1-1-2016

Automated Negotiation Among Web Services

Khayyam Hashmi
Wayne State University,

AUTOMATED NEGOTIATION AMONG WEB SERVICES

by

Khayyam Hashmi

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

2016

MAJOR: COMPUTER SCIENCE

Approved By:

Advisor Date

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my PhD advisor, **Dr. Zaki Malik**, for supporting me during these past years. I could not have asked for a better advisor, one that let me choose my path, help me along it and push me when needed. I really appreciate all the time he spent and all the patience he showed towards me. Secondly I would like to thank my committee members Dr. Shiyong Lu, Dr. Alexander Kotov and Dr. Brahim Medjahed for the constructive feedback and help they provided.

Lastly this whole journey would not have been possible without the support of my family in particular my parents. Mom and Dad this one is for you. Thanks a lot for all the support and help during this long journey. A big thanks to all the friends and family members who supported and helped me with this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: BACKGROUND AND INTRODUCTION

1.1 Introduction

In recent years, information system design has been influenced by the service oriented paradigm to facilitate easy integration between organizations that provide their services on Web Alonso et al. [2004]. With the increasing agreement on the functional aspects of Web services, such as using WSDL Booth and Liu [2006] for service description, SOAP for communication and WS-BPEL Standard [2005] for composing Web services, the research interest is shifting toward the non-functional aspects of Web services Papazoglou and Heuvel [2007]. Moreover, highly specialized component services have emerged that facilitate the process of on-demand composition of component services for formulating highly focused solutions on-the-fly. This highly distributed environment spanning across multiple enterprise boundaries pose certain limitations on different Quality of Service(QoS) components such as availability, reliability, scalability etc, of otherwise functionally equivalent services. These QoS parameters play a prominent role in both the performance as well as the total cost of ownership (TCO) of the system. These consist of both quantitative (availability 99.9 %) and qualitative (privacy, security) parts. Most of the quantitative components are not directly proportional in their cost/benefit curve i.e., 99.999% uptime vs 99.0% uptime. Hence this non liner curve naturally generates a disparity among the provided values for these components and opens them to negotiation.

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. The idea is to simultaneously optimize a series of multiple objectives, considering the constraints on the system. In SLA negotiations, each participant has multiple objectives, and adheres to multiple constraints, that bind those objectives.

1.2 Service Oriented Architecture

Service Oriented Architecture (SOA) is defined as "*a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains*" Ghosh et al. [2007] Katchabaw et al. [1996]. In other words, boundaries of SOAs are usually *explicit*, i.e., the services need to communicate across boundaries of different geographical zones, ownerships, trust domains, and operating environments. Moreover, explicit message passing is applied in SOAs instead of implicit method invocations. The services in SOAs are *autonomous*, i.e., they are independently deployed, the topology is *dynamic*, i.e., new services may be introduced without advanced acknowledgment, and the applications consuming a service can leave the system or fail without notification. Services in SOAs *share* schema and contracts. The message passing structures are specified by schema, and message-exchange behaviors are specified by contracts. Service compatibility is thus determined based on explicit policy definitions that define service capabilities and requirements.

Two major entities are involved in any SOA transaction: service customers and service providers. Figure 3.1 represents a typical Web service interaction model. Service providers offer their services by publishing their information (WSDL) in public directories(UDDI) Malik and Bouguettaya [2009a] Lin et al. [2008]. Customers then query these public directories to find similar service and then bind to the most suitable service Keromytis [2007], where input parameters are sent to the service provider and output is returned to the customer Guinea [2005] Denaro et al. [2006]. These directories serve as place holders and provide minimal functional information about

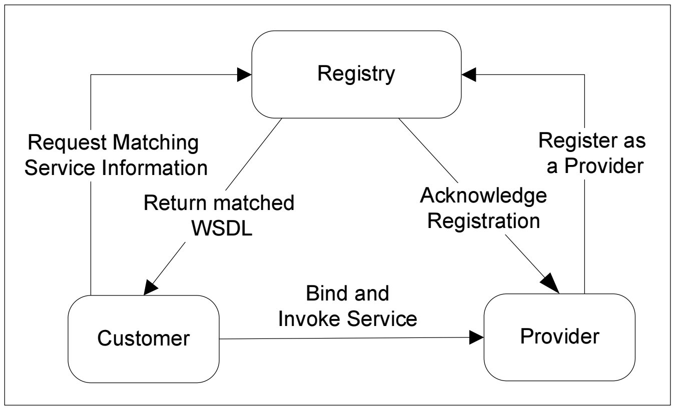


Figure 1.1: Service-Oriented Interaction Model.

the service. Service providers may use *tModels* Curbera et al. [2002a] to provide any additional information. These *tModels* have limited usability when it comes to negotiating non-functional components. A customer looking for a Web service could benefit from a service that provided functional and non-functional requirement could provide the most effective solution by simultaneously negotiating with multiple providers.

1.3 Negotiation

Negotiation is a process that can be defined as the interplay of offers and counter-offers between two entities, with different criteria and goals, working to reach a mutually acceptable solution. Negotiation enhances acquisition opportunities and enables flexible communication that can lead to better solution Yao et al. [2006].Negotiations play a prominent role in the decision making process of different aspects of human life, such as business, scientific, social and political interactions etc Kleindorfer et al. [1993] Mora and Wang [1998].

Web service negotiation is uncertain (due to incomplete information of both parties) and knowledge intensive. Performing such negotiations manually is likely to be ad-hoc and time-consuming. Software that facilitates automated web service negotiations is valuable not only for the consumers and the providers to continuously customize their needs and tailor their offerings, but also to discover overlooked solutions and to maintain documented rationales for future references and reuse.

Just as Real-world negotiations do not require the parties to reach a negotiated agreement; similarly, the automated negotiation has the same options. An entity can choose “no deal” if it cannot negotiate a satisfactory agreement. Furthermore, there are distinct negotiation strategies for “open” and “closed” marketplaces. A closed marketplace is based upon a predefined set of users, who “enroll” in the marketplace and agree to a certain set of rules. An open marketplace has no such agreement; entities are welcome to enter and exit at any time, and are not required to agree to any rules. This adds to the complexity and uncertainty of information. Hence entities need to take into account these uncertain information patterns and deal accordinglyBeam and Segev [1997].

An automated negotiation mechanism requires at least three components Figure 1.2: high-level protocol, objectives and strategiesLomuscio et al. [2004]. The high-level protocol controls the negotiation process depending on types of negotiations (e.g., auction). The objectives of each of the parties are based on a set of negotiable criteria, representing various parameters along with their respective domain values (e.g., price range) and are often modeled as decision making problems. Negotiation strategies include mechanisms (rules and knowledge base) that the agent employs to generate and evaluate offers.

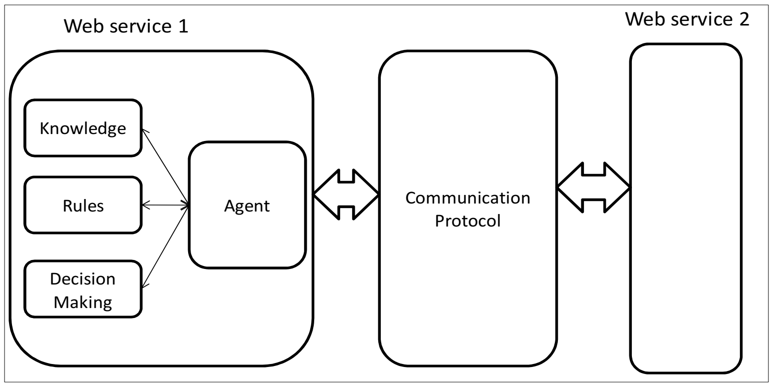


Figure 1.2: Automated Negotiation Components

CHAPTER 2: LITERATURE REVIEW

2.1 Existing Negotiation Systems

Automated negotiation systems have been around for a while. Their applications range for simple auction bots to sophisticated decision support systems. Below are some examples of such systems that are currently being used for automated negotiations.

- NegotiatorBui and Shakun [1996] seeks to guide negotiators to move their individual goals and judgments to enhance the chance of achieving a common solution. It supports problem adaptation through information sharing, concession making, and problem restructuring or reforming. However, NEGOTIATOR helps the negotiators to make decision only without any support to other entities involved in negotiation activities.

- InspireKersten and Noronha [1999] (InterNeg Support Program for Intercultural Research) is a Web-based prototype NSS for intercultural as well as intracultural negotiations. INSPIRE can conduct negotiation anonymously, evaluate the goodness of an offer, and review the history of a negotiation. INSPIRE supports the tasks of preference assessment, analysis of alternative offers, offer exchange, counter-offer evaluation, and assessing compromise efficiency using the Pareto-optimality approach Li [2002]. Although INSPIRE supports the communication among negotiators by exchanging messages, it does not deal with the interactions among different entities in negotiation activities.

- NegoPlanS. Matwin and Koperczak [1996] is an expert system to structure the strategic issues. It uses a rule-based formalism to represent negotiation activities, to develop a problem representation, to provide information, and to maintain a structure that allows the consistency and validity of the model to be verified. A strategy can be developed by the knowledge representation from goals. NegoPlan focuses on the strategies for negotiation activities only, providing no support for other aspects in negotiation activities.

- MIT Media Lab's KasbahChavez and Maes [1996] is an online, multi-agent consumer-to-consumer transaction system. Users create autonomous agents to buy and sell goods on their behalf, and also specify parameters to guide and constrain an agent's overall behavior. Buying and selling agents meet and negotiate in the Kasbah Marketplace directly.

- Tete-a-TeteLab [2000] provides an integrative negotiation approach to retail sales. Shopping and sales agents negotiate across multiple terms of a transaction, including warranties, delivery times, service contracts, return policies, loan options, gift services, and other merchant value-added services.

- Michigan AuctionBotWurman et al. [1998] is a general purposed Internet- based auction server hosted by the University of Michigan. Sellers can create new auctions on AuctionBot by choosing from a set of pre-defined auction types. They can also enter their specific auction parameters such as clearing time, minimum bid increment and proxy bids. In general, a seller will create and auction item, set a reservation price and let AuctionBot manage the bidders and enforce the bidding rules. One distinct advantage of AuctionBot is that it offers Application Programming interfaces (APIs). Buyers can use these API's create their software agents to bid on their behalf at the AuctionBot virtual auction house. Commercial auction sites such as eBay (www.ebay.com) only allows negotiations over a single issue of price. Although these kind of e-markets or auction houses are popular for B2C eCommerce, they are ineffective for B2B Commerce where multiple negotiation issues are often explored.

- MAGNETJaiswal et al. [2004] is a multi-agent marketplace which supports a variety of types of transactions. The transactions range from simple buying and selling of goods and services to complex multi-agent negotiation of contracts.. The MAGNET agents attempt to gain the greatest possible profits from their activities, hence it is suitable for B2C eCommerce where cooperative negotiation behavior is possible. A MAGNET agent can take the role of either a provider or a consumer. To trade in the market, a consumer agent generates a plan which is a collection of tasks with time and precedence constraints, and then submits one or more Requests for Quotes (RFQs) to providers via the market. Any provider who wants to bid will respond. After receiving the bids, the consumer agent decides which bids to accept. Finally, the winning provider execute the tasks included in their winning bids.

2.2 Communication Protocols for Negotiation

A communication protocol defines the syntax, semantics, and synchronization of communication. It is a system of digital message formats and rules for exchanging those messages in or between computing systems. There are many communication protocols that are being used to conduct negotiations. This section discusses some of the widely used negotiation protocols.

WS-Agreement

WS-Agreement [GRAAP] is a protocol for establishing agreements between two parties, such as between a service provider and consumer. It uses XML for specifying the nature of the agreement, and agreement templates to facilitate discovery of compatible agreement parties.

There are three parts to this specification and the language allows them to be combined togather: a schema for specifying an agreement, a schema for specifying an agreement template, and a set of port types and operations for managing agreement life-cycle, including creation, expiration, and monitoring of agreement states.

There are two layers of WS-Agreement as shown in Figure . The agreement layer provides a Web service-based interface that can be used to create, represent and monitor agreements with respect to provisioning of services implemented in the service layer. The service layer represents the application-specific layer of the service being provided.

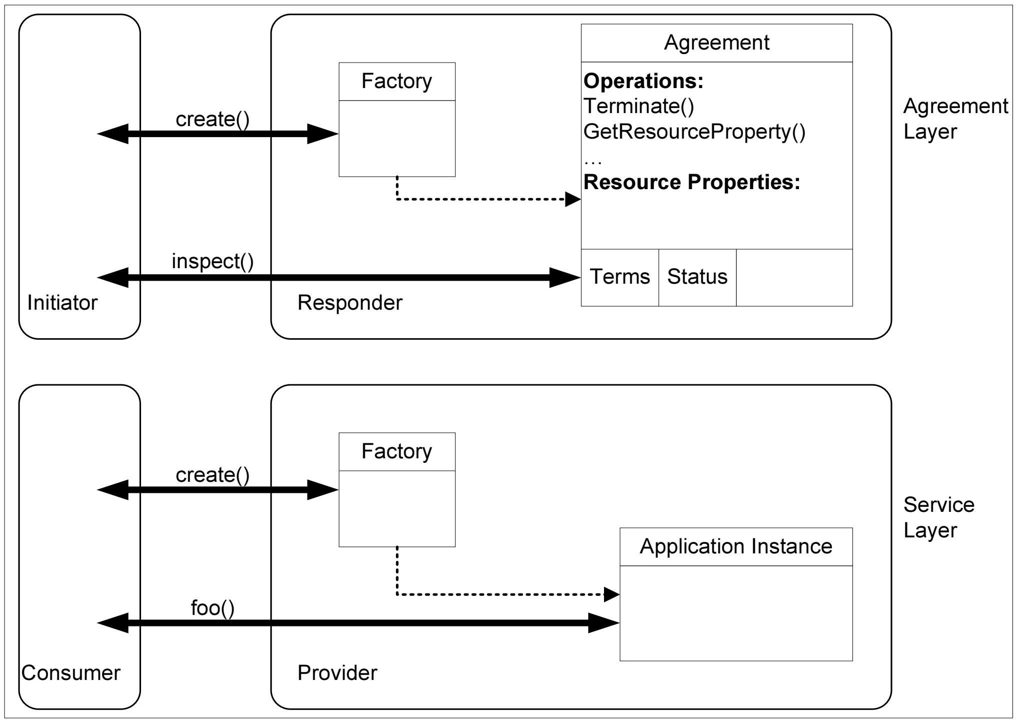


Figure 2.1: WS-Agreement Conceptual Layered Service Model.

Although WS-Agreement does not have any negotiation specific structure but there had been discussions for using it in negotiating agreements among parties Andrieux et al. [2004]. Pichot et. al Pichot et al. [2008] have used WS-Agreement to negotiate SLA's for resource orchestration in Grids.SLAs are a basic building block for Grid resource orchestration and distributed resource management. A bilateral WS-Agreement based negotiation process is used to dynamically negotiate SLA templates. One option is for the originating agent to negotiate separately with each Autonomous System (AS) along each potential path to ensure that an end-to-end path is available. The dominant choice however, is to use a cascaded approach where each AS is responsible for the entire path downstream of itself. To rely on WSAgreement and minimize the extensions to the proposed standard, the idea is not to negotiate SLAs but to negotiate and refine the templates that can be used to create an SLA. An agreement template defines one or more services that are specified by their Service Description Terms (SDT), their Service Property Terms (SPT), and their Guarantee Terms (GT). Additionally an agreement provider can constrain the possible values within the SDTs, SPTs, and GTs by defining appropriate creation constraints within the templates.

OpenCCSA [2007], AgentScapeMobach et al. [2006] and VIOLA MetaScheduling Service MSSWaldrich et al. [2006] use negotiation to refine offers and requests in order to create SLAs. As WS-Agreement does not include a protocol for negotiating the terms of an SLA (but an "accept/reject" protocol for the whole SLA), these three approaches currently use proprietary extensions of WS-Agreement for the negotiation.

Contract Net

Contract NetSmith [1980] is a generic protocol focused on the negotiation task. It is viewed as a task having four components 1) it is a local process that does not involve centralized control, 2) there is a two-way exchange of information, 3) each party to the negotiation evaluates the information from its own perspective, and 4) final agreement is achieved by mutual selection.

Each entity is referred to as a node and the collection of nodes is known as a contract net. Each node in the net takes on one of two roles related to the execution of an individual task: manager or contractor. A manager is responsible for monitoring the execution of a task and processing the results of its execution. A contractor is responsible for the actual execution of the task. Individual nodes are not designated a priori as managers or contractors; these are only roles, and any node can take on either role dynamically during the course of problem solving. Typically, a node will take on both roles, often simultaneously for different contracts. As a result, nodes are not statically tied to a control hierarchy.

A contract is established by a process of local mutual selection based on a two-way transfer of information. In brief, available contractors evaluate task announcements made by several managers and submit bids on those for which they are suited. The managers evaluate the bids and award contracts to the nodes they determine to be most appropriate. The negotiation process may then recur. A contractor may further partition a task and award contracts to other nodes. It is then the manager for those contracts. This leads to the hierarchical control structure that is typical of task-sharing. Control is distributed because processing and communication are not focused at particular nodes, but rather every node is capable of accepting and assigning tasks.

The basic message constructs of contract protocol are Task Announcement, Task Announcement Processing, Bidding, Bid Processing, Contract Processing, Reporting Results, Termination, and Negotiation Tradeoffs.

Lecue at al.Lecue [2009] used a variation of Contract Net protocol for Semantic Web service composition. The issue of aligning data flow in semantic web service composition to ensure the robustness when executing the composed service by preventing any cases when the wrong type of data is passed on from one service to the next is tackled by proposing a unique solution that ensures the robustness of data flow when automatically composing web services through the use of agent-based negotiation between web service providers.

The idea is to apply the abduction process to find out the extra description needed to make the four links i.e. Exact, Plugin, Intersection, Disjoint between services robust. The extra description refers to the information required by input of B but not provided by the output of A to ensure correct data flow between the services A and B. This information is then used to compute the conjunction of all the concept abduction information in a service composition which is later on negotiated between different service components. The main idea of negotiation is to allow all the participants to be equally vocal in the process rather than the traditional techniques that only allow the initiator to control the negotiation process Lecue at al use a customized form of FIPA for Intelligent Physical Agents [2002] Iterated Contract Net Interaction Protocol. It allows the agents to use a wide range of strategies for negotiation. The assumption is that the Initiator agent is responsible for the provision of most specific Extra Description with other participant agents.

The basic message passing constructs of the protocol are Call-for-Proposal, Propose, Refuse, Accept-Proposal and Inform. The initiator agent will evaluate and rank proposals using any of the known set partitioning algorithms. The initiator agent will generate a call for proposal and collect the replies it receives. It then evaluates the received proposals and compute the best course of action. At every round of proposals, the initiator agent will calculate the set of outstanding elements and, unless this set is empty, will proceed to invite proposals covering these outstanding elements. Depending on the configuration of individual initiator agents, the proposal gathering may stop when all known service providers have been contacted or after a fixed number of iterations, even if there are still outstanding elements not covered by proposals. The initiator agent will then proceed to evaluate and rank proposals using any of the known solutions to the set partitioning problemBalas and Padberg [1976]. If a solution is found, it will accept the proposal by notifying the successful party and this will serve as a binding contract.

Another variation of CNP "Iterative model of Contract Net Protocol for negotiation" is used by PaurobballyPaurobally et al. [2007a]. In the CNP The manager initiates the negotiation process through a call for proposals (cfp) announcing the task specification to the contractors. A contractor receiving the cfp evaluates it and decides whether to answer with a refusal or a proposal to execute the task. The manager receives the contractor's proposals and in turn decides which proposals to accept and which proposals to reject. Rejected contractors consider that the negotiation has terminated, while accepted contractors must expedite the task and send back the results of their work to the manager. In the iterative CNP, the process of a manager invoking a cfp and a contractor submitting a proposal is repeated several times until either the manager decides to accept a proposal or the manager's deadline is reached. Thus there are several rounds of proposals in an iterative CNP, and the contractors aim to improve on their earlier proposals to be accepted by the manager. To support this added functionally Two extensions to CNP are implemented. They are do_negotiation and get_results.

Each negotiation party has its own preferences that are used to calculate what values in the negotiation subjects it accepts or responds with. They are defined in a class pref_terms to have private attributes preferred value, reserve value, utility, and weight. Each attribute has an associated preferred value with it. There is usually a relationship between the actual value, preferred value and reserve value, depending on whether that participant wants more or less of that issue. For a contractor preferring a high value for price, then the relation between its preferences is:

reserve value of price $<$ actual value of price $<$ preferred value of price.

The utility of an issue is how much is it worth to a participant. A higher utility means a higher worth. The weight of an issue is its importance relative to the other attributes.

Three different strategies are implemented using the Iterative model of Contract Net ProtocolPaurobally et al. [2007b]. In the truth telling strategy, both the manager and contractors reveal their true preferences. Thus, each cfp is constructed with its preferred value for each issue. A service replies with a proposal where each issue is given its own preferred value for each issue. If the cfp lies outside the reserve values for negotiable issues, then the service's proposal is grounded with the service's reserve values.

In the decrement strategy, the participants have evaluation and generation margins, against which they evaluate a cfp and generate a proposal above or below their reserve values. Thus the parties have chance to converge to an agreement during the negotiation process instead of rejecting immediately.

In the CNP, there is a deadline for receiving proposals from contractors. In the time dependent strategy, a proposal is computed as a function of the proportion of the remaining time over the total time allocated to the CNP. A contractor also has its personal deadline and knows the manager's deadline for receiving proposal. As the time left decreases, the concession rate of a participant increases.

WS-Policy

WS-Policy [W3C] provides a grammar for expressing Web services policies. WS-Policy is used to specify policy information on a broad range of service requirements, preferences, and capabilities. The WSPolicy is represented by a policy expression that is an XML Infoset representation of one or more policy statements. The WS-Policy includes a set of general messaging-related assertions defined in WSPolicyAssertions and a set of security policy assertions related to supporting the WS-Security specification defined in WS-SecurityPolicy . Nevertheless, the current WS-Policy specification also mentions that WS-Policy by itself does not provide a negotiation solution for Web services.

In Comuzzi and Pernici [2005] Marco et. al propose a framework based on WS-Policy for negotiation of Quality of Service attributes between web services. The approach relies on the definition of an extended SOA in which a service index with QoS information is available. Service provider publishes the non-functional attributes, that may be negotiated by the consumer, in the WSDL. This QoS registry could be stored along with WSDL using WSOL. Where WSOL is in fact, is a language, compatible with WSDL, to specify different service offerings for the same service identified by the different values or constraints on the service QoS attributesTosic et al. [2005]. It can include different domain schemas on which the QoS could be defined.

The framework is based on a Negotiation Broker support that knows the decision model for the provider (for semi automated approach) or both the provider and consumer(for a fully automated approach). Negotiation Broker, via different interfaces, provides functionality to let the service provider and the consumer specify policies and identify the services to which they have to be associated. In the more general case, a different policy, and thus, negotiation decision model, can be defined for every single service. However policies can be used for sets of similar services. WSLA is used to describe the contract for non-functional features of the service. WSLA has been designed to support flexible and negotiated Service Level Agreements and provide a framework for managing and monitoring contract specifications Alexander and Heiko [2003]. Service Consumer Interface is used by the consumer to present its policies and allows the broker to present offers. In some cases the user may not specify its policies and this module could infer them from the context. Service Provider Interface allows the provider to submit its policies and decision model. Providers and User policies module stores the WS-Policy documents in which providers and consumers specify preferences for customize the utility functions and negotiation strategies. Provider and User behavioral Engine uses the set of policies to extract the information related to the web service invocation. Negotiation Engine is where the whole process of negotiation takes place and shows the final results. In the end the WSLA generator translates the outcomes of the negotiation process in a WSLA document that constitutes the electronic contract between the consumer and the service provider.

Negotiation supports between Web services requester and provider on an agreement about security requirements and services can be foreseen for WS-Policy in the future.

In literature we see examples of system that are constructed on top of Xplore. One such example is e-AllianceCastellani et al. [2002]. This creates framework that supports different parts of a an alliance in negotiations that work with the limitation of autonomy in inter-organizational alliances. It focuses on how to represent decentralized organizations, modeling the coordination of different concurrent interactions, formalization of negotiations, deploying and maintaining an alliance during its life cycle and creating administrative contracts. The e-Alliance infrastructure is organized in three layers. A first, application dedicated, layer specializes the generic mechanisms provided by the other two layers according to the specific domain. A second layer is dedicated to the support of job insourcing/outsourcing within an alliance and comprises three facilities: AllF (alliance life-cycle management), ConF (contract management), and NegF (negotiation). The third, middle ware and coordination, layer (CooF) offers generic mechanisms to enact negotiations in a distributed environment. The CooF is shared across the partner sites, while the two other layers are replicated on each partner site, enabling a decentralized negotiation and preserving the autonomy of the partners. NegF is main component that manages the negotiations both on global level (negotiations on different jobs) for each party and at a specific level (negotiation on the same job with different participants) by coordinating itself with the NegF of the other partners through the CooF.

A negotiation is organized in three main steps: initialization; refinement of the job under negotiation; and closing. The initialization step allows to define what has to be negotiated (Negotiation Object) and how (Negotiation Framework). In the refinement step participants exchange proposals on the negotiation object trying to satisfy their constraints.

The manager may participate in the definition and evolution of negotiation frameworks and objects. Manager takes all the decision in co-ordination with its NegF. For each negotiation, a NegF manages, one or more negotiation objects, one framework and the negotiation. Negotiation Frameworks gather requirements of managers on negotiations, formalizing plans for the interaction process and the degrees of autonomy in decisions and actions of the NegF. A manager can specify some global parameters: duration; maximum number of messages to be exchanged; maximum number of candidates to be considered in the negotiation and involved in the contract (contractants); tactics; protocols for the NegF interactions with the manager and with the other NegFs.

Negotiation middle ware CooF: It is based on Xplore and supports processes provided by the facilities in the second layer. CooF is the coordinator that supports multi-party, multi-directional, multi-attribute negotiation. This process is modeled by a negotiation graph.

2.3 Negotiation Agents

Negotiation agent could be thought of as the brains behind the negotiation process. This component interacts with the domain knowledge and the system rules to calculate the usefulness of an offer and then generate counter offers against it. Hence, it is responsible for the decision making process. There are different types of negotiation agents that adhere to different auction types.

Trade-Off Based Negotiation

In trade-off based negotiations the concerned parties make tradeoff on different negotiation parameters based on their respective importance (weights) to the negotiator. Normally each round of negotiation has a slightly different feature vector based on the counter offer generated in the previous negotiation round. This cumulative information is used to generate future offers and hence reach a mutual agreement.

Patankar et. al. Patankar and Hewett [2008]used a tradeoff based negotiation mechanism for web service procurement using a bilateral protocol to govern interactions between the negotiation parties. In negotiation each party can define its own set of evaluation function, utility function and offer generating algorithm. For simplicity in Patankar and Hewett [2008] both parties share the same generic tradeoff mechanism for automated offer generation while each party can have its own set of objectives and evaluation function.

The multi round negotiation algorithm used contains strategies that focus on generating a set of offers that have the same utility as the current offers and is based on the offers generated by the opponent agent in the previous round. The idea is to exploit the current utility as much as possible. The generated set of offers is presented to opponent agent that chooses the offer that is most suitable to its preferences based on its evaluation function. The negotiation continues until the opponent presents an offer that is of an equal or greater utility than the agent's previous offer. A deadlock condition may be reached if no offer that is of a higher utility to the opponent than the previous offer is being generated. In such a situation the agent reduces its utility expecting to find, in the lower level an offer that satisfies both agents. This strategy ensures that the agent concedes utility in a more rational way.

