and P. J. H. choemaker. *Decision Sciences: An Integrative Perspective.* Cambridge Univ. Press, 1993.

[16] S. Kraus. Mutli-agents systems and applications. chapter Automated negotiation and decision making in multiagent environments, pages 150–172. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[17] R. Y. K. Lau. Towards a web services and intelligent agents-based negotiation system for b2b ecommerce. *Electron. Commer. Rec. Appl.*, 6:260–273, 2007.

[18] Freddy Lecue, Usman Wajid, and Nikolay Mehandjiev. Negotiating robustness in semanticweb service composition. In *Seventh IEEE European Conference on Web Services)*, 2009.

[19] Y. Li and M. Gen. Nonlinear mixed integer programming problems using genetic algorithm and penalty function. In *IEEE International Conference on Systems Man and Cybernetics*, volume 4, pages 2677 –2682, 1996.

[20] Z. Malik and A. Bouguettaya. Evaluating rater credibility for reputation assessment of web services. In *WISE'07: Proceedings of the 8th international conference on Web information systems engineering*, pages 38–49. Springer-Verlag, 2007.

[21] Z. Malik and A. Bouguettaya. Reputation bootstrapping for trust establishment among web services. *Internet Computing, IEEE*, 13(1):40 –47, jan. 2009.

[22] N. Matos, C. Sierra, and N. Jennings. Determining successful negotiation strategies: An evolutionary approach. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, ICMAS '98, pages 182–, Washington, DC, USA, 1998. IEEE Computer Society.

[23] S. Matwin, T. Szapiro, and K. Haigh. Genetic algorithms approach to a negotiation support system. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(1):102 –114, jan/feb 1991.

[24] Z. Michalewicz and N. Attia. Evolutionary optimization of constrained problems. In *Proceedings of the Third Annual Conf. Evolutionary Programming*, pages 98–108. World Scientific River Edge, 1994.

[25] F. P. Mora and C. Y. Wang. Computer-supported collaborative negotiation methodology. In *J. Comput. Civil Eng*, pages 64–81, 1998.

[26] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[27] E. D. Nitto, M. D. Penta, A. G., G. Ripa, and M. L. Villani. Negotiation of service level agreements: an architecture and a search-based approach, 2007.

[28] X. Niu and S. Wang. Genetic algorithm for automatic negotiation based on agent. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 3834 –3838, june 2008.

[29] M. P. Papazoglou and W. Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16:389–415, July 2007.

[30] J. G. Qi, G. R. Burns, and D. K. Harrison. The application of parallel multipopulation genetic algorithms to dynamic job shop scheduling. *The International Journal of Advanced manufacturing Technology*, 16(8):609–15, 2000.

[31] M. Resinas, P. Fernndez, and R. Corchuelo. A bargaining-specific architecture for supporting automated service agreement negotiation systems. *Science of Computer Programming*, 77(1):4 – 28, 2012. ¡ce:title¿System and Software Solution Oriented Architectures¡/ce:title¿.

[32] Y. Sakurai, M. Yokoo, and K. Kamei. An efficient approximate algorithm for winner determination in combinatorial auctions. In *Proceedings of the 2nd ACM conference on Electronic commerce*, EC '00, pages 30–37, New York, NY, USA, 2000. ACM.

[33] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artif. Intell.*, 135(1-2):1–54, February 2002.

[34] Y .Shang and B. W. Wah. A discrete lagrangian-based global-searchmethod for solving satisfiability problems. *J. of Global Optimization*, 12(1):61–99, January 1998.

[35] Yoshiaki Shimizu. Multi-objective optimization of mixed-integer programming problems through a hybrid genetic algorithm with repair operation, 1999.

[36] M. Siddiqui, A. Villazón, and T. Fahringer. Grid capacity planning with negotiation-based advance reservation for optimized qos. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[37] K. M. Sim, Y. Guo, and B. Shi. Adaptive bargaining agents that negotiate optimally and rapidly. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1007 –1014, sept. 2007.

[38] K. M. Sim, Y. Guo, and B. Shi. Blgan: Bayesian learning and genetic algorithm for supporting negotiation with incomplete information. *Trans. Sys. Man Cyber. Part B*, 39(1):198–211, February 2009.

[39] G. Soremekun, Z. Grdal, R. T. Haftka, and L. T. Watson. Composite laminate design optimization by genetic algorithm with generalized elitist selection. *Computers and Structures*, 79(2):131 – 143, 2001.

[40] W3C Standard. Soap version 1.2. http://www.w3.org/TR/2007/REC-soap12-part0-20070427/, 2007.

[41] D. Whitley. The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In *Proceedings of the third international conference on Genetic algorithms*, pages 116–121. Morgan Kaufmann Publishers Inc., 1989.

[42] P. Wieder. Ws-agreementnegotiation. http://forge.gridforum.org/sf/go/doc15831, 2010.

[43] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67 –82, April 1997.

[44] G. Yee and L. Korba. Bilateral e-services negotiation under uncertainty. In *Proceedings of the 2003 Symposium on Applications and the Internet*.

[45] W. Yuping, L. Hailin, and Y. Leung. An effective uniform genetic algorithm for hard optimization problems. In *Proceedings Third World Congr. Intelligent Control and Automation*, pages 656–660, 2000.

[46] Yilei Zhang, Zibin Zheng, and Michael R. Lyu. Wsexpress: A qos-aware search engine for web services. In *Proc. IEEE Int'l Conf. Web Services (ICWS'10)*, pages 83–90, 2010.

[47] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8:173–195, June 2000.