The offer generating algorithm works based on the fact that offers are generated by splitting the utility gain randomly among the criteria variables. Firstly, the maximum utility that can be gained for each criterion is computed as the difference between the full utility of the agent's preference for that criterion and the value of the criterion in the opponent's last offer . The overall weighted utility is computed as the weighted sum of individual utility gains. Randomness is used for selection of criteria values, a degree of tolerance is calculated and included to guarantee convergence . The process of consumption of utility begins by allowing each criterion to consume a random amount of utility. At this point, the agent's knowledge about the opponent's criteria preferences can be used . By consuming the utility for those criteria that are unimportant to the opponent, they have a higher probability of generating satisfactory offers to the opponent. This offer generation algorithm is run as many time as needed to reach a negotiation or till the time a deadlock is reached.

Auction Based Agents

Auction could be described as the simplest form of negotiation. The consumer bids on the price of an item. Provider has the option of either accepting the offer or rejecting it.

Preist et. al Preist et al. [2003] discuss a service composition agent that both buys components and sells services through auctions. It buys component services by participating in many English auctions. It sells composite services by participating in Request-for-Quotes reverse auctions. Because it does not hold a long-term inventory of component services, it takes risks. It makes offers in reverse auctions prior to purchasing all the components needed, and bids in English auctions prior to having a guaranteed customer for the composite good. The authors present algorithms that are able to manage this risk, by appropriately bidding/offering in many auctions and reverse auctions simultaneously. The algorithms withdraws from one set of possible auctions and move to another set if this produces a better-expected outcome, but will effectively manage the risk of accidentally winning outstanding bids/offers during the withdrawal process. In this work authors only discuss the scenarios with English auction type of negotiation with no one-on one negotiation. It is assumed that the agent maintains a probabilistic model of expected outcomes of each auction based on past performance of similar auctions. They model a fixed price seller that guarantees a sale of a given product at a price p as an auction with a 100% certainty of closing at p. In a trivial case we can always sell a product at 0 price. The algorithm will initially identify the set of options which maximize its a-priori expected utility.

These options will consist of a reverse auction for a given composite service, together with a set of English auctions for the required components. It will place bids in these forward/reverse auctions and will continue to compete in these auctions, placing more bids when outbid. However, if sufficient competing bids are placed to reduce the expected utility of this set of auctions, then it may change to another set of auctions which can generate the same composite service. It will do this if the expected gain from changing to this new bundle outweighs the expected cost of currently held bids which appear in the old bundle but not in the new bundle. If competing bids are placed in one of the reverse auctions it is participating in, and the expected value of that auction decreases sufficiently it may withdraw from that reverse auction. It may use the associated forward auctions in another option, or may withdraw from them as well. The problems of not committing and evaluating each option are solved by limiting the search space to promising offers only.

Negotiation with Uncertain Data

Having as much information as possible about the other parties is important to strengthen one's negotiation capabilities Nguyen and Jennings [1998]. Unfortunately, more often than not, we only have partial information about the negotiation context Luo et al. [2003]. Hence it is very important to be able to manage different types of unknown parameters about the negotiation.

- Negotiation Under Uncertainty

  Yee et al.Yee and Korba [2003] present 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. So the main idea is that of using the negotiation experiences of trusted people with matching interests as aids in deciding which negotiating alternatives and offers should be employed. For legal and other purposes, records should be kept of e-service negotiations (e.g. for non-repudiation). Authors use this information plus a reputation approach to provide a means for enabling parties to more rapidly carry out a negotiation based upon the experiences of others.

- Dynamic Outside Options

  Li et al.Li et al. [2006] discuss a model for bilateral negotiations that considers the uncertain and dynamic outside options. Outside options affect the negotiation strategies via their impact on the reservation price. The model is composed of three modules: singlethreaded negotiations, synchronized multithreaded negotiations, and dynamic multithreaded negotiations. These three modules embody increased sophistication and complexity. The single-threaded negotiation model provides negotiation strategies without specifically considering outside options. The model of synchronized multithreaded negotiations builds on the single-threaded negotiation model and considers the presence of concurrently existing outside options. The model of dynamic multithreaded negotiations expands the synchronized multithreaded model by considering the uncertain outside options that may come dynamically in the future. Poison Process is used to simulate the arrival process of uncertain (dynamic) options. This process follows Poison distribution.

  It further incorporated 4 different heuristic based function to determine the Utility of an option. Three of them namely, conservative estimates, medium estimates and Uniform approximation is based on the prediction of reverse English auction and assumes that the difference between the bids (the required minimum bid) is very small and hence the gain in Utility from the second highest bidder is almost equal to that of the highest bidder. The fourth one Learning is based on the assumption that the system can learn the negotiation history by some means. That could be a survey or a similar negation in past with similar parameters. This comes handy in the Navy Detailing scenario (the motivating example of the chapter) where similar jobs are posted frequently and there are multiple rounds of negotiations.

Experimental analysis is provided to characterize the impact of outside options on the reservation price and thus on the negotiation strategy. The results show that the utility of a negotiator improves significantly if he/she considers outside options, and the average utility is higher when he/she considers both the concurrent outside options and the foresees future options.

Genetic Algorithm based Negotiations

Negotiations are a special class of group decision making problems. Multiparty multiobjective negotiations add a lot of complexity to the already hard problem of negotiation. Negotiation problems can be formulated as constrained multiobjective optimization problems and they can be solved using techniques appropriate for such problems. The idea is to optimize a series of objectives simultaneously while considering constraints on the system. In the case of negotiations, there will be multiple objectives for each of the negotiation participants. The multiobjective multiparticipant nature of negotiation problems suggests such problems are quite complex.The GA approach is consistent with the complex nature of real-world negotiations and is, therefore, capable of addressing more realistic negotiation scenarios . Since genetic algorithms and evolutionary algorithms in general search for entire populations of solutions, they are well suited for multicriterion problems.

Rubenstein-Montano et. al. propose a weighted sum genetic algorithm to support multiple-party multiple-objective negotiationsRubenstein-Montano and Malaga [2002]. A weighted sum approach is used to handle the multiple objectives of each participant. Since all the participant start negotiation from a different position hence they will also have different preference for those objectives and are described by how far their current position is from the objective. Hence the objective is to minimize this distance. The genetic algorithm solution is represented as a 2-Dimentioanl matrix, representing the participants and objectives.

A new operator Trade is implemented to improve the performance of GA and to simulate the actual trade in a negotiation scenario. It is implemented for the exchange of recourses among different participant. Trades must occur between two distinct objectives so that the rows selected for trade must also distinct. Participants can trade some or all of their available objectives and there is at most one trade per pair per generation. Trade is implemented probabilistically. Each matrix in the population is reviewed for possible trading. The participants and objectives involved in the trade are selected randomly. Then it is decided if a trade will actually occur based on the willingness of participants to consider a possible trade. The trade only occurs if both randomly selected participants are willing to make a trade. Essentially, willingness to trade is higher if a participant has more of an objective than he ideally wants. The crossover operator is invoked after trading. Roulette-wheel selection is used for selecting solution pairs for crossoverBaker [1987].

2.4 Combinatorial Negotiations

Combinatorial negotiation is the type of negotiation where entities can negotiate on a combination of items rather than negotiating independently on each item from a set of items. Combinatorial negotiation stemmed from the traditional combinatorial auctions. In a combinatorial auction, a set M of items, $|M|$ = m, is sold to n bidders. The combinatorial character of the auction comes from the fact that each bidder values bundles of items, rather than valuing items directly. The idea is to find such a partition of the items so that the return is maximized for the auctioneer.

Rodrguez-Aguilar et. al Rodrguez-Aguilar et al. [2003] has proposed iBundler a Combinatorial Negotiation based decision-support service for highly constrained negotiation scenarios. iBundler acts as a combinatorial negotiation solver for both multi-item, multiunit negotiations and auctions. Thus, the service can be employed by both negotiating agents and auctioneers in combinatorial auctions. It consists of a three main components. The Manager agent take care of all the communication. It provides brokering services of RFQ, collection of bids, winner determination and contracting services. The Translator agent perform the necessary XML translations for the Solver and FIPA-compliant descriptions for the Manager agent. Solver component extends the *iBundler* with the offering of an XML language for expressing offers, constraints and requirements. New ontologies have been developed based on the negotiation protocols by Tamma et. al Tamma et al. [2002]. The winner determination is modeled as a mixed integer problem similar to the the binary multi-unit combinatorial reverse auction winner determination problem in Sandholm et al. [2002] with side constraints Sandholm and Suri [2001].

Computational Complexity

The winner determination problem in combinatorial auctions in general are NP-hard Sandholm et al. [2002]. However, for some special cases with restricted subsets there exists polynomial time algorithms. Rothkopf et al Rothkopf et al.

[1998] found that if the family of subsets on which a bidder can bid is limited to hierarchical subsets, meaning that every two subsets are disjoint or one is a subset of the other, then the winner determination problem can be solved in polynomial time. The problem of finding an optimal allocation for a combinatorial auction when a linear order exists among the items and where bidders can only bid on subsets of consecutive items is also shown to be polynomially solvable. Furthermore, Rothkopf et al Rothkopf et al. [1998] prove that a combinatorial auction where bidders can bid on subsets of a cardinality of at most two has a polynomially solvable winner determination algorithm. Tennenholtz Tennenholtz [2002] presents a combinatorial network auction, which he proves is computationally tractable. In this auction, the items are assumed to be arranged in a tree, where every node corresponds to an item. The idea is that bids can be submitted only on subsets of items that form a path in the network. If the items are structured in a directed acyclic graph and the bids are allowed on any directed subtree, the winner determination problem already becomes NP-hard Sandholm et al. [2002]. Sandholm Sandholm et al. [2002] also presents some more special cases of combinatorial auction which have polynomial time algorithms.

2.5 Other Approaches

Some of the recent research has been focused on implementing negotiation as a web service and using community knowledge to obtain better SLAs.

Negotiation as a Service

Bui et. al Bui and Gachet [2005] present a basic architecture of an electronic market with a broker service capable of offering negotiation and bargaining services. Instead of each service taking on the responsibility of implementing the whole process of negotiation and communication mechanism these service could be centralized and reused by all the services.

The ultimate goal of the broker is to quickly and efficiently match the supply and demand of Web service. As each transaction is likely to be unique in nature, the broker service should provide a comprehensive set of decision-making, negotiation, bargaining and contracting tools. The broker should also assist its users before, during and after the market transaction takes place.

The assumption is that these Web services could be initiated in one of the two following fashions: pull and push. In the pull mode, the negotiation and bargaining services would be called upon request by either side of the supply and demand. These Web service-on-demand would continue until their users choose to terminate them. In the push mode, the Web service broker would constantly watch and monitor the market, observing the movements of the supply and demand of Web service. Whenever the Web service broker notices that there would be an opportunity for it to offer an genuine service, it would offer its service to its customers. The later case is a rare one.

The framework consists of seven services. Service Discovery deals with the functionality of domain study and finding similar services to the requested service. Adapting and Pricing deals with the task of formalizing of the interoperability/ compatibility in term of semantics and standardizing the price for future communication. Service Ranking deals with the process of short listing the potential clients. This could include customer preferences, previous clients, reputation on top of the functional and pricing attributes. Service Bargaining deals with negotiating the deal with potential clients. Best price adaptation deals with coming up with a comprehensive utility function to select a service. Contract composition is used to finalize and execute any contracts that are generated in the process of these negotiations.

Lastly all these are service are composed to construct a negotiation manager that would perform all the above mentioned processes for a client. The idea is to give the manager all the necessary info and let him negotiate a contract for you and come back with the best possible offer. If that works for the client it can accept the offer and formulate a contract.

Community Based Model

Cappiello et. al Cappiello et al. [2007] propose a model to generate service level agreement on-the-fly. Just before the invocation is performed, the quality of the service is negotiated in order to generate a service level agreement tied to that specific invocation. Their approach relies on a quality model that supports both users requirements and providers capabilities definition. To mediate between these two standpoints, they introduce the community as the actor able to provide a shared knowledge about the quality of a service in a specific application domain. The community defines which relevant aspects of a service can be used as search discriminants in service discovery.

The capability of a service to provide these attributes and their corresponding values is defined in terms of WSOL and WS-Policy as a standalone document to go with the WSDL of each service. It is assumed that all the quality dimensions identified by the community and the standard offering values of these attributes would exist in this capability document. This document would be a starting point for the user negotiation.

The user also defines its own set of required quality attributes and the corresponding attribute values. Each attribute is also assigned a weight based on its importance to the user process. Authors use AHP (Analytic Hierarchy Process) a decision making technique to come up with these weights. To help facilitate the user in identifying these quality attributes authors use profiling. Profiling is the technique through which data are collected and manipulated with the goal of identifying and describing the profile of an entity, such as a user, an object, a product, or a process. It is a structured representation of the information that describes users and their preferences along the services that they require.

Negotiation only takes place if the user have some extra budget to spare after the initial offering. May be for increased value or increased quality of a specific attribute. Authors use two negotiation strategies for the user to decide how to split extra budget across the different negotiable quality dimensions, that we name the vertical and the horizontal strategies. When adopting the vertical strategy, the user has the objective to maximize the quality associated to the highest priority dimension. When the quality of this dimension is maximized, that is, when the remaining extra budget exceeds the price of the negotiation service class , then the algorithm switches to the maximization of the quality of the second highest priority dimension. The horizontal strategy is adopted when the user wants to split the extra budget on the negotiable quality dimensions proportionally to the priorities.

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 general there are two prevalent approaches for building recommender systems: content-based (CB) Mooney and Roy [2000] and collaborative filtering (CF) Goldberg et al. [1992]. 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.

Traditionally, both CB and CF systems have been based on explicit input from the user, usually provided by rating a set of items. To avoid this extra burden on the user, leveraging implicit interest indicators Claypool et al. [2001], such as purchase history, views, clicks, or queries, has recently become more popular in recommender systems. Although implicit acquisitions place a cognitive burden on the users Morita and Shinoda [1994], however, the inferences drawn from from user interactions are not always valid because of the indicators of users interests are often erratic Kelly and Teevan [2003]. User profiles are often difficult to obtain and there quality is generally not that great. The current source of user profiling is mainly based on user ratings. The rate of users leaving comments is very low and is often affected by the data sparseness and cold start problem Sarwar et al. [2001] Schein et al. [2002].

Several studies have suggested incorporating direct social relationships in CF systems. ReferralWeb Kautz et al. [1997] 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., Bonhard and Sasse [2006] Golbeck [2006] Sinha and Swearingen [2001]), music Konstas et al. [2009] and news stories Lerman [2006]. 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. The popular book marking site del.icio.us has been of special interest by researchers.Heymann et al. analyzed del.icio.us for social tagging and shown that searches could be improved by navigable hierarchical taxonomy of tags that has been derived by the tag usage. Li et al. Li et al. [2008] analyzed the same site and find a high similarity between the tag vector of a URL and its keyword vector, as extracted from the corresponding web page. Haplin et al. Halpin et al. [2007] studied the tag distribution at del.icio.us and proposed a generative model of collaborative tagging in order to evaluate the dynamics that lie beneath and found out that the data set follows a power-law distribution. Firan et al. Firan et al. [2007] study personalized recommendation of tracks within the popular music portal Last.Fm, and show that tag-based profiles can produce better recommendations than conventional ones based on track usage. Vatturi el al. Vatturi et al. [2008] study personalized bookmark recommendation using a CB approach that leverages tags, assuming that users would be interested in pages annotated with tags similar to ones they have already used.

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 Perugini et al. [2004]. 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 Lueg [1997]. It has been found Sinha and Swearingen [2001] 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 Shardanand and Maes [1995] among customers is by far the best predictor of a companys 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.

Golbeck et al. Golbeck [2006] considered those social networking sites where users explicitly provide trust ratings to other members. However, for large social networks it is infeasible to assign trust ratings to each and every member so they propose an inferring mechanism which would assign binary trust ratings (trustworthy/non-trustworthy) to those who have not been assigned one. However they assume three crucial properties of trust for their approach to work: transitivity, asymmetry, and personalization. This is contrary to what was proposed by Yu et al. Yu and Singh [2003], who assume symmetric trust values in the social network between two members. Since trust is an absolutely a personal opinion, hence, authors proposed personalization of trust which means that a member could have different trust values with respect to different members. Ma et al. .Ma et al. [2009] assumed that every users decisions on the Web should include both the users characteristics and the users trusted friends recommendations. Under this assumption, the authors proposed a probabilistic matrix factorization framework that employs both the user-item matrix and the users social trust network for the recommendations. Walter et al. Walter et al. [2006] proposed the use of social network information in recommendation systems and analyzed the impact of trust dynamics on the performance of such a system. They studied the effect of preference heterogeneity of agents and network density on usefulness of trust in the system. However, their algorithm would not scale well for large networks and large number of items. Moreover, trust according to them is based only on past experience of recommendations and they make some simplifying assumptions like the social network is static and there are not any malicious entities. Massa et al .Massa and Avesani [2007] studied the trust-aware recommender systems. Their work replaces the similarity finding process with the use of a trust metric, which is able to propagate trust over the trust network and to estimate a trust weight. Josang et al. Jøsang et al. [2006] described a method for trust network analysis using subjective logic (TNA-SL). Their method takes directed trust edges between pairs as input, and can derive a level of trust between arbitrary parties that are interconnected through the network. Even in case of no explicit trust paths between two parties exist, subjective logic allows a level of trust to be derived through the default vacuous opinions. TNA-SL therefore has a general applicability and is suitable for many types of trust networks. However, this method includes the same trust edges multiple times and will produce an inconsistent result.In light of these studies, it can be said that the computational trust models can act as appropriate means to supplement current collaborative filtering approaches used by the recommender systems O'Donovan and Smyth [2005].

2.6 Discussion

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 Jennings et al. [2001]. In existing literature, this has usually been accomplished in an ad-hoc manner Jennings et al. [2001]; Resinas et al. [2012], which is of minimal interest in SOAs due to the high developmental costs of such solutions, lack of ubiquity, and dynamic participants. Consequently, the prime requirements for developing comprehensive negotiation mechanisms include:

- *Multi-Term negotiation.* The typical SLA negotiation involves QoS terms like *Reliability*: measures the ability of a service operation to be executed within the expected time. *Availability*: measures the probability that the service operation is operating at any given moment. *Accessibility* measures the degree that the service operation is able to serve the request (i.e. success rate). *Integrity* measures how the service operation maintains the correctness with respect to the source. *Response time* measures the expected delay between the time that the service operation starts and receiver receives the response to name a few Yu et al. [2008] Zarras et al. [2004]. The various Qos attributes discussed above (and others) influence the negotiation protocol and consumer preferences the negotiation system must support. Hence there may be more than one combination of these attributes that would be suitable to the negotiation context. User preferences could be expressed in a variety of ways, e.g. utility functions Faratin et al. [2002], combination of attributes Elfatatry and Layzell [2005], or fuzzy constraints Luo et al. [2003], but multi-term negotiations require the management of expressive SLA preferences regarding the multiple terms to be negotiated (REQ 1). This would allow the system to capture the relationship among different terms and hence facilitate making better trade-offs during the negotiation process.

- *Heterogeneous participants.* In an open system it is very much expected that all the participants using the system may not be similar. They may implementing heterogeneous (probably incompatible) negotiation protocols. Thus, there is a need for supporting multiple negotiation protocols (REQ 2), or be able to consent on the negotiation protocol for cases where a participant supports multiple ones.

- *Heterogeneous user decision models.* Different participants prefer different negotiation strategies based on their decision models, domains, preferences and history. There are usually two types of decisions that an automated negotiation system has to make. First, it has to generate counter offers in the negotiation by implementing an appropriate algorithm Faratin et al. [2002] Faratin et al. [1998] Luo et al. [2003] Kowalczyk [2002]. Second it has to handle commitment to new SLA i.e. deciding if the agreement is acceptable and convenient to commit and in some case decommitment from previously created SLA Nguyen and Jennings [2005]. This decision is mostly protocol independent. However depending upon the negotiation protocol the counter offer generation could be totally different. For instance, in case of a bargaining protocol, there has to be a response for each negotiation message that is received, where as in an auction protocol, bids could be placed at any time. Hence, an automated negotiation system must implement multiple decision models (REQ 3) so that it could support protocol specific negotiations.

- *Heterogeneous user preferences.* Unlike traditional software environments, SOAs enable delivery of the same service to different consumers with varied quality of service (QoS) requirements Elfatatry and Layzell [2005]. Therefore, it is imperative that service agreements include the provision(s) to negotiate over these attributes. Since negotiation is a dynamic and interactive process, the user preferences could change over time. The negotiation system should allow, the user preference about the negotiation process to be changed over time (REQ 4).

- *Simultaneous negotiations.* Since services are not store-able, hence the service market tends to be very dynamic Gimpel et al. [2003] and the ability to create on-the-fly dynamic solutions emphasizes the need of conducting simultaneous negotiation (REQ 5) of multiple component services with different parties at the same time. On one hand it is necessary for the system to have a global view of all the negotiations to support them properly. However since the user preferences could change over time, it is beneficial for the system to guide the behavior of each negotiation based on how well other negotiations are performing. This allows the system to choose the party that would result in the most profitable agreement.

- *Support for dynamic selection of decision making models.* Simultaneous negotiations are desirable in volatile service markets to allow selection of the most profitable agreements for the participants Gimpel et al. [2003]. This entails that the participants are equipped to change their strategies/decisions at run-time (REQ 6), based on market dynamics and changing contexts Ros and Sierra [2006]. The underlying strategy should be robust enough so that it can adapt to different behaviors of participants, and utilize “peripheral knowledge”. For instance, information relating to whether the participant tends to concede, participant reputation, etc. may be used to strengthen one’s negotiation capabilities Nguyen and Jennings [2005, 1998].Similarly, in some contexts if a more profitable offer is found, there should be a provision to decommit from the current agreementSandholm and Lesser [2001].

We surveyed the literature and analyzed the current negotiation systems on the above mentioned requirements. Table 1 summarizes our findings. An '✓' in a cell means that the corresponding proposal provides explicit support for the corresponding requirement, whereas a '✗' indicates that the feature is not supported and 'n/a' means that there is no information available.

Table 2.1 shows that most of the existing solutions do not do well when it comes to supporting multiple negotiations at the same time or dynamic selection of decision making models. None of the surveyed solutions provide any dependency modeling among different QoS components. This is motivated by the fact that composite solutions often have dependent QoS objectives. For example, if we were to have a composite solution consisting of *serviceA* and *serviceB* and one of the objective was to have services that could handle a load of 1 million transactions per minute. What if we have multiple services offering such a solution for *serviceA* but could not find any service similar to *serviceB* that could meet our current objective. It would then be more economical for the composite solution to downgrade *serviceA* to the level of *serviceB's* solution (since throughput of a system is a composite function of its constituent services). Continuing with this hypothetical scenario, we need the negotiation service to be able to simultaneously negotiate multiple services having multiple objectives with multiple providers. Existing communication protocols [GRAAP] Smith [1980] Lecue [2009] [W3C] Andreoli and Castellani [2001] lack such capabilities. This requires a new standard language that could be used to pass on all these constraints and decision model to the negotiation system.

This leads us to look for a new solution that not only fairs better in comparison with the existing solutions, but also supports all the requirements of a SOA based negotiation system.

Authors	**REQ 1**	**REQ 2**	**REQ 3**	**REQ 4**	**REQ 5**	**REQ 6**
Ashri et al.Ashri et al. [2003]	✓	✓	n/a	n/a	n/a	n/a
Bartolini et al.Bartolini et al. [2004]	✓	✓	✗	✗	✗	✗
PANDAGimpel et al. [2003]	✓	✓	✓	✓	✗	✗
Jonker et al.Jonker et al. [2007]	✓	n/a	✓	✓	✗	✗
Kim et al.Kim and Segev [2005]	✓	✓	✗	✗	✗	✗
Benyoucef et al.Benyoucef and Verrons [2008]	n/a	✓	✓	n/a	✗	✗
Ludwig et al.Ludwig et al. [2005]	✓	✓	✓	✗	✗	✗
Paurobally et al.Paurobally et al. [2007a]	✓	✗	✗	✗	✗	✗
Rinderle et al.Rinderle and Benyoucef [2005]	✓	✓	✗	✗	✗	✗
Strobel Strobel [2001]	✓	✓	✗	✗	✗	✗
DynamiCS Tu et al. [2001]	✓	✓	✓	✗	✗	✗
NegotiatorBui and Shakun [1996]	✓	✗	✓	✗	✗	✗
Bruns et. al Bruns and Cortes [2011]	✓	✗	✓	n/a	✓	✗
InspireKersten and Noronha [1999]	✓	✗	✓	✗	n/a	✗
NegoPlanS. Matwin and Koperczak [1996]	✓	✗	✓	✗	✗	✗
KasbahChavez and Maes [1996]	✓	✓	✓	✗	✗	✗
Tete-a-TeteLab [2000]	✓	✓	✓	✗	✗	✗
AuctionBotWurman et al. [1998]	✗	✗	✓	✓	✗	✗
MAGNETJaiswal et al. [2004]	✓	✗	✓	✓	✗	✗
CremonaLudwig et al. [2004]	✓	✗	✓	✓	✗	n/a
Lecue at al.Lecue [2009]	✗	✓	✗	✗	✗	✓
Marco at al.Comuzzi and Pernici [2005]	✓	✓	✗	✓	✗	✗

Table 2.1: Summary of Automated Negotiation Systems

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" Coello et al. [2007]. It is well understood and accepted that 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 Siddiqui et al. [2006] .Shang and Wah [1998] .Csirik and Woeginger [1997] Kraus [2001]. Moreover, it has been shown that a number of natural variations of this problem are NP-hard, and determining whether a particular allocation is Pareto Optimal is co-NP-completeDunne et al. [2005] . One such variation is the focus of our work where we model the dependencies among the multiple attributes being negotiated. 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 Siddiqui et al. [2006] Nisan et al. [2007] Sandholm [2002] Sakurai et al. [2000].

The reason for using GA in our work is the fact that our multi-attribute negotiation scenario falls under the "incomplete information model" class of problems. Participating services have two options when it comes to evaluating the offers. They can either provide their decision model or can provide their own negotiation module. If a service chooses to opt for the latter case i.e. provide its negotiation component, each offer would then be evaluated by the participant and it is possible that the participant may evolve its decision model based on the past offers and counter offers. This would mean that the search space exploration would become complex. GAs are a good option for such dynamic environment since it is comparatively easier to find an acceptable solution. Second, for dependency modeling we are using knowledge sources i.e. Norms. These knowledge sources are effected by the offers and counter offers of participants along with their dynamic evaluation of offers. This learning process, in turn effects the exploration of search space and bounds the solution. To the best of our knowledge, Genetic Algorithms are a good option for such complex scenarios.

We can find many examples in literature where Genetic Algorithms (GA) have been used to enhance automated negotiation. GAs are used to evolve the best strategies Matos et al. [1998], generate proposals at every round Sim et al. [2007] Sim et al. [2009], track shifting tactics and changing behaviors Matos et al. [1998][25], and learn effective rules for supporting negotiation Matwin et al. [1991].

As mentioned above, due to these benefits researchers have had much success in applying genetic algorithms (GAs) to optimization problems. They are capable of finding solutions for complex problems that cannot be solved by more traditional approaches Michalewicz and Attia [1994]. Some examples include Soremekun et al. Soremekun et al. [2001] and Andrzej and Stanislaw Andrzej and Stanislaw [2000]. Constraints on these problems have been handled by penalty functions Huo et al. [1999], Li and Gen [1996], hard constraints rejecting anything infeasible Fogel and Stayton [1994], Gen and Cheng [1997], decoders Huo et al. [1999], repair algorithms Gen and Cheng [1997], Shimizu [1999], Todd and Sen [1997], special genetic operators to keep solutions in the feasible region Qi et al. [2000], Yuping et al. [2000], and simulated annealing Gallege. et al. [1998], Michalewicz [1995].

Since the introduction of multi-objective problems Schaffer [1985] multi-criterion approaches have been reported in the evolutionary algorithm literature by such researchers. Some examples include Matwin et al. Matwin et al. [1991], Fonseca and Fleming Fonseca and Fleming Fonseca and Fleming [1995] Fonseca and Fleming [1998], Horn and Nafpliotis Horn et al. [1993], Horn et al. Horn et al. [1994], Srinivas and Deb Deb [1994], Osycyka and Kundu Osyczka and Kundu [1996], Murata et al. Murata et al. [1996], Viennet et al. Viennet et al. [1996], Ishibuchi and Murata Ishibuchi and Murata [1998], Van Veldhuizen Veldhuizen and Allen [1999], and Zitzler and Thiele Zitzler and Thiele [1999]. Just as GAs have been shown to be useful for optimization applications in general, evolutionary approaches enjoy widespread support as one of the effective technique for solving multi-objective optimization problems that are too complex to be solved by more traditional methods Fonseca and Fleming, Zitzler et al. [2000]. The conflicting objectives and resultant trade-offs that characterize multi-criterion problems necessitate the generation of a set of optimal solutions rather than a single best solution Deb and Horn [2000].

CHAPTER 3 : AUTOMATED NEGOTIATION FRAMEWORK

3.1 Introduction

In recent years, information system design has been influenced by the service oriented paradigm to facilitate easy integration between organizations that provide their services on the Web Alonso et al. [2004]. With the increasing agreement on the functional aspects of Web services (e.g. using WSDL Booth and Liu [2006] for service description, SOAP Standard [2007] 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 Papazoglou and Heuvel [2007]. Service selection (thereby composition) can be defined as a multi-stage process that ranges from finding functional equivalence, negotiating consumer and provider preferences, to finally creating an agreement. Currently, this selection process is very tedious since it involves human intervention for negotiating and making decisions regarding consumer and provider preferences. Service based information systems likely span across multiple enterprize boundaries where different providers exercise control over their propriety service(s), and certain limitations are put on the different Quality of Service(QoS) attributes such as availability, reliability, scalability etc. of otherwise functionally equivalent services. Since, 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 up them for negotiation.

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. 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 Jennings et al. [2001]. In existing literature, this has usually been accomplished in an ad-hoc manner Jennings et al. [2001] Resinas et al. [2012], which is of minimal interest in SOAs due to the high developmental costs of such solutions, lack of ubiquity, and dynamic participants. Consequently, the prime requirements for developing comprehensive negotiation mechanisms include:

- *Multi-attribute negotiation.* A typical Service Level Agreement (SLA) negotiation involves multiple QoS components (e.g. reliability, availability, accessibility, response time etc.) Yu et al. [2008] Zarras et al. [2004]. There may be more than one combination of these attributes that may be suitable in a specific negotiation context. Thus the negotiation system should allow the users to express multiple attributes for the negotiation process (REQ 1).

- *Heterogeneous negotiation protocols.* In a service oriented information systems, it is expected that the participants may implement heterogeneous (probably incompatible) negotiation protocols. Thus, there is a need for supporting multiple negotiation protocols (REQ 2), or be able to consent on a common negotiation protocol for cases where a participant supports more than one protocol.

- *Heterogeneous decision model.* Different participants prefer different negotiation strategies(auction, bargaining etc.) based on their decision models, domains, preferences and history. There are usually two types of decisions that an automated negotiation system has to make. First, it has to generate counter offers in the negotiation by implementing an appropriate algorithm Faratin et al.

[2002] Faratin et al. [1998] Luo et al. [2003] Kowalczyk [2002]. Second, it has to handle commitment to the new SLA i.e. deciding if the agreement is acceptable and convenient to commit, and in some cases de-commit from previously created SLA Nguyen and Jennings [2005]. This decision is mostly protocol independent. However, depending upon the negotiation strategy the counter offer generation could be totally different. For instance, in case of a bargaining strategy, there has to be a response for each negotiation message that is received, where as in an auction strategy, bids could be placed at any time. Hence, an automated negotiation system must implement multiple decision models (REQ 3) so that it could support protocol specific negotiations.

- *Dynamic user preferences.* Since negotiation is a dynamic and interactive process, user preferences could change over time (during the negotiation process). For instance, the user may change the required value of a QoS attribute during the negotiation process, as it learns new information during the negotiation or may even add or remove new QoS attributes. This allows the system to adapt to the counter offers presented during the negotiation process and adapt a better solution for negotiation. Thus, the negotiation system should allow the user preference about the negotiation process to be changed over time (REQ 4).

- *Simultaneous negotiations.* The ability to create on-the-fly dynamic solutions in SOAs emphasizes the need of conducting simultaneous negotiations (REQ 5) with multiple component services, owned by different parties, at the same time. On one hand, it is necessary for the system to have a global view of all these negotiations to support them properly. However, since the preferences of the parties involved in the negotiation could potentially change, it is beneficial for the system to guide the behavior of each negotiation based on the responses generated by other (simultaneous) negotiations. This allows the system to choose the party that would result in the most profitable agreement.

- *Support for dynamic selection of decision making models.* Simultaneous negotiations are desirable in volatile service markets to allow selection of the most profitable agreements for the participants Gimpel et al. [2003]. This entails that the participants are equipped to change their strategies/decisions at runtime (REQ 6), based on market dynamics and changing contexts Ros and Sierra [2006]. The underlying strategy should be robust enough so that it can adapt to different behaviors of participants, and utilize "peripheral knowledge". For instance, information relating to whether the participant tends to concede, participant reputation, etc. may be used to strengthen one's negotiation capabilities Nguyen and Jennings [2005] Nguyen and Jennings [1998].Similarly, in some contexts if a more profitable offer is found, there should be a provision to decommit from the current agreementSandholm and Lesser [2001].

- *Dependency modeling for multi-service negotiation.*Dynamic service selection is mostly used in two scenarios. One to replace a faulty service in the system and secondly when searching for component services for a newly formulated solution. In the later case we can safely assume that the system would be composed of more than one service(s) and hence it may need to simultaneously negotiate multiple services. Certain system properties are a composite function of its component services e.g the overall throughput of the system is limited by its component service 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 component services (REQ 7). This would allow the system to capture relationships among different attributes and hence facilitate making better tradeoffs during the negotiation process.

This chapter presents *WebNeg*, a framework for a negotiation service that is designed to meet the above mentioned requirements. We assume an environment with simultaneous negotiations among multiple providers, where each communication instance (among the consumer and service provider) is private. We enhance the traditional Genetic Algorithm with a new operator called *Norm*, that makes it possible to share "private", negotiation-related information among all participating Web services without revealing the source of the information. This enables 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. Moreover, *WebNeg* provides an approach of QoS attributes dependency modeling that optimizes the solution when more than one component services are being simultaneously negotiated.

3.2 A framework for Web Service negotiation

Automated SOA-based interactions entail that Web service consumers can dynamically locate the service providers, consent on the terms and conditions of the invocation, and execute the necessary actions on the basis of the negotiated service level agreement. This contractual agreement between the two parties specifies common arrangements and expectations on functional and non-functional requirements (cost, reliability, availability etc.). The non-functional requirements are often not as strict as the functional requirements and can be negotiated for an optimal solution.

Figure 3.1 shows a typical Web service interaction model from the point of view of customer and provider. Service providers offer their services by publishing their information (WSDL) in public directories (UDDI). Customers then query these public directories to find similar service and then bind to the most suitable service. These directories serve as place holders and provide minimal functional information about the service. Service providers may use *tModels* Curbera et al. [2002a] to provide any additional information. These *tModels* have limited usability when it comes to negotiating non-functional components. A customer looking for a Web service could benefit from a service that has published 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 3.2 shows the high level state diagram of the negotiation process. Once the customer initiates the negotiation request, the system goes into the proposal and response state. From a customer's point of view the system is in proposal state once either he has generated the initial offer or is in the process of producing a counter offer. Similarly, once the customer has produced an offer and is now waiting for the producer to respond, the system is in the response state. During the negotiation process when the provider accepts the offer presented by customer the system goes

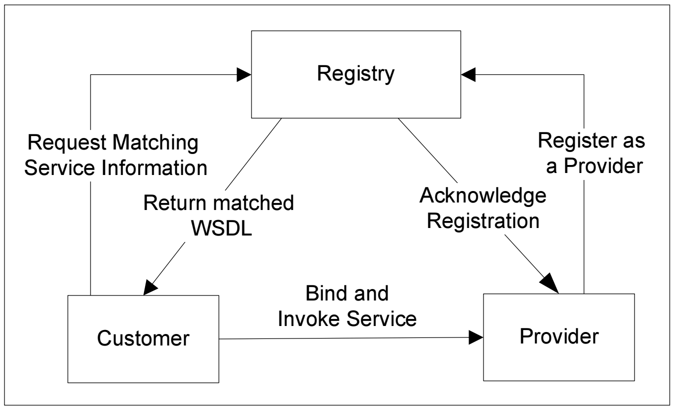


Figure 3.1: Web service Interaction Model

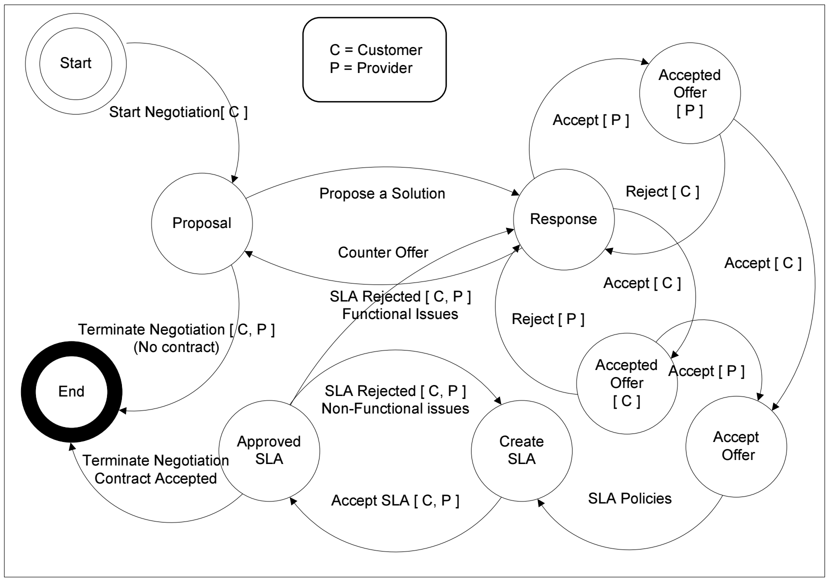


Figure 3.2: Negotiation State Chart

into the Accepted Offer by provider state. It works as a two phase commit and seeks confirmation from the customer on the initial offering. Hence an offer is not binding until it has been confirmed as accepted by both the participants. The same process will be repeated if the offer is accepted by the customer. Once both the parties agree on the operational issues and we reach the Accept Offer state the next step is to articulate the SLA policies from both the participants. This process could also be treated as the negotiation of SLA. In scenarios where participants agree on the SLA terms the system enters the Approved SLA state and negotiation process terminates successfully with an established contract. However, if the parties fail to agree on SLA terms, the system then goes back to the initial negotiation states of proposal and response.

We present the high level architecture of *WebNeg* in Figure 3.3. The proposed model is very flexible in terms of its functionality. It is primarily targeted to be

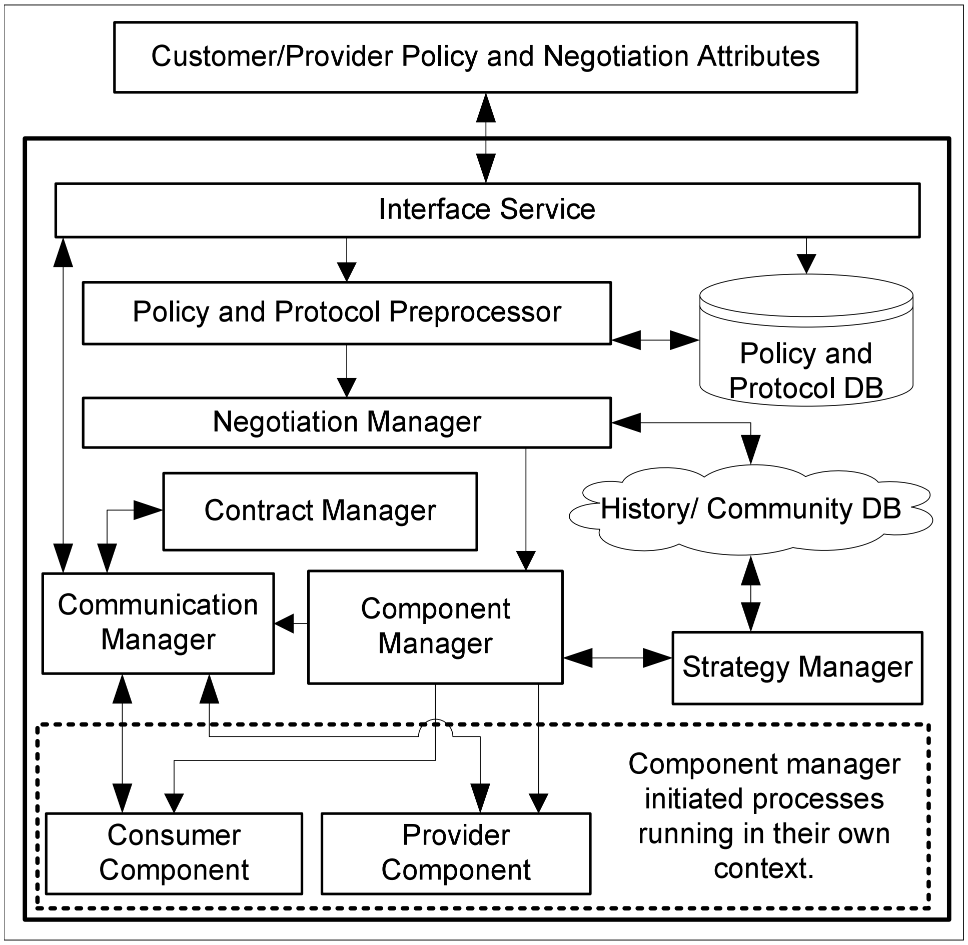


Figure 3.3: WebNeg System Architecture

invoked by a consumer searching among a list of service providers providing similar functionalities. The consumer does not need to implement any negotiation specific component to use the proposed service. The architecture presented is compatible with two negotiation scenarios, (1) the negotiating participants provide their own negotiation component, or (2) send all the necessary information to the service, that handles all the negotiation process. A brief overview of the major modules of the *WebNeg* architecture is presented below and performance assessment details are presented in section 3.4.

3.2.1 Interface Service

The *Interface Service* layer acts as the interface of the whole system. This layer is responsible for all external system communication. The internal components use this service to communicate with both the consumer and provider as well as with other peers within the community (to be discussed later). As mentioned above, a consumer can invoke the *WebNeg* by providing its negotiation attributes (e.g. availability, reliability), negotiation policy as well its decision model. The interface service then communicates with potential providers and requests their decision model and policy attributes for the negotiation process.

3.2.2 Policy and Protocol Preprocessor

Different participants may use different protocols for describing their decision models and policy attributes. This generic module ensures that the system is extendable to incorporate future communication protocols and allows the participants using different communication protocols to still be able to negotiate service attributes and form service level agreements (SLA). This component is responsible for standardizing the inputs from the communicating participants. After receiving this data from the interface 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 participant preferences in the *Policy and Protocol Database*. *WebNeg* then uses this information for future communication with the participants. This component also allows the system to handle various domain specific constraints e.g. it may be used to specify penalties in case of contract violations using policy specific languages such as WS-Policy [W3C], etc.

3.2.3 Negotiation Manager

Figure 3.4 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.

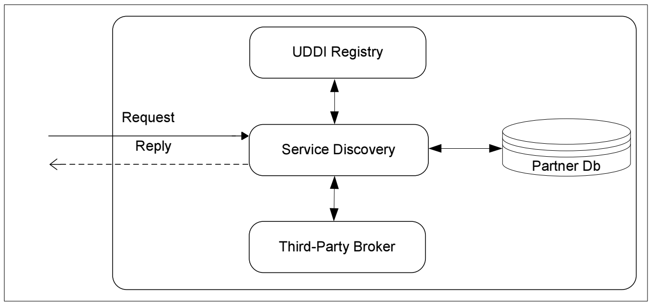


Figure 3.4: Negotiation Manager

It uses the trust model based on the concept of community Malik and Bouguettaya [2007] 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 Malik and Bouguettaya [2009b]. 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.

3.2.4 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 3.5 depicts the architecture of the strategy manager.

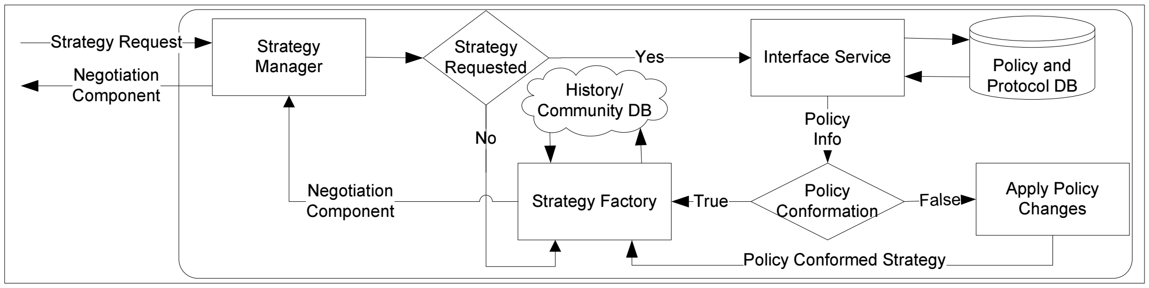


Figure 3.5: Negotiation Strategy Manager

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.

3.2.5 Component Manager

Since negotiation is a multi-party mechanism, *WebNeg* needs to spawn separate components for each consumer and provider Web service. In the most basic scenario, the system would have one consumer and multiple provider components (Req 5). These components though being instantiated as child processes of *WebNeg* operate in their separate contexts and communicate with their respective service through the *Communication Manager*. The *Component Manager* is responsible for creating and managing these components and ensures that the private information provided by the negotiating services is kept isolated. The architecture is flexible enough to let the participants provide their own negotiation components in cases where, the participants believe that they do not want to share any of their private information with even a trusted third party broker or the participants may want more control over the offer/counter offer process (Req 4).

3.2.6 Communication Manager

All the external pre-SLA communications are handled by this component. The component may communicate with its respective service for any decision model or policy/guidance queries. The *Communication Manager* ensures that all the communication is related to the current negotiation and adheres to the negotiation service's policies. Figure 3.6 shows the architecture of *Communication Manager*. On receiving a communication message, the component requests policy information from *Policy and Protocol DB* and checks the conformity of the message with the retrieved information. If the message does not conform to the policies, it goes through a policy conformation change. After conforming with the policies the message goes through translation (if needed) before being delivered to its destination.

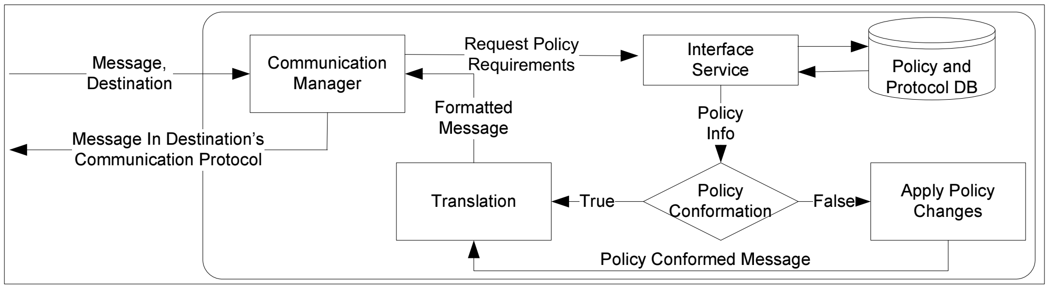


Figure 3.6: Communication Manager

3.2.7 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 3.7. Similar to the other modules, any SLA generation must conform to the system policies. Upon receiving a proposed solution, the 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 moves to the next best available solution, until either an agreement is achieved or the system runs out of options. If a mutually agreeable solution is not found, the process is termed as a failure and the consumer is asked to revise its negotiation model.

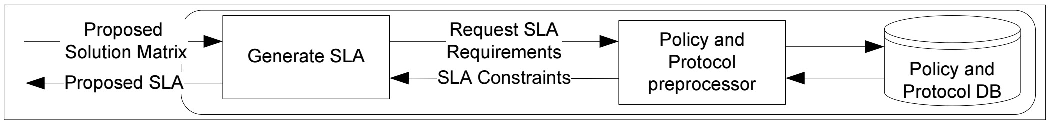


Figure 3.7: Contract Manager

3.2.8 System Engineering View

The development view of the *WebNeg* architecture focuses on the organization of the modules mentioned above. A part of this view is presented as the architecture of the software framework that serves as a guideline for the developers. Figure 3.8 provides the data model used in the implementation of *WebNeg*. 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 3.9 shows different interfaces used by the *WebNeg* system to communicate with different components within the systems as well as with other systems. Negotiation messages are composed of a URL that identifies 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 is responsible of the communication protocol conversion and maintains the overall state of the system including the SLA generation.

Figure 3.10 describes two implementations of *WebNeg*. In the first implementation (denoted by steps 1, 2, ..), both the consumer and provider prefer using the negotiation component of *WebNeg*. Once the client is bound to an implementation of the *WebNeg* functionality, it sends an aggregated request containing all the data needed by the other functionalities including the choice of negotiation strategy and the optional list of preferred providers. It is important to point out that the sequence diagram of Figure 3.10 is only an example of a possible implementation of the proposed framework. As previously mentioned, the framework describes a modular

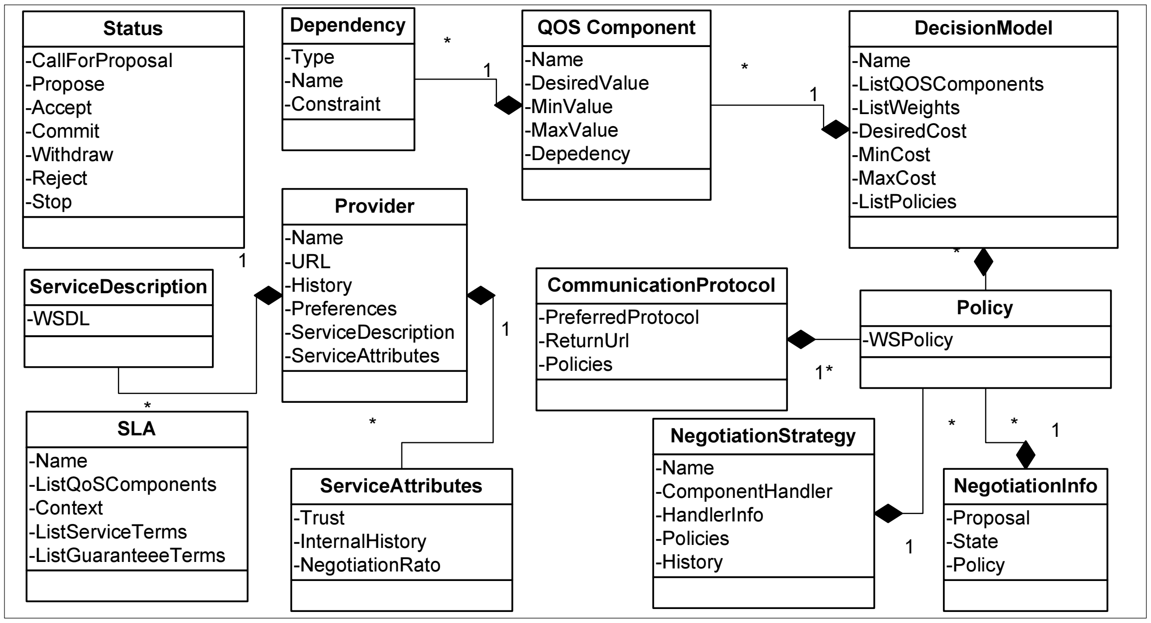


Figure 3.8: WebNeg Data Model

environment that system engineers can rely on to specify their own modes of negotiation (e.g. auction, bilateral negotiation etc.), but it does not dictate the technical details of specific implementations. In other words, the services are stateless and exchanges are limited to request-response pairs in synchronous interactions (this is why the consumer has to submit all the needed information in its initial request).

In the second implementation (denoted by steps A, B, .. in Figure 3.10), both the consumer and provider prefer to provide their own negotiation components. This could be attributed to the fact that a participant may not wish to share its private information i.e. decision model, with any one else. This implementation uses only a subset of the Web services and relies on messages within the context of negotiation. In this case, exchanges can include asynchronous notifications. Even if they are quite different from a technical point of view, both implementation versions illustrate our approach. This demonstrates the modular and flexible nature of *WebNeg*.

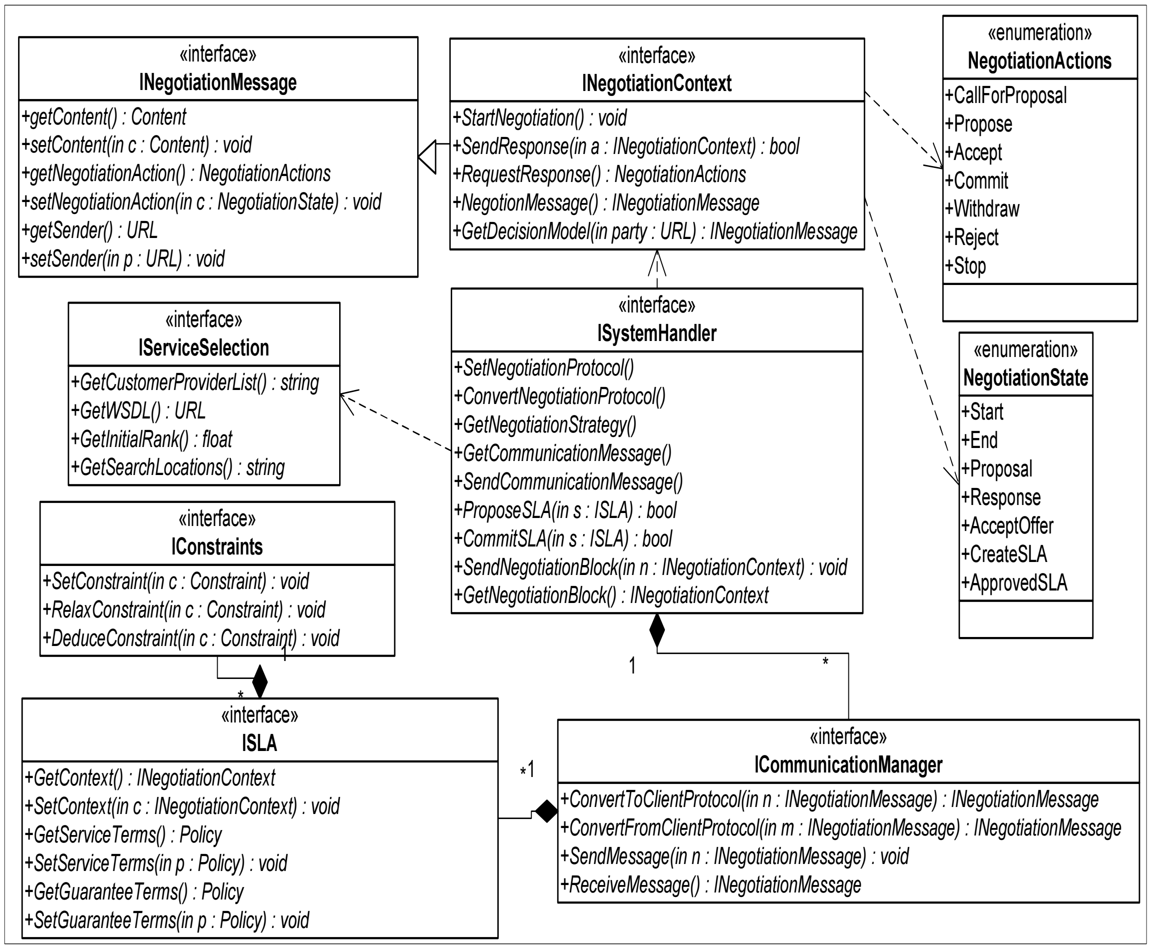

«interface»
INegotiationMessage
+getContent() : Content
+setContent(in c : Content) : void
+getNegotiationAction() : NegotiationActions
+setNegotiationAction(in c : NegotiationState) : void
+getSender() : URL
+setSender(in p : URL) : void
«interface»
INegotiationContext
+StartNegotiation() : void
+SendResponse(in a : INegotiationContext) : bool
+RequestResponse() : NegotiationActions
+NegotionMessage() : INegotiationMessage
+GetDecisionModel(in party : URL) : INegotiationMessage
«enumeration»
NegotiationActions
+CallForProposal
+Propose
+Accept
+Commit
+Withdraw
+Reject
+Stop
«interface»
IServiceSelection
+GetCustomerProviderList() : string
+GetWSDL() : URL
+GetInitialRank() : float
+GetSearchLocations() : string
«interface»
ISystemHandler
+SetNegotiationProtocol()
+ConvertNegotiationProtocol()
+GetNegotiationStrategy()
+GetCommunicationMessage()
+SendCommunicationMessage()
+ProposeSLA(in s : ISLA) : bool
+CommitSLA(in s : ISLA) : bool
+SendNegotiationBlock(in n : INegotiationContext) : void
+GetNegotiationBlock() : INegotiationContext
«enumeration»
NegotiationState
+Start
+End
+Proposal
+Response
+AcceptOffer
+CreateSLA
+ApprovedSLA
«interface»
IConstraints
+SetConstraint(in c : Constraint) : void
+RelaxConstraint(in c : Constraint) : void
+DeduceConstraint(in c : Constraint) : void
1
*
1
*
«interface»
ISLA
+GetContext() : INegotiationContext
+SetContext(in c : INegotiationContext) : void
+GetServiceTerms() : Policy
+SetServiceTerms(in p : Policy) : void
+GetGuaranteeTerms() : Policy
+SetGuaranteeTerms(in p : Policy) : void
*1
«interface»
ICommunicationManager
+ConvertToClientProtocol(in n : INegotiationMessage) : INegotiationMessage
+ConvertFromClientProtocol(in m : INegotiationMessage) : INegotiationMessage
+SendMessage(in n : INegotiationMessage) : void
+ReceiveMessage() : INegotiationMessage

Figure 3.9: System Interface Diagram

In the first scenario if the participating service choose to provide their individual negotiation component, the proposed system only has access to the offers provided in each round and no other piece of information is provided. Hence this makes it a very hard problem to solve. The system needs to explore multiple local maxima in parallel, to find a feasible solution. The system learns the probable decision making factors and records this learning in the form of Norm, which in turn influences the solution for the next round of negotiations. It is to be noted that *WebNeg* does not favor any particular negotiation methodology. We have chosen GA as of the negotiation methodology to be used for *WebNeg*. We feel that even though GA has a slightly

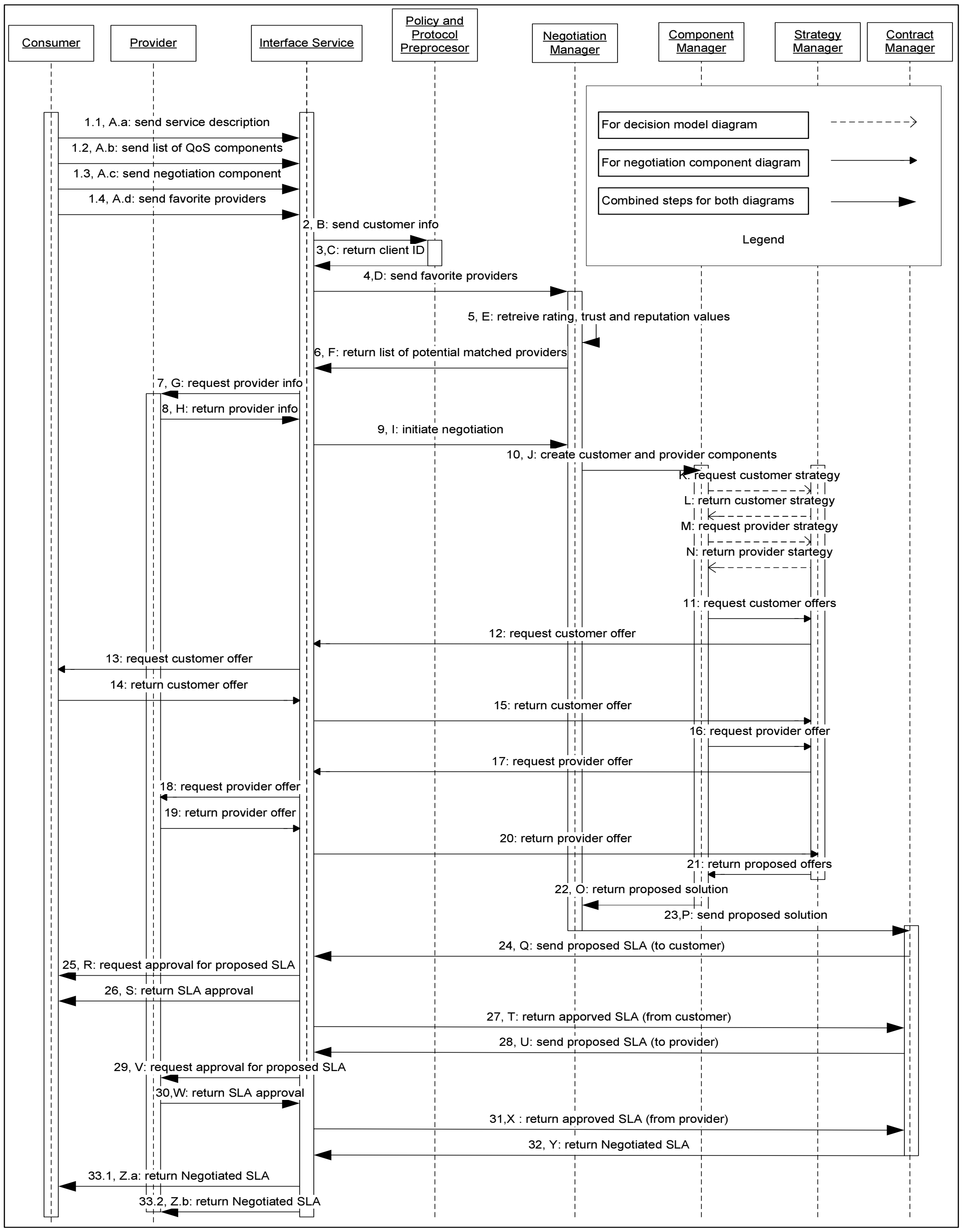

Consumer
Provider
Interface Service
Policy and Protocol Preprocesor
Negotiation Manager
Component Manager
Strategy Manager
Contract Manager
For decision model diagram
For negotiation component diagram
Combined steps for both diagrams
Legend
1.1, A.a: send service description
1.2, A.b: send list of QoS components
1.3, A.c: send negotiation component
1.4, A.d: send favorite providers
2, B: send customer info
3,C: return client ID
4,D: send favorite providers
5, E: retreive rating, trust and reputation values
6, F: return list of potential matched providers
7, G: request provider info
8, H: return provider info
9, I: initiate negotiation
10, J: create customer and provider components
K: request customer strategy
L: return customer strategy
M: request provider strategy
N: return provider startegy
11: request customer offers
12: request customer offer
13: request customer offer
14: return customer offer
15: return customer offer
16: request provider offer
17: request provider offer
18: request provider offer
19: return provider offer
20: return provider offer
21: return proposed offers
22, O: return proposed solution
23,P: send proposed solution
24, Q: send proposed SLA (to customer)
25, R: request approval for proposed SLA
26, S: return SLA approval
27, T: return apporved SLA (from customer)
28, U: send proposed SLA (to provider)
29, V: request approval for proposed SLA
30,W: return SLA approval
31,X : return approved SLA (from provider)
32, Y: return Negotiated SLA
33.1, Z.a: return Negotiated SLA
33.2, Z.b: return Negotiated SLA

Figure 3.10: Combined Sequence Diagram: Consumer and provider provide 1: Decision models , A: Negotiation components

higher computational cost, it is a good starting point for the problem at hand considering the incomplete information and the gradual discovery of information on each round of negotiation. Similarly, when the services do provide their decision models (case 2), the system spawns independent negotiation components for each service so that the private information about the decision models is not shared by the process and only Norm is used to guide the search process.

3.2.9 Service Selection in WebNeg

In *WebNeg*, we create clusters of prospective services based on different quality parameters. These clusters are then ranked according to their utility rank. We start with the highest ranked cluster and negotiate the service agreements. If a suitable match is not found, the process moves to the next cluster. In addition to the cluster being considered for negotiation, we include the consumer's preferred providers list for negotiation which may include providers with business ties to the consumer (e.g. business partners, preferred providers and previously used/contacted vendors, etc). For each service we retrieve the following vector.

$$SelectV_i = \{Trust_i, InternalHistory_i, NegRatio_i\} \tag{3.1}$$

where *Trust* rating for a service is defined as the degree of confidence in the ability of the provider to deliver the promised QoS. Here, we are using our approach presented in Malik and Bouguettaya [2007], to calculate the trust values for all the participating services. We have also defined a bootstrapping approach for new services or service with limited or no transaction history in Malik and Bouguettaya [2009b].

InternalHistory is the combination of length of time that the service has been operational, total number of compositions participated in, and the ratings provided by orchestrators of those compositions. *NegotiationRatio* is defined as the success ratio of the service in forming a contract. It is calculated as the ratio of number of times a service was contacted for negotiation, over the number of times the service successfully formed an agreement.

Once we have matrix of n services with these attributes, the first step is to normalize these values as follows (Symbols defined in Table 2),

$$Z_{ij} = \frac{X_{ij} - \bar{X}_i}{S_i} \; where \; \bar{X}_i = \frac{\sum_{j=1}^{t}(X_{ij})}{t} \tag{3.2}$$

$$S_i = \left(\frac{\sum_{j=1}^{t}(X_{ij} - \bar{X}_i)^2}{t-1} \right)^{1/2} \tag{3.3}$$

This gives us the clusters of services. Based on these clusters we initiate the negotiation process.

3.3 WebNeg Negotiation Methodology

WebNeg uses a weighted sum genetic algorithm to support multi-party multi-objective negotiation. All the Web services provide their respective QoS components to be negotiated. These are called the component vector of a Web service. Each vector is accompanied by a decision model, that include the ranges of all the QoS components.

We assume that all the participating Web services are able to articulate their objectives and prioritize them Ackoff [1978]. Table 3.2 lists the definition of symbols used henceforth.

Table 3.2: Definition of Symbols

Symbol	Definition
f_j	Fitness of the solution for participant j.
F_s	Fitness of the solution s (for all participants).
C_j	The value of jth component of Consumer's vector.
$C_{j(min)}$	The minimum allowed value of jth component of Consumer's vector.
$C_{j(max)}$	The maximum allowed value of jth component of Consumer's vector.
WC_j	The weight of jth component of Consumer's vector.
P_{ij}	The value of jth component of ith Provider's vector.
$P_{ij(min)}$	The minimum allowed value of jth component of ith Provider's vector.
$P_{ij(max)}$	The maximum allowed value of jth component of ith Provider's vector.
WP_{ij}	The weight of jth component of ith Provider.
R_j	Rank for solution j in the system.
N_i	Value of Norm i in the system.
E_{ij}	The willingness of participant j to exchange objective i
A_{ij}	Amount of resource i exchanged by Web service j
G	Total number of generations.
$CrossP_j$	Cross over probability for service j.
$AugVal_{ij}$	The value of ith objective to be added or subtracted for Web service j.
G_{ICheck}	The value of generation when interval window is calculated.
S_{Window}	The value of sliding window.
$SelectV_i$	The initial service selection vector.
$QoS_{a,b,c}$	The quality of service attribute vector with attributes a , b and c.
X_{ij}	Initial QoS attribute value for jth providers ith attribute.
Z_{ij}	Standardized QoS attribute value for jth providers ith attribute.
$\bar{X}_i$	Standardized QoS attribute value for jth providers ith attribute.
S_i	Standard deviation of ith attribute value.
$Prevf_j$	Fitness values of the jth participant in the previous generation.
D_{ij}	Distance among the ith QoS attribute value and ith Norm value for jth participant.
V_{Ci}	Value of dependent ith Norm of the component C of the system.

Since all the participating Web services start negotiation from different positions, they have different preferences for their objectives. Their sates are described by how far their current position is from the consumer's objective. All Web services conform to some constraints in the solution. First a QoS component cannot have negative values (Equation 3.4). Second the QoS values lie between the maximum and minimum allowable values set by the consumer Web service (Equation 3.5). A repair algorithm is applied to GA after each operator, to ensure all these constraints are met.

$$C_j \in \mathbb{Q}{>}\ 0\ ,\ P_{ij} \in \mathbb{Q}{\geq}\ 0, \tag{3.4}$$

$$C_{j(min)} \leq C_j \leq C_{j(max)}\ and\ P_{ij(min)} \leq P_{ij} \leq P_{ij(max)} \tag{3.5}$$

Each chromosome is a combination of consumer and provider genes. If we have n QoS components 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 QoS components. We use a distance function to measure the difference between the consumer and provider Web services. Thus, lower fitness values are desired as they translate to more agreement among the proposed solutions of both 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 Baker [1985]Whitley [1989]. Each solution represents a probable distribution of values that may be agreed upon by the other Web services in the negotiation. The fitness value of a solution is calculated as follows.

$$\Delta_{ij} = \frac{|C_j - P_{ij}|}{C_j} \tag{3.6}$$

$$f_i = \sum_{k=0}^{n} (WC_k * \Delta_{ik} + WP_{ik} * \Delta_{ik}) \tag{3.7}$$

$$F_s = \min_{0 \le i \le G} (f_i) \tag{3.8}$$

3.3.1 Pareto Optimality

A solution $< [\overrightarrow{s_i}], [\overrightarrow{c_i}] > (1 \le i \le I)$ Pareto dominates a solution $< [\overrightarrow{s_i}'], [\overrightarrow{c_i}'] >$ iff

$$\forall\, 1 \le i \le I \quad : \overrightarrow{c_i} \succeq_i \overrightarrow{c_i}' and$$

$$\exists g, 1 \le g \le I \;:\quad \overrightarrow{c_g} \succ_g \overrightarrow{c_g}'$$

That is, all participants prefer their solution vector $\overrightarrow{c_i}$ *to* $\overrightarrow{c_i}'(\overrightarrow{c_i} \succeq_i \overrightarrow{c_i}')$ *and at least one of them (i.e. participant g) strictly prefers* $\overrightarrow{c_g}$ *to* $\overrightarrow{c_g}'(\overrightarrow{c_g} \succ_g \overrightarrow{c_g}')$. *A solution is Pareto optimal if it is not Pareto dominated by any other solution.*

In the context of Web service negotiation, Pareto optimality is defined as the situation in which the profit of one party cannot be increased without reducing the profit of another Web service. Pareto optimality is an integral part of multi-criteria optimization and its importance has been widely recognized Veldhuizen and Lamont [2000]. Negotiation problems require that a set of non-dominated solutions be returned by the technique being used for which no objective can be improved without detracting from at least one other objective. Research in the group decision-making, conflict-resolution, and negotiation-theory literature indicates that individuals will not accept solutions that improve the position of other participants while degrading their own position Kersten [1985] Hashmi et al. [2011]. Thus, the Pareto criterion is necessary to ensure that all participants find themselves in positions at least as good as when the decision making process began or they may leave the negotiation process without reaching a mutual agreement.

However, this notion of Pareto optimality alone is insufficient for real-world problems because it does not consider how objectives are prioritized. Preference articulation is used to measure tradeoffs between different objectives. Veldhuizen and Lamont [2000] provides a description of different types of preference articulation: a priori, progressive, and a posteriori. The GA presented in this chapter implements *a priori* preference articulation, where *a priori* preferences are used to construct a weighted sum objective function (fitness function) prior to the optimization. The fact that *WebNeg* has the decision model and preference information for all the participants, it can compare different solutions.

Pareto optimality is not enforced after each generation in the current solution, 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 and updated after each iteration. This secondary population or *Elitism* is a an important concept in genetic searches Baker [1985] and is used in *WebNeg*.

3.3.2 Norm Operator

A new operator *Norm* is implemented to improve the performance of GA and to simulate the exchange of resources based on the common knowledge of the society in a negotiation scenario. The *Norm* operator is based on the observation that in each society people follow certain trends or norms to conduct negotiations. These norms are either informed by the environment or are discovered by the population based on the prior experiences. These norms are transferred through generations and different people follow different norms. Often people are inclined to follow a new norm if they think it will benefit them or abandon a norm if they feel they not being benefited form it. Most helpful norms tend to attract more followers, which in turn re-enforces these norms. People tend to abandon less useful norms in the favor of useful ones. Once in a while people just hop around trying to find out the norm that works the best for them. These norms serve as a guide for achieving their desired goals. Assume a society with n norms 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. Population in Set 1 is effected by the values of *Norm 1* and they in turn effect the values of the *Norm*.

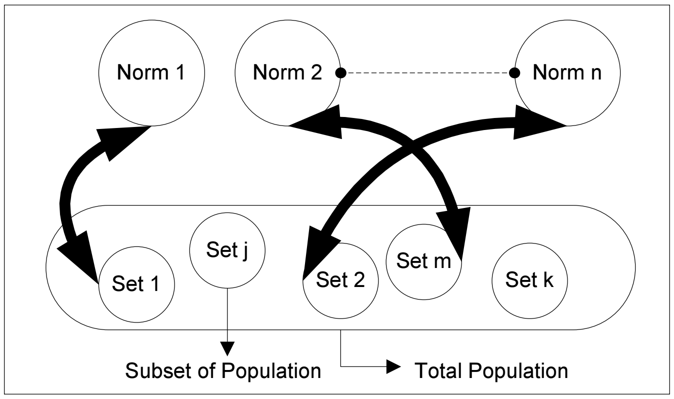


Figure 3.11: Norm operator in relation to population sets

We have the *Norm* operator behavior defined above in the GA, so that it takes less time to find the solution and to reduce the search space. Each QoS negotiation component is represented as a norm and certain members of the population follow a certain norm. After each generation, the followers update the values of their respective norm. If increasing the value of the norm resulted in a better overall fitness value for the follower, it would influence the norm into increasing its value. The increase is dependent on the difference between current and previous values of the objective of the reporting follower and the current norm value of that objective. Both consumers and providers share the same values of norms. Hence norm values act as an indirect information source for the consumer about providers decision model and vice versa. Ideally, one consumer and n providers are involved, so sharing these values do not reveal any private information. These norm values have the bias of n+1 agents and are averaged out.

We implemented *Norm* for the exchange of objectives among different participants. Exchange must occur between two distinct objectives. Participants can trade some or all of their available objectives. There is at most one exchange per consumer and provider 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. Willingness to exchange is higher if a participant has more of an objective than he ideally wants and if the norm that he is following is influencing a lower value of that specific objective. If the current Web service is following norm m i.e. N_m then the willingness to exchange is calculated as

$$E_{ij} = |\frac{C_i}{N_m}| \tag{3.9}$$

and the amount exchanged would be

$$A_{ij} = (1 - WC_i)|1 - \frac{C_i}{N_m}| \tag{3.10}$$

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

$$E_{ij} = |\frac{C_i}{P_{ij}}| \tag{3.11}$$

and the amount exchanged would be

$$A_{ij} = (1 - WC_i)|1 - \frac{C_i}{P_{ij}}| \tag{3.12}$$

The *Norm* operator holds the cumulative knowledge of the entire system. It is used to share private knowledge without revealing the identity of any participating Web service. In the beginning, norms are populated with intelligently guessed values, and members of population are randomly assigned to different norms. After each iteration, every member of the population assesses its performance and provides feedback to the norm being followed. When progressing from generation i to generation *i+1*, following norm m, the difference in the norm value will be calculated as follows. First we need to find out if the recent changes have improved the overall fitness of the follower. We do this by calculating the difference between the current and previous fitness value of the current follower.

$$c = f_j - Prevf_j \tag{3.13}$$

If c >0 there is an increase in the overall fitness of that member of the population. Hence the positive change should be shared with other participants. Hence we update the cumulative knowledge value of that *Norm*. If a member of population does not follow any *Norm*, the algorithm just skips over that member and moves on to the next member in the population. Assume that provider i follows *Norm* i and the j component of the QoS vector was changed. If it is a positive change, we will add some value to the *Norm i* indicating that a higher values is suited assuming that the current P_{ij} value is higher that the N_i value.

$$P_{ij} > N_i \textit{ and } D_{ij} = P_{ij} - N_i \tag{3.14}$$

D_{ij} will give us the difference between the value of the current *Norm* and the member's corresponding QoS attribute value. The difference in the value for *Norm* will be calculated as

$$\Delta_N = (\frac{R_j}{j}) * (\frac{D_{ij}}{P_{ij}}) \tag{3.15}$$

$$N_i^{'} = N_i + k * \Delta_N \tag{3.16}$$

Where $N_i^{'}$ is the updated value of norm i and k is the learning factor, representing how much weight is given to the history as compared to the current value of the norm. In our experiments we found out that one tenth is a good value for k. Similarly, if a negative change is observed in the value of the any QoS vector resulting in better fitness value, then we augment the *Norm* value accordingly. Moreover, if a member is constantly improving its fitness value by following a norm m it will keep following that norm. On the other hand a decrease in the fitness value of member over a period of time will increase its chances of stop following that *Norm*. The probability of a member switching the norm it is following is calculated as

$$P_{switch} = 1 - \left(\frac{\sum_{1}^{h} (\frac{(P_{ij} - N_i)}{P_{ij}})}{h} \right)^2 \tag{3.17}$$

Where h represents the last h generations of this member Web service.

3.3.3 Crossover and Mutation

The crossover operator is invoked after applying the Norm operator. Roulette-wheel selection is used for selecting solution pairs for crossover. Roulette-wheel selection is analogous to a roulette wheel where the probability with which an individual is selected is proportional to its fitness value Baker [1987].

Solution rankings are used to perform 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 Whitley [1989]. Augmentation of the population for roulette-wheel selection is performed as follows:

$$CrossP_j = 1 - \frac{1}{R_j}(R_j - 1) \tag{3.18}$$

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 performed on the consumer/provier QoS vector pair. It has been proved that custom operators provide superior performance for real-valued problems Wolpert and Macready [1997].

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 an experimentally predetermined mutation rate. Mutation here involves arbitrarily changing one element of the negotiation vector and then applying a repair algorithm to ensure that objective values lies within the valid range for that member. A random mutation value is generated for each member in the population and compared to the mutation rate. If the mutation value is less than or equal to the mutation rate, mutation will occur in that solution.

3.3.4 Norm Dependency Modeling

We extend the *Norm* operator to incorporate the ability to perform a parallel search of multiple Web services, for them to be part a of the same composition. Assume that Web service C provides car insurance. To provide car insurance quotes, Web service C needs to gather the driving history and the credit score of an applicant. For simplicity, lets assume that Web service C negotiated contracts with Web service A for driving history and Web service B for credit score. Among other QoS components, Transactions Per Seconds (TPS) was one of the negotiated components and Web service A's negotiated contract included the proposition of providing 100 TPS. While negotiating with Web service B and Web service C the system found out that the maximum TPS offered by Web service B is 75 TPS which suits Web service C's budget. This entails that in a liner composition of Web service A, Web service B and Web service C the maximum throughput of the composition cannot exceed 75 TPS. The scenario would be same if we assume that system negotiate Web service B prior to negotiating with Web service A. One solution would be to create some temporary contract and then re-negotiate a new contract based on all found objectives. 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 components to the negotiating service, we may find a better solution.

In *WebNeg*, we use a sliding window approach to minimize the discrepancy among the dependent QoS components. The idea behind the approach is that since all negotiation participants optimize their QoS component values based on the corresponding *Norm's* value, we restrict the progress of corresponding *Norm* value for the dependent QoS component among all Web services (TPS for Web service A and Web service B). This ensures that the difference in the negotiated values is not greater than a predefined threshold of ΔV. Hence, sliding window is the interval that allows the *Norm* values for dependent QoS components to learn form the population space just like the *Norm* values for independent QoS components. At the end of the sliding window interval the *Norm* value for the dependent QoS component is restricted. The sliding window interval is calculated dynamically based on the level of discrepancy among the *Norm* values and participating services' preferred values. Therefore, 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 where we can calculate the pace of learning for the *Norm* values i.e. Interval Check point. Then we calculate the interval of the sliding window in term of number of generations. At the end, we calculate the generation number where we restrict the *Norm* values. Initial check point is described as:

$$G_i = G_{ICheck} \tag{3.19}$$

Where G_i is the *ith* generation of GA. The sliding window interval is calculated as follows:

$$\Delta_{Interval} = \frac{(f_i - f_j) + ((C_i + P_i) - (C_j + P_j))}{3} + \Delta N_{ij} \tag{3.20}$$

and the sliding window time will then be calculated as :

$$S_{Window} = \frac{G}{\Delta_{Interval}} \tag{3.21}$$

Where G is the total number of generations of GA. At every sliding window distance we restrict the corresponding *Norm* values of the dependent QoS component.

$$V_{Ci} = V_{Di} = min(V_{Ci}, V_{Di}) + \frac{min(V_{Ci}, V_{Di})}{100 + \Delta_{Interval}} \tag{3.22}$$

This ensures that the *Norm* values for the dependent QoS component closely follow each other and leads the system to a solution where the negotiated QoS values for the dependent components do not have a large disparity among them which leads to an over all better solution.

3.4 Study and Results

To evaluate the *WebNeg* architectural framework, we performed three experiments covering different scenarios. First, we compared the performance of *WebNeg's Norm* operator with a traditional GA with only mutation and crossover operators, a random search, and a hill-climber based approach. Second, we compared *WebNeg* with similar approaches used in the literature SBA Nitto et al. [2007], NBA Niu and Wang [2008], SWC Lecue [2009], BLGAN Sim et al. [2009] and GTFSN Figueroa et al. [2009] on the basis of the utility of the proposed solution and the time it takes to achieve that solution. Third, we implemented the dependency modeling for *Norm* (in multi-attribute negotiation scenario) and performed experiments to show the effectiveness of our approach.

Experiment Environment

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 500 generations each. WSDream-QoSDataset Zhang et al. [2010] 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 *Webneg*'s negotiation component.

3.4.1 Experiment 1

In the following, we describe the experiment details for approaches used in comparing the performance of *WebNeg's Norm* operator.

Traditional GA: A traditional GA was implemented by removing the *Norm* operator. It only uses the simple GA operators of crossover and mutation. All other parameters are the same as that of *WebNeg's* GA with *Norm* operator.

Random Search: Random search simulates the behavior of arbitrarily exploring the search space in the hope of finding a solution. It is applied on one half of the gene at a time. Either the consumer's Web service gene or the provider's Web service gene parameters are augmented using Equation 3.23. This augmentation likelihood is determined randomly. Once selected, a random number is generated for each QoS parameter that lies between the allowable range for that participant using Equation 3.24. Then all the numbers are aggregated by subtracting their respective minimum values. This summation is then averaged out and either randomly added to, or subtracted from all the parameters. Then, the repair algorithm is applied (to ensure that all the constraints from Equations 3.4 through 3.6 are satisfied), and solutions are ranked to be taken to the next generation.

$$AugVal_{ij} = \frac{\sum_{j=0}^{n}(Random_{ij} - P_{ij(min)})}{j} \tag{3.23}$$

$$Random_{ij} = Random(P_{ij(min)}, P_{ij(max)}) \tag{3.24}$$

Hill-Climber: Hill-climber uses the concept of randomly exchanging the QoS values. It is also uses Equation 3.18 to determine the amount of objectives to be exchanged. However, *WebNeg* uses either Equation 3.10 or Equation 3.12 to determine if the exchange will occur. Once a gene is randomly selected, the exchange takes place.

Figure 3.12 shows results of a sample execution of the above mentioned techniques after each generation. Note that the actual output values are listed without *Elitism*. Lower values of degree of disagreement (on the Y-axis) are desired as they show a higher chance of reaching an agreement. For instance, assume that consumer A wants a solution that has an *Availability* value of 98% and the provider B presents a solution that has an *Availability* value of 95%. The degree of disagreement among the consumer A and provider B is small and hence they are more likely to reach a solution. Moreover, note that both the consumer and provider must have some overlapping search space values for the algorithm to identify a solution. If both the consumer and provider have mutually exclusive ranges of QoS components, the algorithm fails and no solution is returned.

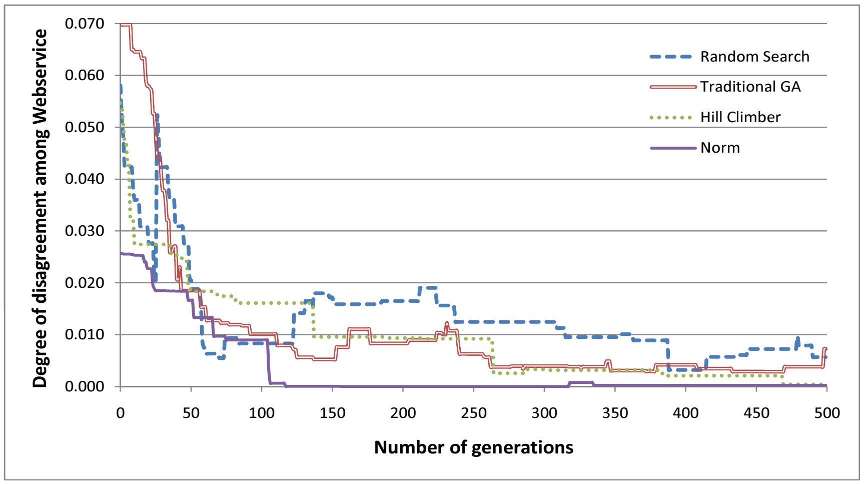


Figure 3.12: Sample run of WebNeg's Norm operator for multi-party negotiation, in terms of degree of disagreement among consumer and providers offers

The graph confirms the assumption that the probabilistic nature of the traditional GA does not guarantee that the best solution will be passed on to the next generation. Hence, using *Elitism* to ensure *Pareto optimality* is an important factor in *WebNeg* as discussed earlier in the methodology section. *WebNeg's Norm* operator takes almost $1/4^{th}$ the time to reach an agreeable solution. The graph shows that *Norm* found a mutually agreeable solution after 100 generations, where as *Hill Climber* took 475 generations, *Traditional GA* took 450 generations and *Random Search* took 375 generations to find their respective best solutions. Hence, we can safely deduce that a solution is found faster in *WebNeg* and since it found solutions with less degree of disagreement, the solution quality is also improved.

Figure 3.13 shows the learning graphs of *Norm* for the above experiment run. We can see that *Norm* values for Throughput, Reliability and Availability stabilize fairly quickly but the *Norm* value for Response Time stabilizes around the 100th generation. Correspondingly, we can see in Figure 3.12 that our technique converges to the solution around the 100th generation. Since any GA (and hence *Norm*) can not

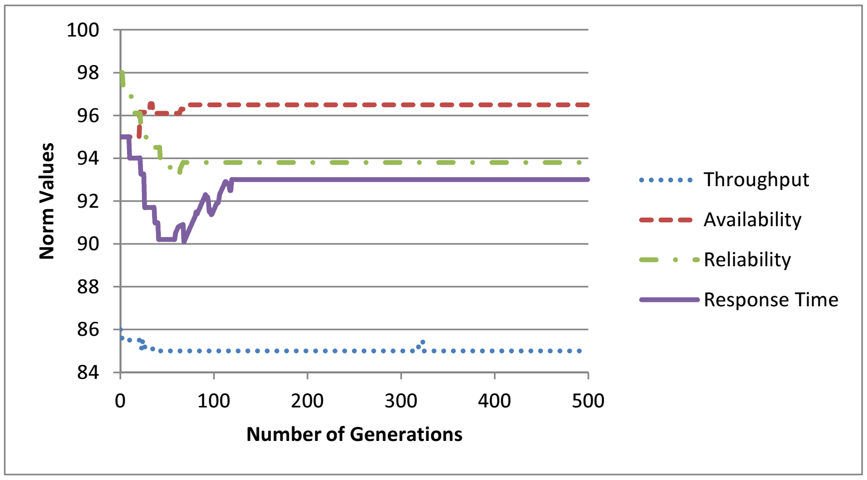


Figure 3.13: Sample learning graph for Norm operator in WebNeg's multi-party negotiation scenario

guarantee the same solution every time, it is appropriate to analyze the performance of *Norm* over multiple runs. Table 5.4 shows the average of 200 runs for the four algorithms. As mentioned earlier, lower values (degree of disagreement) are desired as they show a higher chance of reaching an agreement. We can see that the best solution of 0.00002 returned by *Norm* provides an optimal solution in comparison with the 'best solution' returned by any of the other technique. As far as the worst solution is concerned, *Norm* still performed better than any of the other techniques. The worst solution of 0.03163 returned by *Norm* is almost twice as good as that of *Hill Climber*, the second best technique. The average solution returned by *Norm* also shbiows an improvement from the next best i.e. (*Hill Climber*). Similarly, *Norm* exhibits the lowest standard deviation of 0.01157. Low mean and low standard deviation indicates that our technique performs consistently better in comparison with other techniques.

Table 3.3: Average Results over 200 Iterations For Norm

	Random Search	Traditional GA	Hill Climber	Norm
Min	0.00568	0.00027	0.00041	0.00002
Max	0.06153	0.08547	0.05171	0.03163
Mean	0.02718	0.02192	0.01461	0.00925
Std. Dev	0.01663	0.02177	0.01668	0.01157

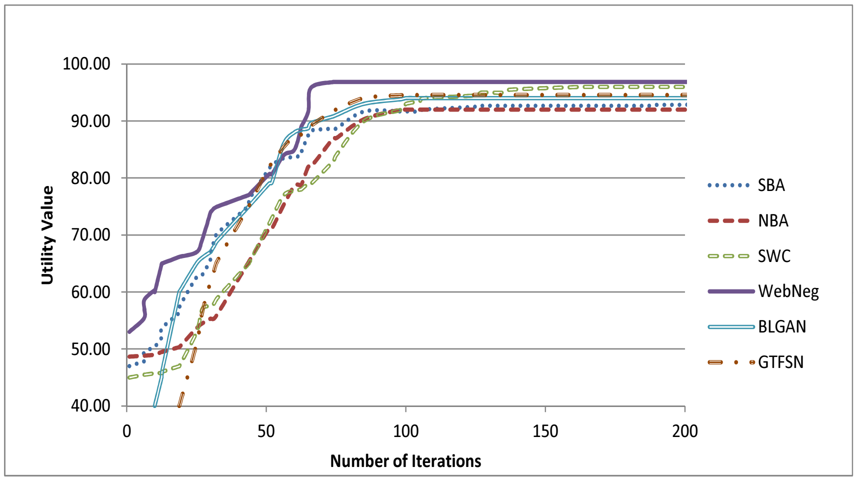


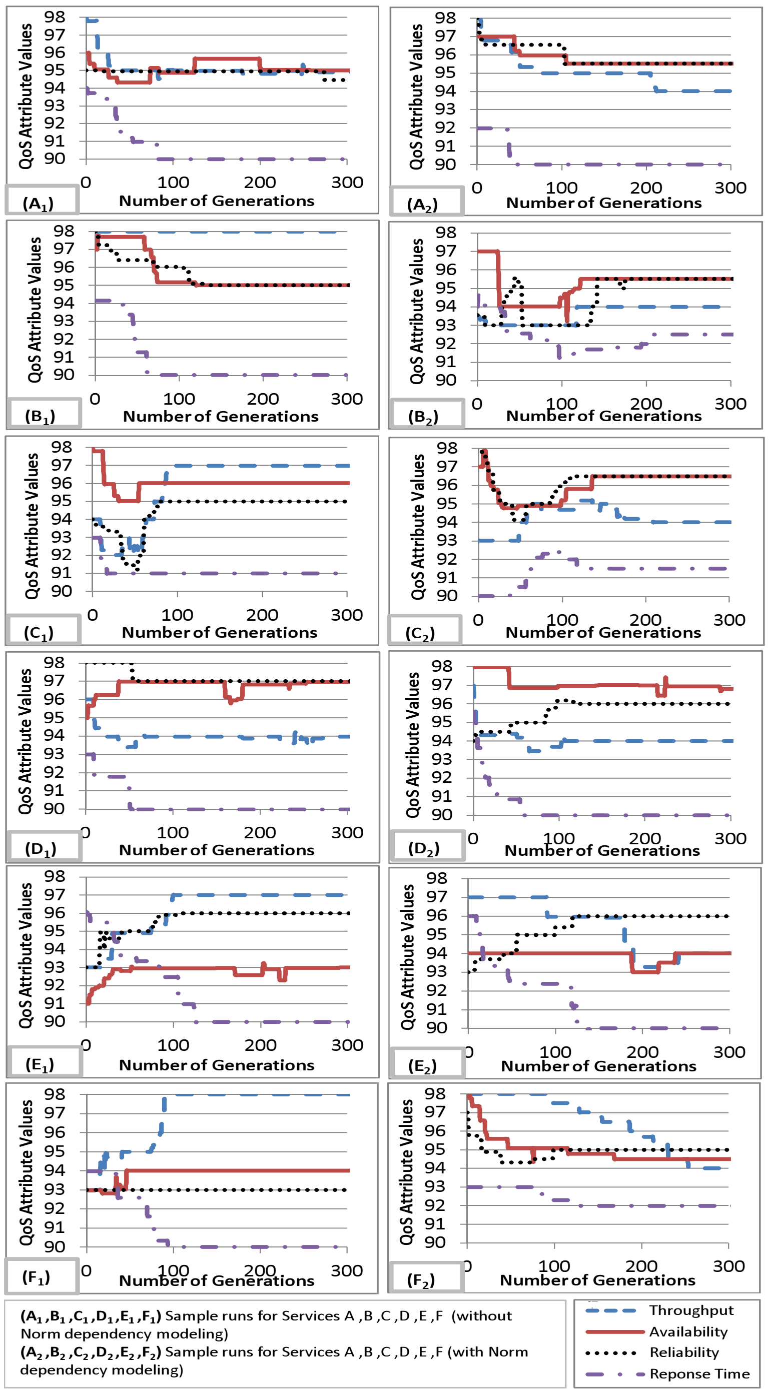
Figure 3.14: Utility value comparison of WebNeg with similar service negotiation techniques

3.4.2 Experiment 2

In this experiment we compare *WebNeg* 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 Nitto et al. [2007] 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 consumer 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 Niu and Wang [2008] 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 Lecue [2009] 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. BLGAN Sim et al. [2009] uses a Bayesian learning based approach with GA and incomplete information model to learn the reserve price of it opponent. GTFSN Figueroa et al. [2009] presents a game theoretical model of signaling games for Service Level Agreement negotiation. The results are presented in Figure 6.14. We can see that *WebNeg* 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 *WebNeg* finds a solution within the 97% utility range in about 66 generations, while SWC (the second best) takes about 3 times more iterations. Other techniques fail to generate such a solution and NBA and SBA plateau around the 92% range while BLGAN and GTFSN only reach a solution of around 94% utility. The results suggest that our approach outperforms similar methods both in terms of finding the optimal solution and the amount of time it takes to find that solution.

3.4.3 Experiment 3

For showing the results of dependency modeling we considered the following 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 better deals, more business, and provide clients a one-stop



(A_1,B_1,C_1,D_1,E_1,F_1) Sample runs for Services A ,B ,C ,D ,E ,F (without Norm dependency modeling)
(A_2,B_2,C_2,D_2,E_2,F_2) Sample runs for Services A ,B ,C ,D ,E ,F (with Norm dependency modeling)

Figure 3.15: Experiment results of service negotiation without $(A_1 - F_1)$ and with $(A_2 - F_2)$ dependency modeling

shop for their vacation planning. The proposed vacation package includes taxi to the airport, plane tickets, hotel reservation, car rentals, sight-seeing, plus the travel insurance quote (i.e. service offered by sites such as Orbitz, Expedia, Travelocity, etc.). We can see that this is a scenario where all the six services are executed in a sequence to achieve the desired results. For ease of comprehension we refer taxi service as Web service A, flight reservation as Web service B, hotel reservation as Web service C, car rental as Web service D, sight-seeing as Web service E and travel insurance as Web service F.

We use *WebNeg* to negotiate these six Web services (*Web service A*, *Web service B*, *Web service C*, *Web service D*, *Web service E* and *Web service F*) where, the negotiation vector consists of four QoS attributes $<$ Throughput, Availability, Reliability, Response Time $>$. We assume that *Throughput* is the dependent QoS component among all the Web services. Figure 5.3-($A_1 - F_1$) shows the values for consumer offers when the Web services are negotiated separately from each other. The negotiated vector for *Web service A* (Figure 5.3 -A_1) is <95,95,95,90>, *Web service B* is <98,95,95,90>, *Web service C* is <97,96,95,91>, *Web service D* is <94,97,97,90>, *Web service E* is <97,93,96,90> and *Web service F* is <98,94,93,90>.

We can see that the dependent attribute of *Throughput* has a value of 95 TPS in *Web service A* reached around generation number 83. In *Web service B* the value of 98 TPS is reached around generation number 110. In *Web service C* the value of 97 TPS is reached around generation number 90. In *Web service D* the value of 94 TPS is reached around generation number 98. In *Web service E* the value of 97 TPS is reached around generation number 104. In *Web service F* the value of 98 TPS is reached around generation number 100. The overall output of the composed system will thus have a *Throughput* value of 94 TPS i.e. minimum of the TPS vales 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 all six Web services, we can see (in Figure 5.3-($A_2 - F_2$)) that the negotiated QoS vector for *Web service A* is <94,95.5,95.5,90>, *Web service B* is <94,95,95,92.5>, *Web service C* is <94,96.5,96.5,91.5>, *Web service D* is <94,97,97,90>, *Web service E* is <94,94,96,90> and *Web service F* is <94,94.5,95,92>. We get the TPS of 94 for *Web service A* around generation number 200 and the same TPS value of 94 for *Web service B* around generation number 120. The TPS value of 94 for *Web service C* was reached around generation number 212 and the same value for *Web service D* around generation number 124, for *Web service E* the same value was reached around generation number 190. Finally the TPS value of 94 for *Web service F* was reached around generation number 255. So the composite system will have the over all TPS value of 94. We can clearly see that though the TPS value is the same but we get a better overall solution for the system i.e. component services now have higher negotiated values for *Availability*, *Reliability* and *Response Time* e.g. *Web service A* has higher value for *Availability* i.e. 95.5 compared to the original 95. Similarly *Web service B* has higher value of 92.5 instead of 90 for *Response Time*. We can see a similar increase in other negotiated QoS values for the remaining Web services. 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 the component Web services. This increases the overall utility of the system and shows the practicality of our proposed technique.

3.5 Conclusion and Future direction

We presented a framework (*WebNeg*) for Web services negotiation to enable consumers and providers in negotiating QoS attributes for SLA. *WebNeg* uses a GA based approach to conduct multi-party multi-objective negotiations. It integrates the concepts of Pareto optimality and multiple decision making preferences of the participants. 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 *Norm's* improved performance in comparison with similar optimization techniques. We further extended our approach to incorporate dependency modeling for different QoS parameters among multiple services to formulate optimized solutions.

A limitation of the presented technique involves the assumption of *a priori* decision model articulation, which requires that all the negotiating participants can identify and share their preferences at the beginning of the negotiation. These limitations can be overcome if participants decide to provide their own negotiating component rather than articulating their preferences. However, this would limit the effectiveness of sharing private information. Secondly, the results obtained using the dependency modeling of *Norm* take twice as much time as the ones without using the dependency modeling which, indicates that there is room for improvement in the *Norm's* learning model.

We are currently investigating on enhancing the effectiveness of private information sharing by exploring the possibilities of having people 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 *WebNeg*. Existing communication protocols [GRAAP] Smith [1980] Lecue [2009] [W3C] Andreoli and Castellani [2001] lack such capabilities. This requires a new standard language that could be used to pass on all these dependency constraints and decision model to *WebNeg*. We are exploring the options of extending WS-Negotiation Hung et al. [2004] and WS-AgreementNegotiation Wieder [2010] by adding the support of complex logical functions for articulating these and similar complex decision models. We are exploring the options of extending WS-Negotiation Hung et al. [2004] and WS-AgreementNegotiation Wieder [2010] in this regard. We are also working on a solution that moves away from the centralized approach in the favor of a more adaptive distributed model.

CHAPTER 4 : SOCIAL RECOMMENDATION BASED NEGOTIATION

4.1 Introduction

In recent years, information system design has been influenced by the service oriented paradigm to facilitate easy integration between organizations that provide their services on Web Alonso et al. [2004]. 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 Medjahed et al. [2003].

Currently, the Web 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 Booth and Liu [2006] for service description, SOAP Standard [2007] for communication etc.), the research interest is shifting towards the non-functional aspects of Web services Papazoglou and Heuvel [2007]. Since Web services likely span across multiple enterprize boundaries where different providers exercise control over their propriety 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. Negotiations play a prominent role in the decision making process of different aspects of human life, such as business, scientific, social and political interactions etc Kleindorfer et al. [1993] Mora and Wang [1998]. 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 Golbeck [2008] Pitsilis and Knapskog [2012]. In recent years 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 and YouTube for photo and video sharing respectively, Digg for social news reading, and Delicious for social bookmarking. These sites mainly rely on their users to create and contribute content from online relationships and to provide personalized recommendations. 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.

In this chapter 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 information (i.e., determine the trustworthiness of the claims published by individual services) and leverage the collected information to construct a decision model for conducting multi-objective and multi-agent negotiations of component services.

4.2 Motivation and Approach

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 Jennings et al. [2001]. In existing literature, this has usually been accomplished in an ad-hoc manner Jennings et al. [2001] Resinas et al. [2012], 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.) Yu et al. [2008] Zarras et al. [2004], there may be more than one combination of these attributes that may be suitable under a specified negotiation context. This implies that the negotiation system should not restrict its user to a single negotiable attribute(e.g. price) rather it should allow the users to express multiple attributes for the negotiation process. In SOAs it is very much expected that all the participants using the system may not be similar. They may implement heterogeneous (probably incompatible) protocols. Thus, there is a need for supporting multiple negotiation protocols, or be able to consent on the negotiation protocol for cases where a participant supports multiple ones. Different participants prefer different negotiation strategies(auction, bargaining etc.) based on their decision models, domains, preferences and history. Hence, an automated negotiation system must implement multiple decision models so that it could support client specific negotiations. Unlike traditional software environments, SOAs enable delivery of the same service to different customers with varied quality of service (QoS) requirements Elfatatry and Layzell [2005]. Moreover, since negotiation is a dynamic and interactive process, the user preferences could change over time. The user may change the required value of a QoS attribute during the negotiation process, (as it learns new information during the negotiation) or may even add or remove new QoS attributes. Thus, the negotiation system should allow the user preference about the negotiation process to be changed over time . Since services are not stored or downloadable, the market environment tends to be very dynamic Gimpel et al. [2003]. The ability to create on-the-fly dynamic solutions emphasizes the need of conducting simultaneous negotiation with multiple component services, owned by different parties, at the same time. Simultaneous negotiations are desirable in volatile service markets to allow selection of the most profitable agreements for the participants Gimpel et al. [2003]. This entails that the participants should be equipped to change their strategies/decisions at runtime, based on market dynamics and changing contexts Ros and Sierra [2006]. Dynamic service selection is mostly used in two scenarios. One to replace a faulty service in the system and secondly when searching for component services for a newly formulated solution. In the later case we can safely assume that the system would be composed of more than one service(s) and hence it may need to simultaneously negotiate multiple services. Certain system properties are a composite function of its component services e.g the overall throughput of the system is limited by its component service 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 component services.

SNRNeg presents a framework for a negotiation service that is geared towards meeting the above mentioned requirements. Our framework 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. SNRNeg uses a trust-based recommendation system that uses the distributed information present in a social network and filters it based on the trust relationship among its peers. SNRNeg is designed toward a scenario where a customer is involved in simultaneous negotiations with multiple providers. It makes use of the private information of each negotiation process (without compromising the identity of any participant) to adapt quickly, and significantly reduces the search space by guiding the negotiation process toward a mutually agreeable solution and incorporates the relationship dependencies among different component services and QoS attributes.

4.3 Scenario

In this section we present a scenario to motivate the problem and our proposed approach. 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, and provides clients a one-stop shop for their vacation planning. The proposed vacation package includes plane tickets, hotel reservation, car rentals, sight-seeing, etc. plus the travel insurance quote (i.e. service offered by sites such as Orbitz, Expedia, Travelocity, etc.). 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.

4.3.1 Current Approach

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 ran into some unexpected issues. Some of the services were so old that either their implementation did not exist or were not able to produce any output. Some of results returned by the services were outdated or plain incorrect. Some of the services took unexpectedly long time to return any meaningful results. The most interesting was that different services returned 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 trial and error and then hope that he finds a good combination of services that provide cost effective results to the company and consequently to its end users.

4.3.2 Proposed Approach

John spent some time going through the trial and error method and finally gave up on it. Being a software engineer he started 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 that he feels confident about, and believes that is one of the best solutions out there. He started thinking on how this could be achieved in a normal world scenario. His usual plan of action is to search online, read reviews on what he wants to buy, and then ask some of his friends that he trusts may have 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. 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 service and in turn would 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

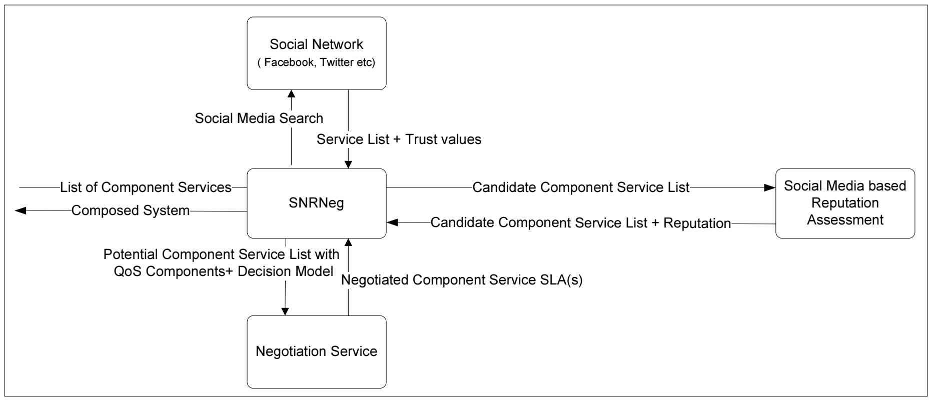

Social Network
(Facebook, Twitter etc)
Social Media Search
Service List + Trust values
List of Component Services
Composed System
SNRNeg
Candidate Component Service List
Candidate Component Service List + Reputation
Social Media based Reputation Assessment
Potential Component Service List with QoS Components+ Decision Model
Negotiated Component Service SLA(s)
Negotiation Service

Figure 4.1: Proposed approach for the social network based negotiation service.

non-functional QoS components as shown in Figure 4.1. All these components would in turn reflect on how his own solution is perceived by the end users. Some of these components are independent of each other i.e. availability, reliability and some are dependent e.g. throughput etc. Hence, he will have to be careful when selecting service to make sure that not only they work well individually but they should also be feasible for the composite solution.

4.4 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 strategy considering the heterogenous preferences of each node and its interaction with that particular item.

Let us consider a set S_a of N_a nodes a_1, a_2, a_3,.....,a_i. 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 chapter 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_o of N_o object as o_1, o_2, o_3,.....,o_j. These objects represent anything that could have a rating. In our running example these are Web services. We further assume that these objects are classified under one or more N_c categories from S_c, denoted by c_1, c_2, c_3,.....,c_k. 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_i is associated to one certain preference profile which is one of N_p preference profiles in the system, where $S_p = p_1, p_2, p_3, ..., p_l$. Such a profile p_i is a mapping which associates to each object $o_j \in S_o$ a particular corresponding rating $r_j \in [-1, 1], p_i : S_O \rightarrow [-1, 1]$. This is illustrated in Figure 4.2.

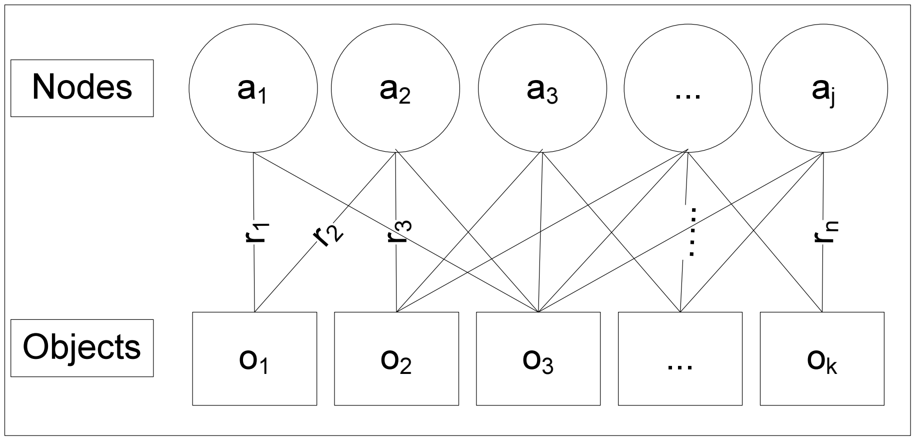


Figure 4.2: Nodes and Objects mapping

Each node a_i keeps track of a trust value $T_{a_i,a_j} \in [0, 1]$ to each of its neighboring node a_j. The trust value between two nodes T_{a_i,a_j} 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 Malik and Bouguettaya [2009b]). 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 equation 4.1.

$$T_{a_i,\ldots,a_j} = \prod_{(a_k,a_m)\in path(a_i,a_j)} T_{a_k,a_m} \tag{4.1}$$

i.e. 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. Figure 4.3 illustrates a part of such a social network of nodes and a chain of trust relationships between two nodes.

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 to for the selected category c_i 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).

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

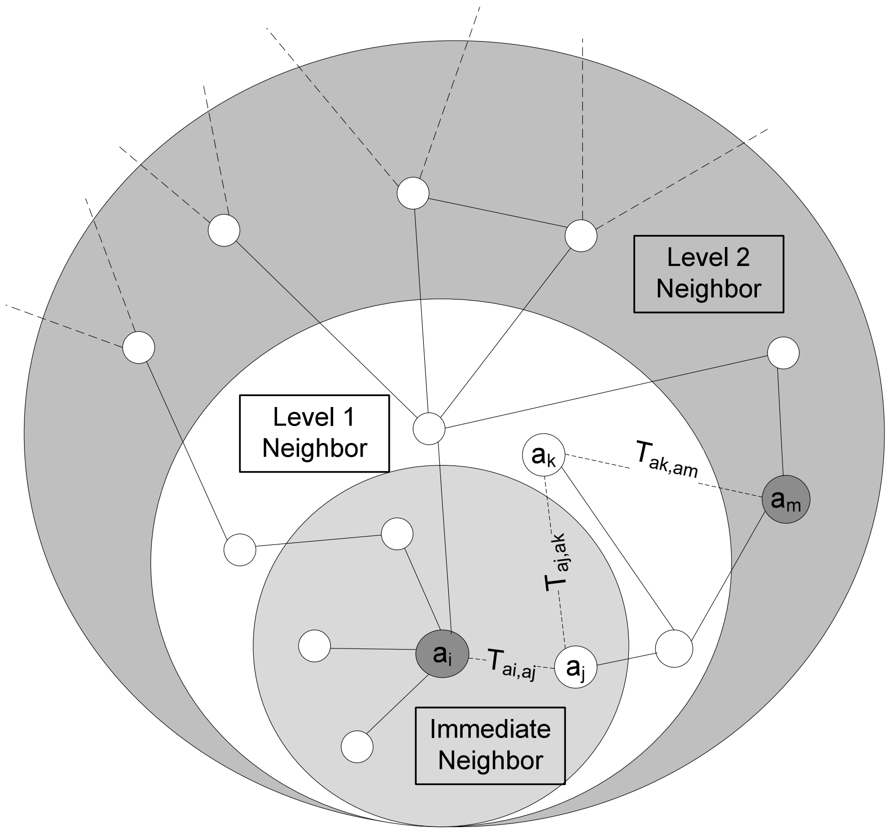


Figure 4.3: Social network and trust paths

```
1 Node $a_i$ prepares a $query(a_i, c_j)$ for category $c_j$ and then transmits it to its
  neighbor
2 Each neighbor $a_k$ receives the $query(a_i, c_j)$ and either
3 returns a response $(a_k, a_i, (o_j, r_j), Ta_i, .., a_k)$, it is 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 pass the $query(a_i, c_j)$ to its own neighbor if it does not know the rating
  $r_j$ for the particular category $c_j$
```

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 objects to recommended. We denote the $query(a_i, o_j) = Q$ and a response $(a_k, a_i, (o_j, r_j), T_{ai,...,ak}) \in R$ 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 will convert the recommendation into an intermediate variable Γ such that it lies in the range or $[-\infty, \infty]$

$$\Gamma_{a_i,...,a_j} = \frac{1}{2} ln \left(\frac{1 + 2(T_{a_i,...a_j} - T_0)}{1 - 2(T_{a_i,...a_j} - T_0)} \right) \in [-\infty, \infty] \tag{4.2}$$

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

$$\gamma = exp(\Gamma_{a_i,..,a_j}) \tag{4.3}$$

$$L(response(a_k, a_i, (o_j, r_j).T_(a_i, ...a_k))) = \frac{\gamma}{\sum_R \gamma} \tag{4.4}$$

Where Γ 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 received a recommendation from another node, but through multiple paths. For example, a_i may be linked to a_k through a_j , but also through a_l . Then, each of the two responses would be assigned a probability according to equation 4.5. 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 Jøsang and Pope [2005].

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 o_k be the chosen object. Then, assuming node a_i having profile

$p_i, p_i(o_k) = r_k$ is the experience that a_i has made by following the recommendation transmitted through a_j . Now if $r_k \geq 0$ then

$$\alpha_{a_i,a_j}(t+1) = \beta\alpha_{a_i,a_j}(t) + (1-\beta)r_k \tag{4.5}$$

and for r_k <0 then

$$\alpha_{a_i,a_j}(t+1) = (1-\beta)\alpha_{a_i,a_j}(t) + \beta r_k \tag{4.6}$$

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:

$$T_{a_i,a_j}(t+1) = \frac{(1-\alpha_{a_i,a_j}(t+1)}{2} \in [0,1] \tag{4.7}$$

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 loose 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.

4.5 Performance Study

We use our previously developed GA based approach to determine the efficiency of our approach and 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 measure on four QoS components (reliability, availability, throughput and accessibility). We used a random directed graph to show the social network with pseudo-random assignment of trust values for each link to simulate a real world scenario. We simulated 1000 transactions where each transaction is performed in one time unit and then the results are fed back into the system. We averaged out our results for 15 rounds. The GA based negotiation process was tested over 200 iterations consisting of 500 generations each.

The trust value based recommendation is simulated on our random graph. Each of the 50 graph nodes are asked for recommendation in 3 different categories. The system is evaluated on the consistency of recommendations after each round. We can see in the graph 4.4 that the system concedes at a different pase based on the β values. Higher values allow the system to quickly reach at a stable state.

After receiving the list of recommended services we calculate the reputation of these services for each category using our previously developed trust module.

We simulated the reputation assessment process with a real world scenario where the number of honest and dishonest ratings fluctuate (i.e. one is higher than the other at a particular time instance). Figure 4.5 shows the case where majority of the ratings are honest (e.g., for the list of services obtained from the social network). For

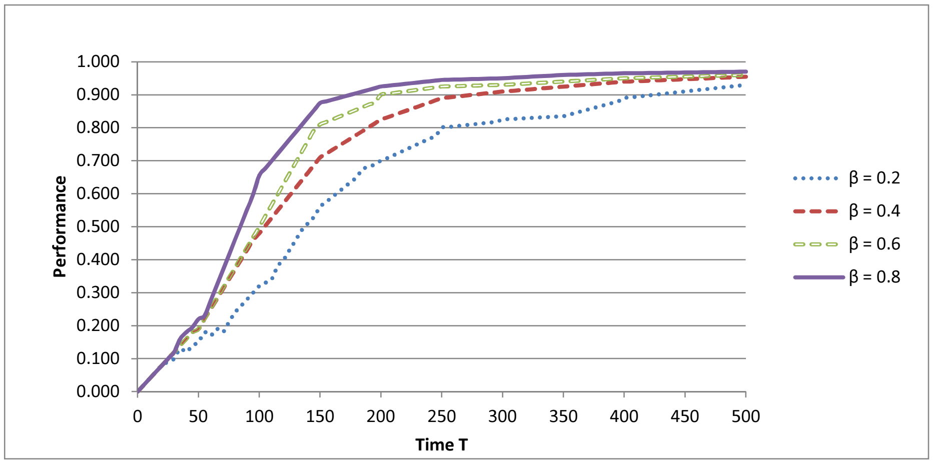


Figure 4.4: Recommendation system performance

complete experiments and details please see Malik and Bouguettaya [2009c] Malik and Bouguettaya [2009d]. Figure 4.5-A shows the comparison between original provider performance and the assessed reputation for providers that perform consistently.

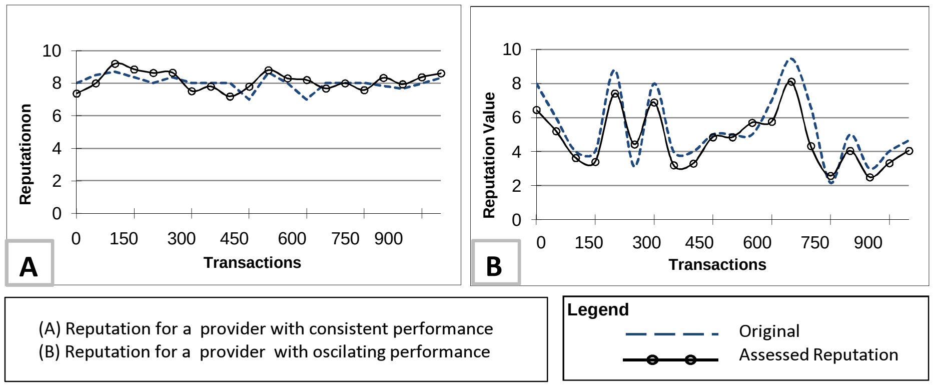


Figure 4.5: Reputation assessment for high rater credibility scenario

Once we have the reputation values for the candidate services we use them for negotiating a SLA based on QoS values. Figure 6.14-A 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 Nitto et al. [2007] 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 Niu and Wang [2008] 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 Lecue [2009] 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. We compare the results of SWC with 20 services (*SNRNeg* had 50 providers). The results are presented in Figure 6.14-B. 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) take 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

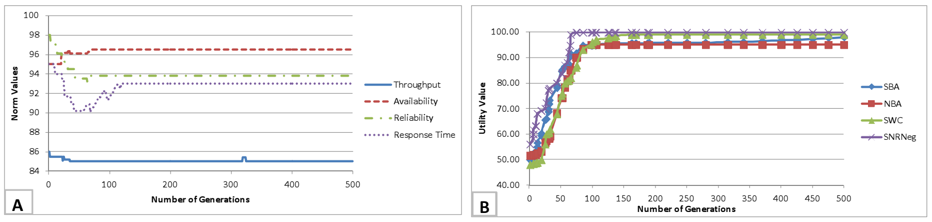


Figure 4.6: A: Sample run for SNRNeg B: Utility value comparison of SNRNeg with similar service negotiation techniques

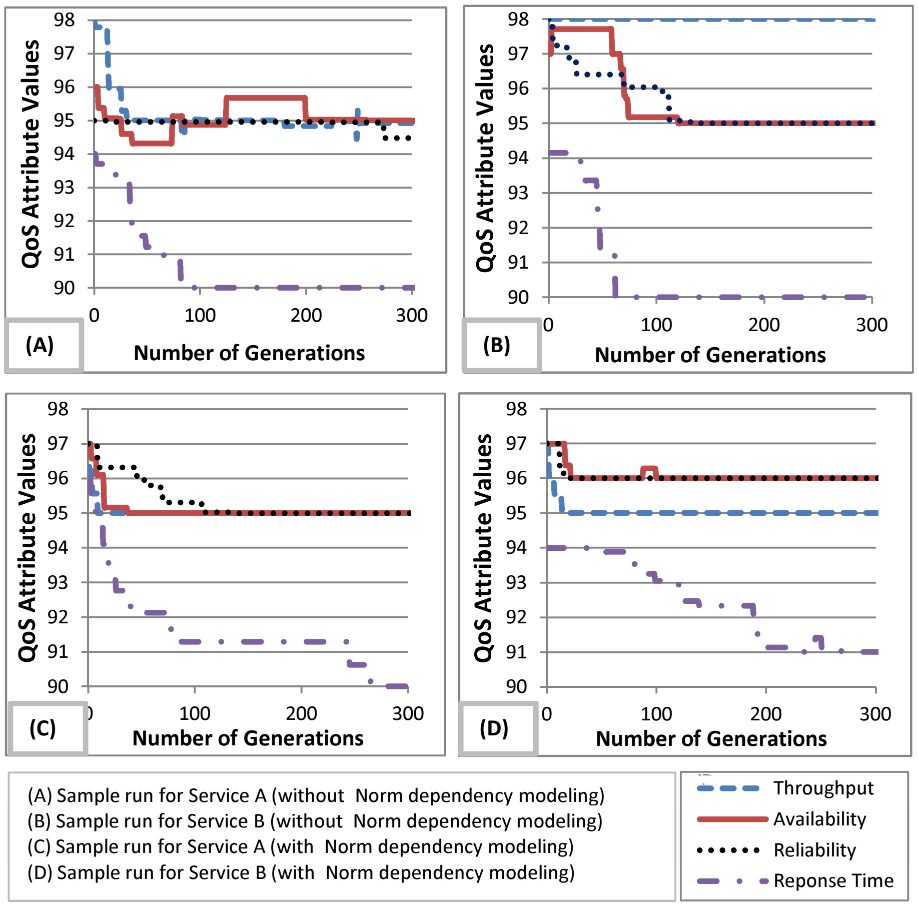


Figure 4.7: SNRNeg dependency modeling. A,C - Service run without dependency modeling. B,D Service run with dependency modeling

among *Service A* and *Service B*. Figure 4.7-A and 4.7-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 vales 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 Figure 4.7-B and 4.7-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.

4.6 Conclusion and Future direction

We have presented *SNRNeg*, a framework for Web service negotiation using social networks to enable customers and providers establish SLAs (using the trust relations ships 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 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 felicitates 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. Secondly our system assumes that these graphs are static in nature and hence work using snapshot of them. In reality these graphs are very dynamic and hence would require a much more agile approach of dealing with them. Thirdly, our negotiation service takes the assumption of a static environment, where the Web service procurement time window is so small that the user preferences do not change during the course of negotiations.

We are currently working on identifying the patterns found in 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 people 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 behaviorial factors of the social Web. Moreover, we are also looking into how to distribute (propagate) the rating of a multi-node path recommendation.

CHAPTER 5 : SEMANTIC WEB RULES AND NEGOTIATION

5.1 Introduction

Service Oriented Architecture (SOA) is becoming a very popular software design paradigm. A service (Web service in our case) is an autonomous, platform-independent program accessible over the web. In recent years, information system design has been influenced by the service oriented paradigm to facilitate easy integration between organizations that provide their services on Web Alonso et al. [2004]. The increase in popularity and automation requires these services to be accessed directly by applications rather than by humans. A service is characterized by its input parameters, the outputs it produces, and the actions that it initiates. The input parameter may contain some pre-conditions, and likewise, the outputs produced may have to satisfy certain post-conditions. In order to make Web services more practical and easier to use with automated tools we need an infrastructure that allows users to discover, deploy, synthesize, and compose services automatically. 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 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 Booth and Liu [2006] for service description, SOAP Standard [2007] 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 Papazoglou and Heuvel [2007]. In contrast to functional equivalence where and exact match is required, non-functional components could be matched within some tolerance level. Hence, we can use automated negotiation to formulate an acceptable solution.

Negotiations play a prominent role in the decision making process of different aspects of human life, such as business, scientific, social and political interactions etc Kleindorfer et al. [1993] Mora and Wang [1998]. 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. The idea is to simultaneously optimize a series of multiple objectives, considering the constraints of both the service providers and customers.

Most of the automated negotiation solutions H.Tung and Lin [2005] Lau [2007] Jonker et al. [2007] Paurobally et al. [2007a] assume that all the negotiating parties have the capabilities to support a prospected negotiation protocol and that they all allow same policies and rules to be articulated for negotiation. This could be as specific like using the exact same units for quantification of different parameters i.e. mega-bites vs kilobites of data. Although this assumption is very important in a negotiation system, in automated SOA service discovery, requesters and service providers might not know each other in advance. This means that it is quite probable that negotiation cannot be enacted, since a common supported negotiation protocol is not easy to be found.

This chapter provides the details of our automated negotiation framework that allows users to use different negotiation protocols to participate in a multi-party and multi-criteria automated negotiation process for Quality of Service components for a composite solution. We use semantic web rules to convert different protocols/units so that the negotiation process could take place among heterogeneous participants. We further enhance this concept buy using ontologies to formulate Service Level Agreements after a successful negotiation process.

5.2 WS-Negotiation

We use WS-Negotiation as one of the protocols that is used to conduct negotiation in our system. WS-Negotiation is a well-established specification that allows users to express and manage such SLAs. It defines a standardized protocol for managing agreements, while being flexible concerning their actual (domain-specific) content. However, it does have a few shortcomings.

Consider the example of a workflow engine F that constructs a process consisting of several services e.g. service1 followed by service2. If each individual service provider participating in the workflow could guarantee the QoS values of its individual component services then the QoS values of the composition would be greatly improved. Thus, F could support the negotiation of QoS agreements for the overall process with its own client. This implies that the composition needs to negotiate multiple QoS parameters during the same negotiation process. Hence it needs to send multiple offers/counter offers. One shortcoming of WS-Negotiation is that the specification does not allow for such multi-round negotiations, but only considers one-shot agreement creation, i.e., the responder has to immediately accept or reject an offer.

Furthermore, existing and valid agreements cannot be modified once established, except by terminating the existing SLA and creating a new one. The negotiation language should allow the existing SLA's to be re-negotiated based on the new parameters. Hence the WS-Negotiation specification should allow for multi-round

Sample 1 WS-Negotiation XML

```
<?xml version='1.0' encoding='UTF-8'?>
<env:Envelope xmlns='http://provider.example.com/2003/ns'
xmlns:env='http://www.w3.org/2003/05/soap-envelope'>
<env:Body>
<negotiationMessage id="1854" ref="None"
type="Offer">
<sender>user.example.comURI</sender>
<receiver>car-rental.example.com</receiver>
<content>
<issue>types-of-car
<alternative preference = "1">
compact
</alternative>
<alternative preference = "2">
full size
</alternative>
</issue>

</content>
<expiry>07/01/2003</expiry>
</negotiationMessage>
</env:Body>
</env:Envelope>
```

agreement negotiations, support re-negotiations and allow both parties to terminate the process. An example of a message is shown in Sample 1.

Now if the WS-Negotiation specification would allow the negotiation participants to send a *counterOffer* proposal that would allow the participants to increase its response type from the original *accept,reject,terminate* to one where it can send a counter offer in response to the original offer from the participants, it will be able to support multi-round negotiations. Similarly if we allow the participants to *retractOffer*, if the original offer has not yet been responded to, it will allow the participants to propose better offers without getting stuck into waiting for response for an initial offering. Similarly we can allow the offers to have an expiry time that would void it after a certain period of time. This will allow its participants to propose a new offer. This will also consider the scenarios where a negotiation participant may become unresponsive. The negotiation participants are allowed to agree on this expiry time in the beginning of the negotiation process. Similarly there must be a process to re-negotiate an existing SLA. Although this re-negotiation could be achieved using a two step process where the initial SLA is first canceled and then a new SLA is negotiated. However, this process does not guarantee that the parties re-negotiating the SLA would end up creating a new SLA. The second round of negotiations may end up as a failure and the participants will loose their initial SLA. Hence having the option to *re-negotiate* an SLA allows a lot of flexibility in the negotiation architecture. below is an example of how WS-ReNegotiation may be used to achieve this process.

Sample 2 WS-Renegotiation constructs XML

```
<wsag-neg:RenegotiationExtension>
    <wsag-neg:ResponderAgreementEPR>
wsa:EndpointReferenceType
    </wsag-neg:ResponderAgreementEPR>
    <wsag-neg:InitiatorAgreementEPR>
wsa:EndpointReferenceType
    </wsag-neg:InitiatorAgreementEPR> ?
    <wsag-neg:ResponderNegotiationEPR>
wsa:EndpointReferenceType
    </wsag-neg:ResponderNegotiationEPR>
    <wsag-neg:InitiatorNegotiationEPR>
wsa:EndpointReferenceType
    </wsag-neg:InitiatorNegotiationEPR> ?
    <wsag-neg:NegotiationOfferContext>
wsag-neg:NegotiationOfferContextType
    </wsag-neg:NegotiationOfferContext>
    <xsd:any /> *
</wsag-neg:RenegotiationExtension>
```

wsag-neg:RenegotiationExtension: This is the outermost element of a Renegotiation Extension document. This document is passed to an agreement factory as a critical extension to createAgreement. An agreement factory MUST be able to understand all critical extensions that are contained in a createAgreement call . If this is not the case, the factory MUST return an error. *wsag-neg:RenegotiationExtension/wsag-neg:ResponderAgreementEPR*: This REQUIRED element specifies the endpoint reference to the original instance of the responder agreement. If an Agreement Responder decides to accept an offer for a renegotiated agreement, the state of this agreement MUST change to Completed. *wsag-neg:RenegotiationExtension/ wsag-*

neg:InitiatorAgreementEPR: This OPTIONAL element specifies the endpoint reference to the original instance of the initiator agreement. This element is used in symmetric layouts of the agreement port type. If an Agreement Responder decides to accept an offer for a renegotiated agreement, the state of this agreement instance MUST change to Completed. *wsag-neg:RenegotiationExtension/wsag-neg:ResponderNegotiationEPR*: This REQUIRED element specifies the endpoint reference to the negotiation responder's negotiation instance. Implementations use this reference to identify the negotiation process in which an agreement offer was negotiated. *wsag-neg:RenegotiationExtension/wsag-neg:InitiatorNegotiationEPR*: This OPTIONAL element specifies the endpoint reference to the negotiation initiator's negotiation instance. Implementations use this reference to identify the negotiation process in which an agreement offer was negotiated. *wsag-neg:NegotiationExtension/wsag-neg:NegotiationOfferContext*: This REQUIRED element specifies the negotiation offer context for this agreement offer. It MUST refer to a valid negotiation offer where this agreement offer is a counter offer to. *wsag-neg:RenegotiationExtension/any*: This OPTIONAL element contains domain specific extensions that can be used to realize augmented renegotiation mechanisms.

Similarly we use WS-Policy to articulate policies for the *Policy and Protocol Manager* as show in Figure 3.6 and Figure 3.7. An example of such policy is presented in Sample 3.

5.3 Policy Conversions

The increase of the service market reach and the emergence of different computing environments requires tools that will allow these heterogeneous environments to effectively communicate with each others. We will use the example of SLA creation in this chapter to advocate our approach of using ontologies and semantic web

Sample 3 WS-Policy XML

```
<wsp:Policy ...>
<nb:NegotiationPolicy>
<nb:NegotiationContext> ...
<nb:DesirabilityFactor>0.7</nb:DesirabilityFactor>
<nb:OtherContext> ... </nb:OtherContext>
</nb:NegotiationContext> ...

<nb:Goals>...</nb:Goals>
<nb:Issues>
<nb:Issue>
<nb:Name>Availability</nb:Name>
<nb:Type>Decimal</nb:Type>
<nb:Unit>Percentile</nb:Unit>
<nb:Preference>0.4</nb:Preference>
<nb:Option>
<nb:Name>Gold</nb:Name>
<nb:BestValue>98.9</nb:BestValue>
<nb:WorstValue>99.9</nb:WorstValue>
<nb:ThreshlodValue>99.5</nb:ThreshlodValue>
</nb:Option>
</nb:Issue>
</nb:Issues>

<nb:Constraints><nb:Constraint><wsp:Policy>
        <wsp:All>
            <nb:Condition><nb:Issue>Price</nb:Issue>
                <nb:Operator>&gt;</nb:Operator>
                <nb:Value>40</nb:Value>
            </nb:Condition>

            <nb:Condition><nb:Issue>Availability</nb:Issue >
                <nb:Operator>&lt;</nb:Operator>
                <nb:Value>99.4</nb:Value>
            </nb:Condition>
            <nb:MaxNegTime>10</nb:MaxNegTime>
        </wsp:All>
    </wsp:Policy>

</nb:Constraint></nb:Constraints>
<nb:ConsumerContext>
<nb:Location>Canada</nb:Location>
<nb:EntityType>Company</nb:EntityType>
<nb:Size>Medium</nb:Size>
</nb:ConsumerContext>
<nb:Metadata>
<nb: PolicyName></nb:PolicyName>
<nb:PDate></nb:PDate>
<nb:CustomerInfo>...</nb:CustomerInfo>
<nb:MaxNegTime>...</nb:MaxNegTime>
<nb:DesirabilityFactor>0.7</nb:DesirabilityFactor>
</nb:Metadata>
</nb:NegotiationPolicy>

</wsp:Policy>
```

rules for allowing heterogenous environments to participate in the automated negotiation process and the whole SLA life cycle without the need for human interaction. The same principals are applicable for the protocols and policies in the architecture. Considering the limitation of space we will only discuss the generation of SLAs.

One of the major obstacles in this attempt for automation is the lack of formal semantics associated with the SLA terminology. Customers and service providers need a common language for SLA negotiation, in order to understand each other's offers and bids. Current SLA specifications only define the format of expressing an SLA offer, but the content can use different languages, terms and metrics. To solve the problem, one possibility could be to construct an XML schema definition of SLA metrics, but the general experience is that a semantic definition supports better the re-use, re-combination and translation of descriptive elements. The key concepts of SLA ontology are presented in Figure 5.1.

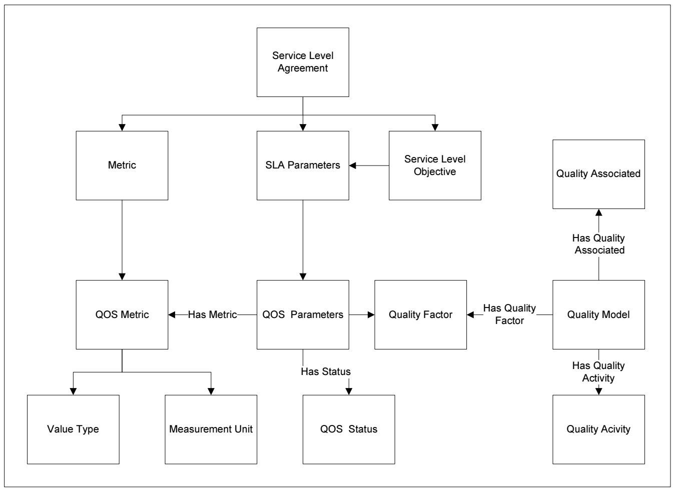


Figure 5.1: Key Concepts of Quality of Service's Service Level Agreement Ontology

The QoS ontology, which is also part of the business ontology, collects the metrics and quality attributes to be used in SLAs. The basic concepts are taken from the quality model defined by OASIS in the Web Services Quality Model (WSQM) specification Kim [2005]. WSQM complements existing SLA-related specifications with a general view on quality related roles, processes and attributes. WSQM uses the term Quality Factor for QoS parameters and further categorizes it into sub-factors and layers concerning the users view, inter-operability and management. We characterize each QoSParameter with an associated Metric which is further characterized by ValueType (float, integer, boolean, etc.), a Value and a MeasurementUnit (e.g. euro, kB, ms). Finally, the QoSParameter can have several statuses depending on if it is requested by a customer, or offered by a provider, etc.

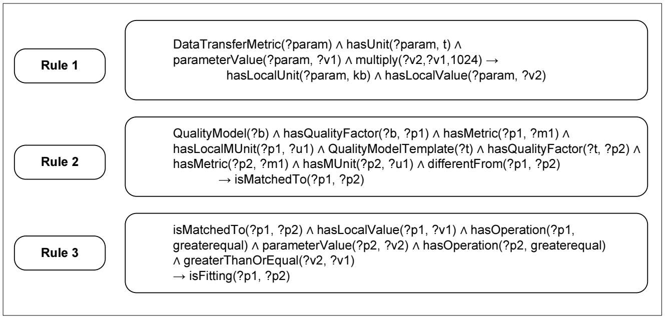
These QoS SLA concepts are not limited to the ones presented above. They could be further extended and expanded based on both domain specific and domain independent requirements. The above mentioned concept only serve as an example. The same concepts and rules are applicable for using any competent of the negotiation process, assuming that we have well defined ontologies for those components.

Negotiation participants can extend the common ontology to consider their internal requirements forming local ontologies. These local ontologies extend the common ontology with locally used parameter types and also with local technological knowledge, which enables a better understanding of SLA requests by the provider in terms of its own local infrastructure. Service providers can add here the definition of locally used QoS parameters, metrics or measurement units. They can also add descriptions about local environment, such as available resource types and their parameters, licenses or platform dependencies. Furthermore, the mapping of received SLAs into the local environment can be supported with such local ontologies, either by new instances (such as conversion rates) or by conversion rules. In the next sections we present examples of these rules and demonstrate that the approach works on available Semantic Web technologies and it provides an adaptable solution without changing program code.

5.3.1 Conversion Rules

In order to improve interoperability between all customers and providers, SLA template files relating their parameters and metrics to a common conceptual model are defined following the common QoS SLA conceptual model. Both the customers requests and providers templates are formalized as WS-Policy (plus WSLA elements).

We can use semantic rules for conversion among compatible types as defined in OWL ontologies. Figure 5.2 show some example of such rules.



Rule 1

```
DataTransferMetric(?param) ∧ hasUnit(?param, t) ∧
parameterValue(?param, ?v1) ∧ multiply(?v2,?v1,1024) →
          hasLocalUnit(?param, kb) ∧ hasLocalValue(?param, ?v2)
```

Rule 2

```
QualityModel(?b) ∧ hasQualityFactor(?b, ?p1) ∧ hasMetric(?p1, ?m1) ∧
hasLocalMUnit(?p1, ?u1) ∧ QualityModelTemplate(?t) ∧ hasQualityFactor(?t, ?p2) ∧
hasMetric(?p2, ?m1) ∧ hasMUnit(?p2, ?u1) ∧ differentFrom(?p1, ?p2)
         → isMatchedTo(?p1, ?p2)
```

Rule 3

```
isMatchedTo(?p1, ?p2) ∧ hasLocalValue(?p1, ?v1) ∧ hasOperation(?p1,
greaterequal) ∧ parameterValue(?p2, ?v2) ∧ hasOperation(?p2, greaterequal)
∧ greaterThanOrEqual(?v2, ?v1)
→ isFitting(?p1, ?p2)
```

Figure 5.2: Sample semantic rules

Rule 1 describes a unit mismatch type of conversion. Where let us say that customer data requests are in Kilo Bytes where as the provider measures that same in Mega Bytes. These are one the most common types of policy mismatches that need to be take care of when participant use langues/communication with different syntax but the same semantics. The same mechanism could be used to define additional, custom rules to provide more complex metric conversions.

Another important task would be to identify matching policies i.e. policies that are aligned and could be converted to the participant's individual protocols. We can use Rule 2 to identify the matching pairs. Then pairs identified by isMatchedTo are evaluated one bye one, and accepted if the offer is stronger than the request, with rules that are similar to Rule 3.

5.4 Study and Results

To determine the applicability and efficiency of our semantic rule based approach for automated negotiation we enhanced our previous work Hashmi et al. [2011] by enriching the architecture with the semantic web rules and compared the results.

5.4.1 Experiment 1

In our previous work Hashmi et al. [2011] we developed a *Norm* operator based GA technique for automated negotiation and performed experiments covering different scenarios. We compared the performance of GA with *Norm* with other methods of solving similar problems. We used 1) a traditional GA with only mutation and crossover operator, 2) a random search and 3) a hill-climber. We used experiments to determine the GA parameters such as population size, number of generations, crossover rate and mutation rate. We repeated those experiments after introducing semantic web rules and compared the results.

Our development environment consisted 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. The client negotiated four QoS components of reliability, availability, throughput and accessibility with multiple providers. We performed 200 iterations consisting of 500 generations each, for all the four algorithms and analyzed the results for efficiency and completeness.

Table 5.4 below shows that average of 200 runs for all four algorithms. We measure the degree of disagreement among the clients requested QoS values. Lower values of degree of disagreement are desired as they show a higher chance of reaching an agreement. The starred rows are the results of the execution runs after enabling Semantic Web Rules in the system.

Table 5.4: Average Results over 200 Iterations For Semantic Rules Based System

	Min	Max	Mean	Std. Dev	Iteration	Time
Ran. Search	0.00568	0.06153	0.02718	0.01663	370	7.239
Ran. Search*	0.00564	0.06143	0.02578	0.01583	360	8.101
Trad. GA	0.00027	0.08547	0.02192	0.02177	437	7.967
Trad. GA*	0.00020	0.08547	0.02001	0.0290	429	9.332
Hill Climber	0.00041	0.05171	0.01461	0.01668	466	8.990
Hill Climber*	0.00032	0.05012	0.01323	0.01535	450	9.842
Norm	0.00002	0.03163	0.00925	0.01157	100	3.210
Norm*	0.00002	0.03012	0.00895	0.01129	99	3.704

The Table 5.4 suggest that the result of the all methods improved when we ran them with the semantic web rules. That is mainly attributed to the fact that now some services that we earlier thought to be incompatible are participating in the negotiation process hence increasing the search space. The addition of Semantic Web Rules suggest that now negotiation process needs more work that would go into the execution and translation of these rules. Hence, we need to look into the execution time overhead and discuss parameters like average GA iteration for the best result and average execution times. These matrices were not important in the earlier analysis and hence were not discussed but now they are certainly of more interest. We can see that although there is a small computation overhead involved in the proposed technique, the benefits outweigh the small performance cost. This can be seen by the fact that Min, Max and Standard Deviation for all the methods improved except the *Norm*. This can be reasoned that the improvement in the degree of disagreement was not significant for the *Norm*operator as it had a very good solution to start with. However, we performed the second sets of experiments to further investigate this behavior.

5.4.2 Experiment 2

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 on Microsoft .Net version 3.5. WSDream-QoSDataset Zhang et al. [2010] 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.

We used our system to negotiate 3 services (*Service A*, *Service B*, *Service C*) where, the negotiation vector consists of four QoS attributes < Throughput, Availability, Reliability, Response Time >. We assume that *Throughput* is the dependent QoS parameter among all the services.

Figure 5.3 shows the results of our experiments. Figure 5.3 shows the utility values of individual service and the composite solution. 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.

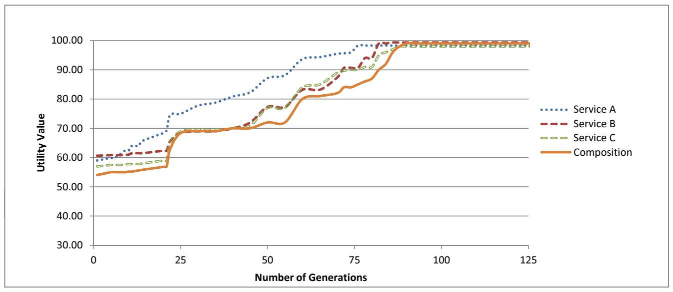


Figure 5.3: Experiment results of service negotiation

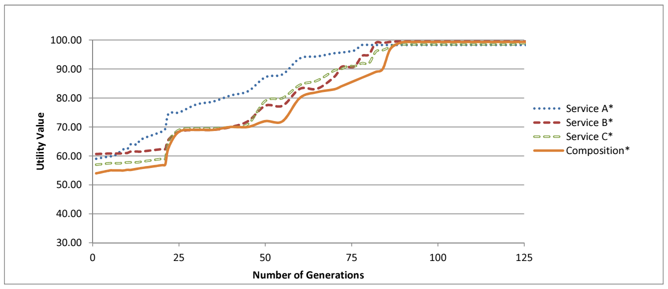


Figure 5.4: Experiment results of service negotiation

Figure 5.4 shows the results of our experiments with Semantic Web Rules. Figure 5.3 shows the utility values of individual service and the composite solution, The negotiated vector for *Service A** is <98,98,98.5,99>, *Service B** is <97.9,98,98,98.3>, *Service C** is <98.2,98.7,99,98> with the utility values of 98.83, 99.50 and 98.90 respectively. Hence the system achieves a maximum utility value of 99.30. We can see that *Service B** and *Service C** improved their values and consequently the composition's utility value also improves. *Service A** on the other hand did not benefit from the the addition of Semantic Web Rules.

5.4.3 Experiment 3.

In this experiment we compared *NegF* and *NegF** (with Semantic Web Rules) 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 Nitto et al. [2007] 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 Niu and Wang [2008] 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 Lecue [2009] 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 6.14. We can see that *NegF** improves on the initial *NegF* solutions out performing similar solutions found in the literature. 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.

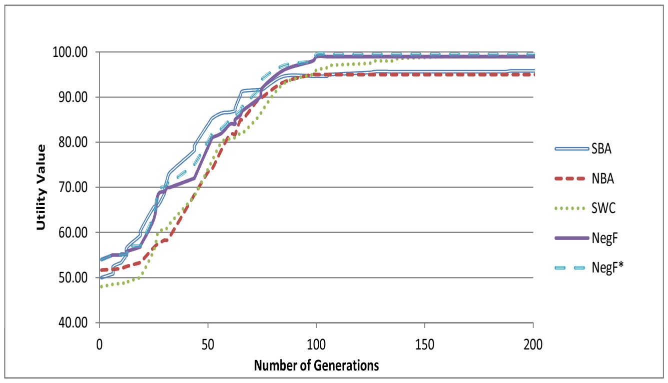


Figure 5.5: Utility value comparison of NegF* with similar service negotiation techniques

5.5 Conclusion and Future direction

In this chapter we have presented different protocols used in our automated negotiation framework. We showed how we enhanced the effectiveness of our existing solutions using Semantic Web Rules and ontologies. We discussed some shortcoming in the existing protocols and their probable solutions. In the end we compared the results of our previous work Hashmi et al. [2011] with our enhanced approach to analyze the effectiveness of our approach. We are currently investigating on making our Semantic Web Rules based approach faster with minimal performance degradation over the traditional approaches. We are also developing ontologies for other components of the system. We are also working on a solution that moves away from the centralized approach in the favor of a more adaptive distributed model.

CHAPTER 6: INVOCATION PATTERNS AND DEPENDENCY MODELING

6.1 Introduction

In the recent years, the Service-Oriented Architecture (SOA) paradigm has gained momentum as a means to develop applications. In SOAs, loosely-coupled software artifacts (commonly referred to as services) may implement specialized functionalities which can be combined with other services from various business partners or public entities into composite services to provide value-added functionality. Two major entities are involved in any SOA transaction: Service consumers, and Service providers. As the name implies, service providers provide a service on the network with the corresponding service description Malik and Bouguettaya [2009e]. A service consumer needs to discover a matching service to perform a desired task among all the services published by the different providers. The consumer binds to the newly discovered service(s) for execution, where input parameters are sent to the service provider and output is returned to the consumer. In situations where a single service does not suffice, multiple services can form a composite system to deliver the required functionality Papazoglou and Hall [2008].

In a running composite system, services may exhibit errors, undergo changes, or become unavailable, and may need to be replaced by other services (to achieve better performance or lower cost). The process of composing/replacing service(s) is a time consuming, error-prone and often a non-optimal process. Using automated composition techniques one could improve upon these results. Apart from the functional properties, within SOAs, Quality of Service (QoS) expresses the non-functional quality attributes of a service, such as the response time, cost, reliability or the supported security protocols etc. The overall performance of a composite system thus depends heavily on how the individual QoS values of the component services effect the composite solution. The orchestration of individual services in a composition also plays a significant role on the QoS values of the composition. Most existing approaches consider this as a local (service level) optimization problem and lack a coherent framework for the specification, and optimization of service compositions focusing on the global (system wide) QoS properties of the system considering the orchestration and dependency relationships among component services.

Services in SOAs are *autonomous*, i.e., they are independently deployed, the topology is *dynamic*, a service can leave the system or fail without notification, the services in an SOA may be invoked using a different invocation model. Here, an invocation refers to *triggering a service (by calling the desired function and providing inputs) and receiving the response (return values if any) from the triggered service.* There are six major invocation relations defined in the literature for service compositions D'Mello and Ananthanarayana [2009] Yu et al. [2007a] Menasce [2004]: Sequential Invocation, Parallel Invocation, Probabilistic Invocation, Circular Invocation, Synchronous Activation, and Asynchronous Activation. A brief overview of these follows.

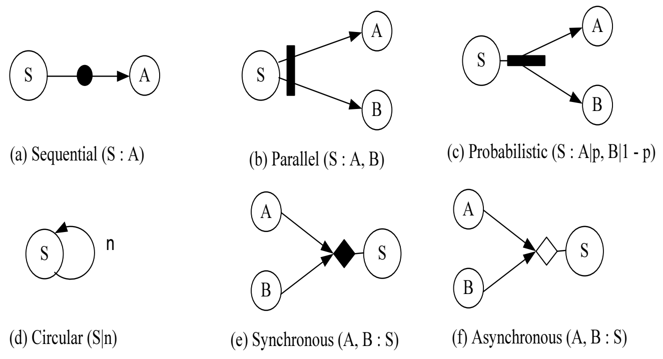

S
A
(a) Sequential (S : A)
S
A
B
(b) Parallel (S : A, B)
S
A
B
(c) Probabilistic (S : A|p, B|1 - p)
S
n
(d) Circular (S|n)
A
B
S
(e) Synchronous (A, B : S)
A
B
S
(f) Asynchronous (A, B : S)

Figure 6.1: Major SOA Invocation Models

Sequential Invocation: In sequential invocation, a service S invokes a service A. It is denoted as Sequential (S : A) (see Figure 6.1-(a)). Sequential invocation is also defined as a serial invocation.

Parallel Invocation: In parallel invocation, a service S simultaneously invokes multiple services. For example, if S has service A and service B which are independent and successors of S, S can invoke both A and B at the same time. It is denoted as Parallel (S : A, B) (see Figure 6.1-(b)).

Probabilistic Invocation: In probabilistic invocation, a service S invokes service(s) with a probability (see Figure 6.1-(c)). For example, if S invokes service A with the probability p and service B with the probability 1 - p, it is denoted as Probabilistic (S : A|p, B|1 - p). The probabilistic invocation is also defined as fork invocation.

Circular Invocation: In circular invocation, a service S invokes itself multiple (n) times. It is denoted as Circular (S|n). A circular invocation can thus be defined as sequentially invoking itself n times (see Figure 6.1-(d)).

Synchronous Activation: In synchronous activation, a service S is invoked only after all of its preceding services that were invoked in parallel have completed their execution. For example, if S has synchronous preceding (invoked in parallel) services A and B, both these services would need to complete before S can progress. It is denoted as Synchronous (A, B : S) (see Figure 6.1-(e)).

Asynchronous Activation: In asynchronous activation, a service S is invoked only after one of its preceding services that were invoked in parallel have completed their execution. For example, if S has asynchronous preceding (invoked in parallel) services A and B, either A or B's completion would cause S to progress. It is denoted as Asynchronous (A, B ; S) (see Figure 6.1-(f)).

It is highly likely that a composite system exhibits more than one type of service invocation. The fact that certain system properties are a composite function of its component services e.g. the overall throughput (transactions per second) of the system is dependent on the component service that has the least transaction per second (bottleneck). Hence, it is important to consider these invocation patterns while considering and modeling the dependencies of the different QoS attributes of the component services. In this chapter we propose a service composition approach that tries to mitigate the fact that different users may experience different QoS value for a given service and hence when the service is composed into a system the observed QoS values could be different from the published QoS values. We use a collaborative filtering based approach to predict the observed QoS values for a service, then use the composition structure to model the dependency modeling relationship among different QoS values of component service and then use a multi-objective Markov Decision Model to formulate a globally optimal solution.

The rest of the chapter is organized as follows. Section 2 gives an overview of Service Oriented Architectures, invocation patterns and presents our motivating example. Section 3 presents our dependency modeling framework. Section 4 discusses the collaborative filtering approach. Section 5 details our multi-objective service composition approach. Section 6 discusses the literature review. In Section 7 we present the results of our experiments and Section 8 presents the conclusion and our future directions.

6.2 Service Composition Optimization Strategies

In this section we look at different service composition optimization goals, and the quantity of information available for the decision making process. It is assumed that all the services under discussion are functionally equivalent and QoS attributes are the primary selection parameters. We start by defining a travel reservation service example to motivate the problem and our proposed approach.

Sample Scenario

Assume that a user wants to attend a conference and needs to make travel arrangement for his journey. He needs to purchase an airline `ticket` and reserve a `hotel` for this travel. Moreover, he needs some `transportation` to go from the airport to the hotel and from the hotel to other venues "Site-seeing". Assume that the user would be using a SOA-based online service that is a *one-stop shop* providing all the five options (airline ticket, hotel, attractions and transportation) through different component services.

The online reservation system provides many services such as: attraction service which are outsourced to three services (representing individual services): Art, Museums, and Area tours. This service provides arrangement to visit different areas through sub-contractor companies. For clarity, Figure 6.2 shows the different options. User may select Art, Art and Museum, Art and Area tour, Museum or Area tour. The transportation service also works through subcontractors that provide car, bus or bike. These companies provide different services based on the distance between the places user plans to visit (user has the option to choose the mode of transportation). The system also provides services to calculate the distance between two points. The system employs two other services: one for airline ticket, and the other for hotel reservation.

Figure 6.2 shows a sample structure of the travel reservation scenario. This structure depicts a combination of different invocation models that may be used to

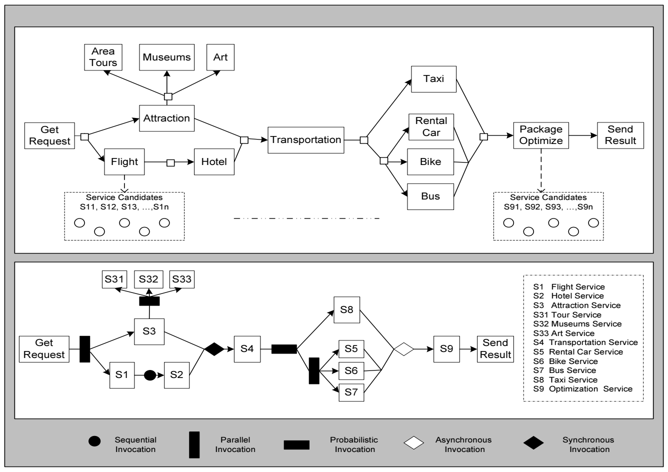


Figure 6.2: Scenario with Invocation Models

compose the system. Since the user is looking for a travel arrangement that include: booking a ticket, booking a hotel, transportation (car, bike or bus) based on the distance between the places and visiting some attractive places, these services can be invoked in parallel. Booking a ticket and find attractions is an example of *parallel invocation*. There are three choices that provide attraction (Area tour, Museums and Art). Since user has to choose among these service instances, this is an example of *probabilistic invocation*. Similarly car, bike and bus services can be classified as probabilistic invocations. When the system provides the results of transportation to the reservation service, for generating a discount, this invocation is an example of *asynchronous invocation*. Finally, *synchronous invocation* appears when *package optimization* service waits for other services (hotel, ticket and attraction) to compute the final trip cost.

Local Optimization

The most basic scenario for service selection deals with the local resources and constraints. In the local optimization approach, the selection of a component

Web service that performs a given task of the composite system is done as a stand alone component without considering any system-wide constraints or limitation of other components of the composition. When the composite system wants to perform a certain task (as a part of initial system composition or replacing a faulty service during execution process) the system gathers QoS information for each candidate Web service. After collecting the QoS information, the system constructs a QoS vector which is then evaluated to calculate the usefulness of each service. The candidate services are then ranked based on their usefulness or utility and the best available service is selected. This selection process could include user defined weights and constraints such as execution time, availability etc.

In our running scenario from Figure 6.2 assume that the system is considering candidates S11, S12 and S13 for the `flight` service. Assume that the QoS vector used to make the decision consists of four QoS attributes <*Throughput, Availability, Reliability, Execution Time*>. For the sake of simplicity, we assume that the service prices are static and all the above QoS attributes are equally important for the system. Assuming that the QoS vector for S11 is <91,95,95,88>, for S12 it is <94,92,92,88> and S13's vector is <90,98,98,90>. For local optimization the system will select service S13. Although S11 has a lower value for execution time but S13 has much higher values for Availability and Reliability. The system is able to make this decision since it has all the local information regarding the candidate services to compare and make an informed decision. The selected service was the best possible choice in the local context i.e. local maxima.

Global Optimization

Since local optimization is performed on a per service basis, it is possible that it may not be the optimal choice in the context of the composite system. In Figure 6.2 let us assume that `hotel` service has a QoS vector of S2 $<$ 93,95,95,92$>$. One of the system wide constraints could be that both `flight` and `hotel` service should finish their execution in 180 ms. In order to meet this constraint, the system cannot choose service S13 $<$90,98,98,90$>$ since it does not satisfy the system constraint (i.e. $92 + 90 = 182 > 180$). Hence, it would have to select S11 $<$91,95,95,88$>$ which is clearly not the best local choice but turns out to be the best global choice for the composite system. Global optimization is thus heavily dependent on the amount of system wide information and constraints for optimal decision making. Similarly, if we add another dimension of information, e.g. that we are looking to optimize the throughput (after meeting the execution time constraint) then we can see that service S12 $<$94,92,92,88$>$ will be a better choice since it has a higher throughput value; although it was the second best choice in the previous scenario, and the last choice in local optimization.

Let us look at another case of global optimization in our running example by adding structural information to our current discussion of service selection. Assume that we have two options for `Attraction` service: S3a $<$98,95,95,180 $>$ and S3b $<$93,96,96,180$>$. Here service S3a is a better choice for `Attraction` service since it has far better throughput and comparable values of other QoS components. Now if we look at the structure of our system in Figure 6.2 we see that `Attraction` service is invoked in parallel with `Flight` and `Hotel` services which in turn, are sequential in nature i.e. Parallel (Get Request: S3, Sequential (S1:S2)). We can see that based on our current selection of S2 $<$ 93,95,95,92$>$ for `hotel` and S12 $<$94,92,92,88$>$

for `flight` the maximum throughput for the current composite solution with the Sequential invocation (S12:S2) is 93 (min (93,94)). This, in turn implies that the maximum throughput of the current composite solution with the Parallel invocation (Get Request: S3, Sequential (S12:S2)) cannot exceed 93 i.e. (min (98,93)). Based on this information our choice of S3a <98,95,95,180 > for `hotel` service is not a global optimal service for the composite system as S3b <93,96,96,180> offers better QoS values for availability and reliability while still matching the maximum throughput value of 93 for the parallel invocation of the scenario. Hence, the service selection framework could benefit from the invocation pattern information for making optimal decisions to improve the overall QoS of the composition. To differentiate, we refer to such relationships among the same QoS components of different service of the composition as *dependency modeling* in the rest of the chapter.

6.3 Dependency Modeling Framework

In order to demonstrate our proposed approach for QoS dependency modeling we will be using the following quality components (note that other QoS attributes can be modeled in a similar manner).

- *Cost* The cost $q_{co}(s, f_i))$ of an operation f_i of a service s is cost incurred (fee paid to the service provider) to invoke an operation in order to perform a particular task. These values are either published by the service providers or are available on demand. The cost is measured as a per invocation unit.

- *Execution Time* Given an operation f_i of service s, the execution time $q_{et}(s, f_i)$ measures the difference between the time when the request is sent and the time when the results are received. The total execution time is the sum of the processing time $T_{process}(s, f_i)$ and the transmission time $T_{trans}(s, f_i)$. It is measured in milliseconds.

- *Reliability* The reliability $q_{re}(s)$ of a service s is measured as the probability of a system to accurately respond to a request (i.e., the operation is completed and a message indicating that the execution has been successfully completed is received by a service requestor) within the maximum expected time frame indicated in the Web service description. The reliability (or success rate) is dependent on the hardware and/or software configuration of Web services and the network connections between the service requesters and providers. The value of the reliability can be computed from data of past invocations using the expression $q_{re}(s) = (K - N_f(s))/K$, where $N_f(s)$ is the number of faulty executions of the service s, and K is the number of times service s has been invoked.

- *Availability* The availability $q_{av}(s)$ of a service s is the probability that the service is accessible. The value of the availability of a service s is computed using the expression $q_{av}(s) = T_a(s)/\theta$, where T_a is the total amount of time for which service s is available over and observed θ amount of time. Availability is usually measured in the percentage. The higher the value is, the more a system is available.

- *Throughput* The throughput $q_{th(s)}$ of a service s is measured as the number of completed requests per unit amount of time.

Table 6.5 shows how to calculate different QoS values for major SOA invocation models as discussed above.

Table 6.5: QoS values for different invocation models

	Sequential	Probabilistic	Circular	Parallel Synchronous	Parallel Asynchronous
Cost	$\sum_{i=1}^{n} q_{co}(f_i)$	$\sum_{i=1}^{n} p_i * q_{co}(f_i)$	$q_{ac}(f) * c$	$\sum_{i=1}^{n} q_{co}(f_i)$	$\sum_{i=1}^{n} q_{co}(f_i)$
Execution Time	$q_{et}(s, f_i) = T_{process}(s, f_i) + T_{trans}(s, f_i)$	$\sum_{i=1}^{n} q_{et}(s, f_i)$	$q_{ac}(s, f_i) * c$	$max(q_{et}(s_1, f_i), ..., q_{et}(s_n, f_i)))$	$max(q_{et}(s_1, f_i), q_{et}(s_2, f_i), ..., q_{et}(s_n, f_i)))$
Reliability	$\prod_{i=1}^{n} q_{re}(f_i)$	$\sum_{i=1}^{n} p_i * q_{re}(f_i)$	$q_{re}(f)^c$	$\prod_{i=1}^{n} q_{re}(f_i)$	$max(q_{re}(f_1), q_{re}(f_2), ..., q_{re}(f_n)))$
Availability	$\prod_{i=1}^{n} q_{av}(f_i)$	$\sum_{i=1}^{n} p_i * q_{av}(f_i)$	$q_{av}(f)^c$	$\prod_{i=1}^{n} q_{av}(f_i)$	$max(q_{av}(f_1), q_{av}(f_2, ..., q_{av}(f_n)))$
Throughput	$min(q_{th(s_1)}, q_{th(s_2)}, ..., q_{th(s_n)})$	$\sum_{i=1}^{n} p_i * q_{th}(s, f_i)$	$q_{th}(s)$	$min(q_{th(s_1)}, q_{th(s_2)}, ..., q_{th(s_n)})$	$max(q_{th(s_1)}, q_{th(s_2)}, ..., q_{th(s_n)})$

For the composite system we would need to calculate the over all QoS values suitable for negotiation for the new requested service. This is mainly dominated by the structure of the composition. Apart from the simple invocation methods mentioned above we also need to look into complex invocation patterns. One of the key elements of a composite system is a complex loop. A complex loop can be defined as a *Circular Invocation* with a linear or non-linear complex execution path. Loops may contain different combinations of *Invocation Patterns* i.e. nested loops, probabilistic invocations etc. Hence, the QoS values of a complex loop structure could be calculated by the probability of number of iterations of the loop i.e. p_c, the probability of the exiting the loop p_e and QoS values of the execution path of the loop. We can always flatten a loop structure into repeatable blocks of linearly executing patterns. A loop may have multiple entry and exit points. We can assume that the actual entry and exit points of a loop structure could be ignored for the ease of calculation as they have minimal affect on the actual QoS values. However, the probability of exiting out of the loop p_e along with the individual probabilities of the different execution patterns within the loop are required for the QoS calculations.

Let us assume that the transition probability of execution of Service s_{i+1} after executing Service s_i is denoted by p_i (in a linear execution path this will always be equal to 1). The probability of exiting a loop after executing Service s_i is denoted by $p_{e_{i,j}}$ (in a linear execution path this will be 0) where $p_i + \sum_{j=1}^{m} p_{e_{i,j}} = 1$ and $i \in [1, n]$. The cost of the loop is depicted by c_i and the total execution time is shown as t_i. After x executions of the loop where $x \in [0, +\infty]$ the loop can been transformed into x linear executions paths for the component services and we already know that the probability of executing the next service is $p_{e_{i,j}}$ where $j \in [1, m]$. So the probability for the combination of service for the execution for x where $x \in [0, +\infty]$ time and the execution of any service after Service S_1 is $(\prod_{i=1}^{n} pi)^x p_{e_{1,j}} (x \in [0, +\infty], j \in [1, m])$. Similarly the probability for executing Service S_{k+1} after executing Service S_k where $k \in [2, n]$ is $(\prod_{i=1}^{n} pi)^x (\prod_{i=1}^{k-1} pi) p_{e_{1,j}} (x \in [0, +\infty], j \in [1, m])$. Let $p_0 = 1$ then the probability of executing a service after Service S_k is

$$p'_{e_{k,j}} = \sum_{x=0}^{+\infty} (\prod_{i=1}^{n} p_i)^x (\prod_{i=0}^{k-1} p_i) p_{e_{k,j}} \tag{6.1}$$

$$= \frac{(\prod_{i=0}^{k-1} p_i) p_{e_{k,j}}}{1 - \prod_{i=1}^{n} p_i} where (k \in [1, n], j \in [1, m]) \tag{6.2}$$

Now the probability that service $S_1 ... S_n$ is executed x times and then the loop is terminated at service S_k is $(\prod_{i=1}^{n} p_i)^x (\prod_{i=0}^{k-1} pi)(1 - p_k)$. The cost for this execution accumulated by the services will be $x \sum_{i=1}^{n} c_i + \sum_{i=1}^{k} c_{i'}$ and the execution time will be $x \sum_{i=1}^{n} t_i + \sum_{i=1}^{k} t_{i'}$. The reliability for this execution will be $(\prod_{i=1}^{n} q_{re(i)})^x (\prod_{i=1}^{k} q_{re(i)})$. Hence we get the following equations for Cost (q_{co}), Total Execution Time (q_{et}), Reliability (q_{re}) and Availability (q_{av}) respectively.

$$q_{co} = \sum_{k=1}^{n} \sum_{x=0}^{+\infty} \left((\prod_{i=1}^{n} p_i)^x (\prod_{i=0}^{k-1} p_i)(1 - p_k)(x \sum_{i=1}^{n} c_i + \sum_{i=1}^{k} c_i) \right) \tag{6.3}$$

$$q_{et} = \sum_{k=1}^{n} \frac{(\prod_{i=0}^{k-1} p_i)(1 - p_k)(\sum_{i=1}^{k} t_i + \prod_{i=1}^{n} p_i \sum_{i=k+1}^{n} t_i)}{(1 - \prod_{i=1}^{n} p_i)^2} \tag{6.4}$$

$$q_{re} = \sum_{k=1}^{n} \sum_{x=0}^{+\infty} \left((\prod_{i=1}^{n} p_i)^x (\prod_{i=0}^{k-1} p_i)(1 - p_k)(\prod_{i=1}^{n} r_i)^x \prod_{i=1}^{k} r_i) \right) \tag{6.5}$$

$$q_{av} = \sum_{k=1}^{n} \frac{(\prod_{i=0}^{k-1} p_i)(1-p_k)\prod_{i=1}^{k} r_i}{1-\prod_{i=1}^{n}(p_i * r_i)} \tag{6.6}$$

$$q_{th} = \sum_{k=1}^{n} \frac{(\prod_{i=0}^{k-1} p_i)(1-p_k)\prod_{i=1}^{k} r_i}{1-\prod_{i=1}^{n}(p_i * r_i)} \tag{6.7}$$

After calculating the individual QoS values for each invocation pattern we now need to calculate the max possible QoS value for each component of the QoS vector in order to establish a target QoS value that would be used for global optimization. Since our QoS calculation follow a single entry and single exit format we can use the refined process tree approach Vanhatalo et al. [2009] to find the different execution paths and apply our previously described invocation patterns. In this regard, Algorithm 6.3 calculates the individual QoS values for all components of the QoS vector at a given node. In this algorithm we use depth first on all the nodes and create tree for all the path to the leaf nodes. Once we get to the leaf node we recursively backtrack and calculate the QoS value for that path. If we run into a conditional branch we add both the paths as child node and this allows us to still find all the possible execution paths for the composition. We need to focus on the QoS value for the paths that contain the service that we want to replace. This will ensure that for global optimization we are only considering the relevant paths. Initially, we will set the QoS values for service to be replaced at the maximum possible values to calculate the target QoS negotiation values for the service (refer to section 6.2). We can see that the proposed depth first algorithm has low time and space complexity. With N nodes and E edges the first exploration of the graph takes $O(|V|+|E|)$ and in the second exploration for the graph we only visit all nodes once, (since the QoS calculation at every invocation point takes a linear amount of time) our execution takes $O(|V| + |E|)$ time.

Algorithm 1 QoS Calculation for a service execution

```
 1: DepthFirstQoS ( Node N, Tree F )
 2: T = create a new tree;
 3: Add T as child of F;
 4: if N has no children then
 5:     Get a path to parent node of N
 6:     for All component of QoS vector do
 7:         Apply the QoS pattern and multiply it with the probability to get the QoS value at current invocation point.
 8:     end for
 9: end if
10: for each child node of N do
11:     if isVisited = true then
12:         Add all the child nodes to T
13:         if current node = leaf node then
14:             Get a path to parent node of T
15:             for All component of QoS vector do
16:                 Apply the QoS pattern and multiply it with the probability to get the QoS value at current invocation point.
17:             end for
18:         else
19:             Call DepthFirstQoS(current node, T)
20:         end if
21:     end if
22: end for
```

Figure 6.3: QoS Calculation for a service execution

Once we determine the dependency relationship we need to be able to predict the QoS values that we may be able to get from our composition. That could be done by predicting the observed QoS values of individual services and then using them in the dependency calculation. We present our collaborative filtering approach in the next section that is used for this purpose.

6.4 Collaborative Filtering

In ideal situation Web services should meet all the published performance criteria for all of its users. However it is rarely the case. We use a collaborative filtering based approach to mitigate the discrepancies between published and observed QoS values for the Web services. We can loosely classify the Web service attributes into two broad classes of certain and uncertain attributes. The value of certain attributes remains fixed or unvarying for all the users i.e. service name, service provider, cost (publisehd/mutually agreed upon). The value of uncertain attributes on the other hand may change or vary from user to user, or even for the same user, from one invocation to the other invocation, mostly based on the environmental factors. These attributes have a marked tendency to deviate from their published values. Response time is one such example of an uncertain attribute. Many factors can influence the response time of a service e.g. internet speed, network congestion, slow hardware. These factors are usually out of the hands of the service provider. These uncertain attributes may influence the perceived overall QoS value of the systems. However, most of the service selection algorithms do not take the uncertainty of these attributes into account which leads to inaccurate/sub-optimal service matching, these system only consider the published QoS values of the services. However, as discussed above the observed QoS value may differ from the published QoS values for a user. It is well accepted that it is impractical (and nearly impossible) for a service consumer to invoke all the services (under consideration) to record their observed QoS values and then use them for service composition. We can instead utilize the experience of other (similar) users to predict the observed QoS values for services and use them as a metric for service composition. This requires us to investigate an approach that first of all would be able to search users that are similar (in terms of environment) to the current service consumer and have invoked the service under consideration with reported observed QoS values for the service. Secondly, use the experience of similar users to predict observed QoS values for the service consumer. These predicted QoS values could later be used in the negotiation process to minimize the affect of uncertainty of QoS values of the services. Note that it is common for the end user to receive lower values of QoS components than the promised/published QoS values (observed response time of 98ms as compared the published response time of 60ms) and very rare if not highly improbable to experience better than published QoS values. Hence, it is safe to assume that using observed QoS values will not over compensate a service provider for its published QoS values which in turn, could result in over promise of QoS to the consumer.

We use collaborative filtering on the observed QoS values to solve the problem of uncertainty. The two most commonly used memory based collaborative filtering approaches are Pearson correlation and Vector cosine based similarity. Pearson coefficient McLaughlin and Herlocker [2004] is symmetric, invariant to scale and location of variable and can also detect negative correlation making it suitable for our problem at hand. The similarity between two users i and j can be calculated using Pearson Correlation Coefficient as:

$$sim(i,j) = \frac{\sum\limits_{k}(v_{i,k} - \bar{v_i})(v_{j,k} - \bar{v_j})}{\sqrt{\sum\limits_{k}(v_{i,k} - \bar{v_i})^2}\sqrt{\sum\limits_{k}(v_{j,k} - \bar{v_j})^2}} \tag{6.8}$$

where $v_{i,k}$ denotes the QoS value that user i received from service k, v_i denotes the average QoS that user i received from all the services invoked, and the summation is over all the services that have been invoked by both i and j.

In the real life scenarios we observe that it is rather rare to have a consumer that has a larger number of similar consumers or a service that has large number of similar services. Hence, we will be working with a very limited data set. The prediction confidence could be increased if we use both content and user based similarity to predict the observed QoS values. In our approach when we have a a missing QoS value for a service and we do not have any similar users that have invoked that particular service, we find out similar services and use their QoS values to predict the missing QoS value, and vice versa. We assume that $S_i \neq \emptyset$ and $U_p \neq \emptyset$. We use f_u and f_i to assign weightage to both the filtering values.

$$f_u = \sum_{i \in S_i} \frac{Sim(i,j)}{\sum_{i \in S_i} Sim(i,j)} \times Sim(i,j) \tag{6.9}$$

We use $\lambda(0 \leq \lambda \leq 1)$ to adjust the influence of both user(u) based and content based(i) filtering.

$$\begin{aligned} W_u &= \frac{f_u \times \lambda}{(f_u \times \lambda) + (f_i \times (1-\lambda))} \quad and \\ W_i &= \frac{f_i \times \lambda}{(f_i \times \lambda) + (f_u \times (1-\lambda))} \end{aligned} \tag{6.10}$$

where $W_u + W_i = 1$, hence the predicted QoS value would be

$$\begin{aligned} Q_{val} = W_u \times (\bar{v}_i + \frac{\sum_{j \in S_i} Sim(i,j)(v_{j,p} - \bar{v}_j)}{\sum_{j \in S_i} Sim(i,j)}) + \\ W_i \times (\bar{v}_q + \frac{\sum_{q \in U_p} Sim(p,q)(v_{p,q} - \bar{v}_q)}{\sum_{j \in U_p} Sim(p,q)}) \end{aligned} \tag{6.11}$$

Using the above mentioned approach implies that the initial set of services that are being considered for selection process have a very high chance of forming a service agreement. The observed and predicted QoS values help eliminate the uncertainty in perceived QoS values. The next step is to formulate a globally optimal composition using the dependency modeling and the services with better predicted observed values.

6.5 Multi-Constraint Modeling

The initial filtering process provides us with a good platform of a subset of services to be considered for service composition. For service composition we use the concept of Markov Decision Process (MDP). MDP is an Artificially Intelligent model for making decisions in environments where there is a higher percentage of uncertain outcomes. MDP has been successful applied in different domains to solve decision making problems Mastronarde et al. [2013] Carter et al. [2014] Pesce et al. [2014] Patrick [2012]. We model each dependency relationship as a single constraint and use this model to solve the global optimization dependency problem for QoS parameters of component Web services.

In our system each Web service has a unique identifier ID and a QoS vector. Each QoS vector is a combination of five QoS values i.e. QoS = <Cost, Execution-Time, Reliability, Availability, Throughput> (it can easily be generalized to more QoS component values since $QoS =< QoS_1, QoS_2, .., QoS_n >$). Markov Decision Process is defined as $< S, A, P, R, \gamma >$. S represents the set of states of the system; A stands for the set of actions in the system and $A(s)$ is a subset of action available at state s i.e. $A(s) \in A$ where $s \in S$; P is the probability of an action such that $P_a(s, s') = Probability(s_{t+1} = s' \mid s_t = s, a_t = a)$; R is the expected or immediate reward for the current transition and γ is used to factor in the importance of current reward and the future rewards. We can extend this basic single constraint MDP to a multi-constraint MDP by enhancing the reward function to consider the multiple constraints. We modify the reward function to compensate for multiple constraints. Let us assume that the system has c number of constraints. The reward function will be as follows:

$$R_a(s,s') = [R1_a(s,s'), R2_a(s,s'), ..., Rc_a(s,s')]^T \tag{6.12}$$

In addition to the above constraints we need to consider the fact that every composite system has an entry and exit point. We can argue that this entry and exit point may contain more than one service. Some systems can be invoked using multiple services and similarly their execution may take multiple routes to reach different end states. Hence, the updated model for Web service will have 7 tuples $MDP =< S, S_s, S_e, A, P, R, \gamma >$. Where S_s is a set of starting states for the composition and S_e is the set of end states of the composition where $S_s \in S$ and $S_e \in S$. Here an action corresponds to the execution of a Web service and the reward vector contains one reward for every QoS component of the Web service. For Web service ws and our QoS attributes under consideration the reward vector will be:

$$\begin{split} ws(q)(s,s') = [ws(q_{co})(s,s'), ws(q_{et})(s,s'), ws(q_{re})(s,s'),\\ ws(q_{av})(s,s'), ws(q_{th})(s,s')]^T \end{split} \tag{6.13}$$

The solution of MDP is a decision constraint and a constraint is the procedure for selecting Web services. These constraints, represented by ζ are basically just mappings from states to actions, defined as $\zeta : S \rightarrow A$. Each constraint can represent a single composite solution of Web service. Hence, our system searches for a set of Pareto optimal constraints which optimize QoS attributes of the composite system. The set ζ^o of Pareto optimal constraints is defined by:

$$\zeta = \left\{ \zeta^O \in \coprod \mid \nexists \zeta \in \coprod, s.t. V^{\zeta^O}(s) >_o V^{\zeta}(s), \forall s \in S \right\} \tag{6.14}$$

where $\coprod$ defines the set of all constraints and dominance relation is represented by $>_o$. For $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_n), a >_o b$ means that $a_i \geq b_i$ is satisfied for all i and $a_i > b_i$ for at least one i. Moreover $V^{\zeta}(s) = (V_1^{\zeta}(s), V_2^{\zeta}(s), ..., V_m^{\zeta}(s))$ is the value of the vector s as per the constraint ζ i.e.:

$$V^{\zeta}(s) = E_{\zeta} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\} \tag{6.15}$$

where E_{ζ} represents the expected value when the service follows constraint ζ, s_t at time t with the reward vector r_t.

Q-learning is calculated as:

$$Q_{\zeta}(s, a) \models E_{\zeta} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\} \tag{6.16}$$

6.5.1 Single constraint Multi-objective Service Composition

In our approach we use Q-learning for QoS objectives. We spawn a Q-learning service for every QoS objective. Based on the multiple objective criteria the importance (weight) of every QoS objective for a Web service is learned rather than using predefined weights. Every service i selects a Web service ws_i at each state S in such a way, that optimizes the relative QoS objective of the Web service. Once this process is completed then the Web service assign a weight $Wi(s)$ to their selected service and later negotiate among themselves to select the most suitable candidate to be executed at each state. The service having the maximum weight will be able to execute its option at each state. The objective is:

$$W_k(s) = Max_{i\in 1,....,n} W_i(s) \tag{6.17}$$

Therefore, in this case the Web service k is called the leader service and is allowed to invoke Web service $W_k(s)$. In the next round the Web services evaluate the results of the previous selections and adjust their $w_i(s)$ values based on positive or negative outcome of the previous round. Hence, we may have a different leader service in the next round.

W represents the difference in the predicted versus actual reward received by the Web service. Web services predict to receive a reward value P if their selected service was executed at s. If their service was not executed then instead of receiving the predicted reward P it receives the actual reward A. So W = P - A. In the case when a service's suggested selection was executed P = A and if not then the service will receive a negative reward that is equal to (P - A). So if service k ends up being the leader in a certain round then all service except k will update their W values using the following:

$$W_i(x) \to (Q_i(x, a_i) - (r_i + \gamma max_{b\in a} Q_i(y, b))) \tag{6.18}$$

At this point the next state "s'" and the reward vector r_i is influenced by the leader service rather than being a decision of each individual service.

6.5.2 Multiple constraint Multi-objective Service Composition

In our second approach we introduce the concept of convex hull to solve the multiple constraint problem for service selection for composite solutions. The convex hull is defined as the smallest convex set that contains all the points that lie on the boundary of this convex set. We combine this concept, which is similar to the Pareto front, into our Q-learning based approach based on the fact that both concepts are essentially maxima over different trade-off factors in the linear domains.

We use a value iteration based method to obtain a set of service selection constrains that is Pareto optimal:

$$V(s,a) = V(s,a)(1-\alpha) + \alpha \left[\gamma \mathcal{R} \bigcup_{a'} V(s',a') + r(s,a) \right] \tag{6.19}$$

$V(s,a)$ represents the set of vertices of the $\mathcal{R}$ when action a is taken at state s for all possible Q-value vectors , r is the reward vector, discount value for the process is represented by γ, rate of learning is controlled by α and the operator $\mathcal{R}$ represents the set of extracted vertices of the $\mathcal{R}$.

Our solution follows the greedy exploration methodology and the dominance relationship between Q-vectors is used for selecting a particular action. In this approach we do not backtrack based on the maximal expected reward for each vector rather we use the set of maximal expected rewards for the given set of constraints as the basis for backtracking.

6.6 Literature Review

QoS based service selection has been an active research topic for service composition. Most of the research in this area has been focused on local optimization of QoS attributes that may not result in a globally optimal solution Booth et al. [2004] Curbera et al. [2002b] Chinnici et al. [2007]. Some early efforts of global optimization using integer programming, heuristic based searches and critical path have been presented in Zeng et al. [2004a], Aggarwal et al. [2004], Berbner et al. [2006], Ardagna and Pernici [2007], Ai et al. [2011].

A mixed integer programming based global optimization approach has been presented in Alrifai and Risse [2009]. In this approach authors decompose the global constraints into local constraints by mapping them to a set of pre-computed local QoS values. This approach provides locally efficient component services that are then combined to formulate the composition. The major shortcomings of this approach lies in the fact that all QoS dimensions are considered independently and no dependency or correlations among these components is taken into consideration. Secondly, in some scenarios where the QoS requirements are very aggressive, the approach translates the global requirements into very constrained local requirements and resultantly the algorithm fails to finds a solution where as a solution adhering to the global constraints may exist in the system.

Canfora et al. Canfora et al. [2008] proposed a Genetic Algorithms (GAs) based QoS-aware composite service binding and rebinding approach. This approach allows the service orchestrators to apply non-linear QoS aggregation formulae as compared to linearization for the traditional approaches. However, their solution focuses on collecting usage pattern data to predict the need of re-binding for different invocation instances. This increases the system overhead and the service needs to be a part of the composition for it to be optimized. Secondly the proposed solution assumes that there will always be a solution that will meet all the requirement. Yu et al. Yu et al. [2007b] present a broker based solution for QoS based service selection. They present user-defined utility function for both of their models i.e. combinatorial and graph based model. They consider invocation patterns in their utility functions and present different heuristic based approaches to find near optimal solutions. However, they present two different solutions for sequential executions and complex structures i.e. loop etc and rely solely on the published QoS values. Zeng et al. Zeng et al. [2004b] present an Integer Programming based solution that finds optimal service composition solutions. However, incase of multi-path scenarios they only optimize the solution based on the path with the highest probability as well as their definition of critical path uses the worst case scenarios of all the service in the execution path, which may not be a good approximation and hence is not suitable for large systems or systems with dynamic service needs.

In Ivanovic et al. [2013] authors propose a discrete probability distribution based model to solve the uncertainty of QoS values of the services. They incorporate composition control along with probability distribution on a uniform framework. The proposed approach works better than either using the mean or median and is independent of the any normality or uniformity constraints. This allows to directly convert the observed behavior in inputs for the system and a single analysis could approximates the results that could have been obtained by exhaustive black box simulations of the composition. This method however does not consider the dependency modeling behavior of the system and assumes that the probability distribution is independent of the factors like location of users and any other differentiating factors. In Zemni et al. [2010] present a soft constraint based solution for QoS service selection. They model selection characteristics using soft constraints, assign preferences to the penalties and constraints. Then the cumulative ranking of the above mentioned factors to compute the composite rank of the solution. This allows the users more flexibility incase services do not meet the exact composition criteria. However, the proposed approach does not consider the effect of dependency modeling and leaves it to the user to use a simple ranking based solution.

Collaborative Filtering (CF) approaches i.e. both memory based and model based algorithms have been widely used in recommendation systems Burke [2002], Linden et al. [2003], Ma et al. [2007], Resnick et al. [1994]. Memory based approaches use the historic values of a user to recommend similar items Krzywicki et al. [2015], Jin et al. [2004] on the other hand model based approaches Xue et al. [2005],Hofmann [2003], Hofmann [2004], Hofmann and Hartmann [2005] apply different machine learning and statistical methods to learn user rating behaviors. Memory based approaches are relatively easier to implement, do not need any training set and can easily accommodate new rating data from users. However, they do not scale well as the data set grows. On the other hand item based approaches usually outperform memory based approaches in terms of scalability and quality of recommendation but they suffer from the fact that the model needs to be updated/ rebuild when new data points are received. There have been efforts to use collaborative filtering in service composition. Margaris et al. Margaris et al. [2015] use collaborative filtering for QoS requirements for WS-BPEL scenarios. They combine the collaborative filtering based score and the QoS requirements of the users to calculate an aggregate score for WS-BEPL scenarios. Karta Karta [2005] have used both Pearson correlation and vector based similarity approaches in collaborative filtering on MovieLens data set for service selection, comparing their work with multidimensional recommender system. However, this approach uses static QoS ontology and ranking registry data.

6.7 Study and Results

To study the efficiency of our approach we implemented our motivational scenario and ran multiple simulations. We compared the single constraint and multi constraint approaches along with our invocation patterns and collaborative filtering based solution. Moreover, We compared our approach with similar approaches in the literature. The experiment environment consists of a Windows server 2008 (SP2)-based Quad core machine with 8.0 GB of ram. We used WSDream-QoSDataset Zhang et al. [2010] , which contains multiple 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 assign cost values to them.

6.7.1 Experiment 1 : Single Constraint Algorithm

In the first experiment we compared the single constraint algorithm with the Q-learning approach Wang et al. [2010]. We compare the effectiveness of our approach in formulating a composite solution with no pre-defined user weight preferences against the user defined weight vector for the Q-learning approach. We use the accumulated reward for a composition to compare the quality of the discovered solution. The Q-learning approach uses a weight vector of $\omega = (q_{co} = 0.2, q_{et} = 0.2, q_{re} = 0.2, q_{av} = 0.2, q_{th} = 0.2)$ for this experiment i.e equal weightage for all the QoS components. Fig. 6.4 shows the results of our experiment of average results of 30 runs of both the algorithm with varying number of Web services. We can see that the single constraint approach out performs the Q-learning approach regardless of the number of Web services. The difference in quality of solutions increases when the number of Web services increase. This is attributed to the fact that our approach scales better at exploring the Pareto front than the Q-learning approach. Secondly, pre-defined weights guide the search process for the Q-learning based approach with may not be the best case solution since learning the weights on the fly could find solutions in other wise unexplored regions.

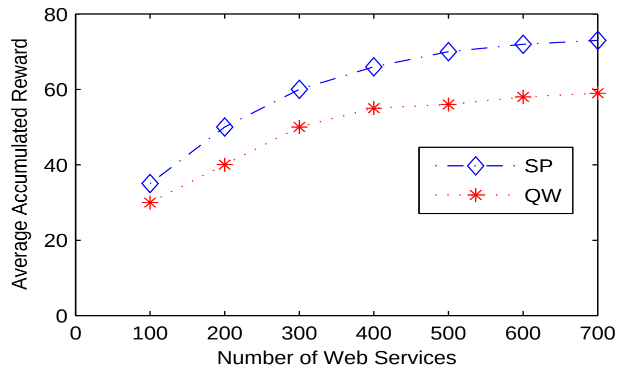


Figure 6.4: Single Constraint Algorithm

6.7.2 Experiment 2 : Multi Constraint Algorithm

In this set of experiments demonstrate that our algorithm finds Pareto optimal solutions considering the dependency relationship among the different QoS components. We assign multiple concrete service (50 and 100) respectively to the abstract services and observe the proposed solutions. In the first experiment we assigned 50 concrete Web services to every abstract component Web service. We use three QoS attributes i.e. Cost, Availability and Response time. The objective is to minimize cost and response time while maximizing the availability of the system. Figure 6.5 shows the results of pareto optimal solutions found. As we can see that the proposed algorithm has been able to find high quality solutions.

In the next experiment we increased the number of concrete from 50 to 100.

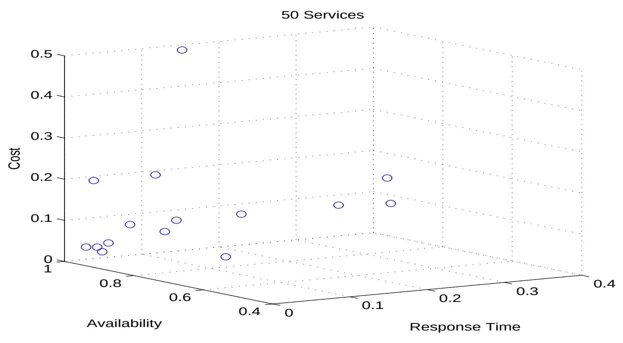


Figure 6.5: Results of composition with 50 services

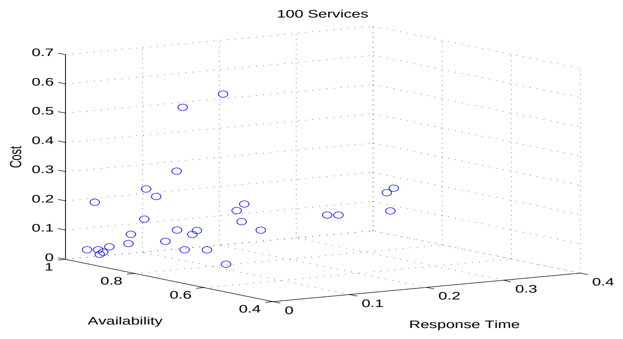


Figure 6.6: Results of composition with 100 services

In the next experiment we show that our solution converges very effectively towards the solution with varying number of services. We tested our approach with four abstract Web services and varied the number of concrete services to 100, 200, 300 and 400 Web services. As we can see from Fig. 6.7, our solution takes longer time for finding optimal solutions when the number of concrete Web services are increased. With 100 concrete Web services, our solution converges at around 600 episodes mark, while when we increase the number of concrete Web service to 200 it take it about 800 episodes to converge. Similarly it takes 1200 and 1400 episodes to find solutions for 300 and 400 concrete Web services respectively. This shows that our approach can handle large number of Web services and still performs efficiently with the increased number of Web services.

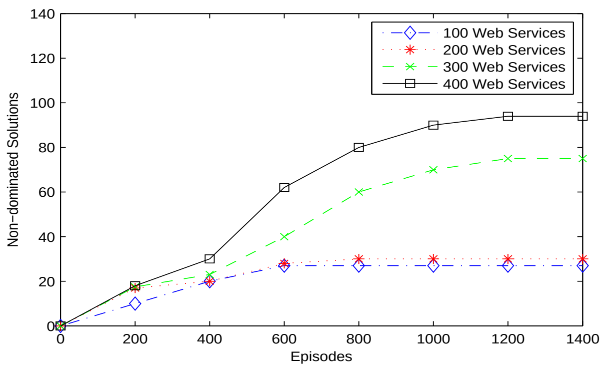


Figure 6.7: Multiple Constraint Algorithm

6.7.3 Experiment 3 : Individual QoS value comparisons

In this set of experiments we show the difference in performance of solution when we use just single constraint MDP versus multiple constraint MDP and adding the information from dependency modeling and collaborative filtering. Figure 6.8 shows the individual QoS value comparisons for our proposed approach. We can see the performance of multiple constraints , multiple constrains with invocation patterns and multiple constrains with invocation patterns with filtering on the different QoS values of Cost, Execution Time, Reliability, Availability and Throughput when tested on a pool of 50 to 400 service. We can see that Multi-policy with invocation patterns with filtering consistently yields better results for all the QoS values. We can see that introducing invocation patterns into multiple constrains algorithm yields significant improvement for QoS value of Cost, Reliability and Availability. These QoS values rely heavily on the orchestration pattern of service within a composition. Reliability benefits most from the introduction of collaborative filtering as compared to other methods. Availability was the least improved QoS value among our experiments as the services did not show any significant variation in their availability values. However, it is evident that adding the invocation information and predicting observed QoS values improves the c

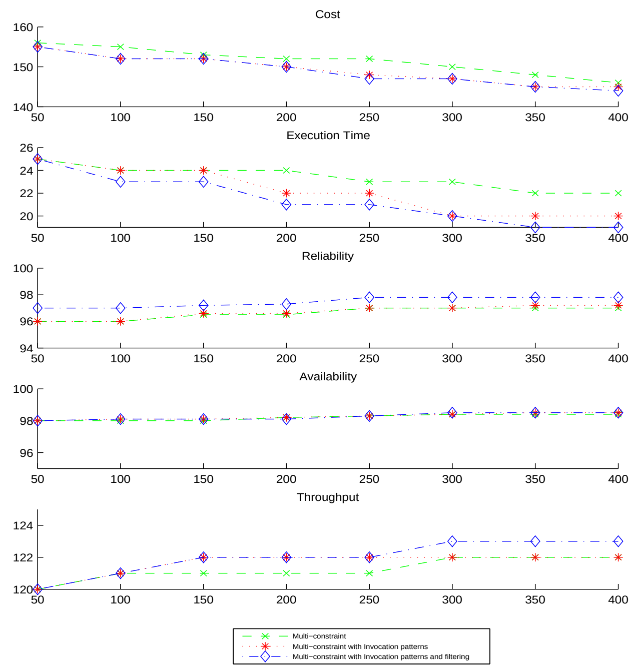


Figure 6.8: QoS value comparison

6.7.4 Experiment 4 : Individual QoS value comparisons for Invocation patterns

In this set of experiments we compare the impact of using our approaches for different QoS invocation patterns. We implemented three different SOA systems that use 15, 30 and 45 services respectively with different invocation patterns. The number of distinct execution paths for each of these systems is 16 , 51 and 77 respectively. We replace one service from the composite system at different invocation points to measure the impact on the total utility of the system. Figure 6.9 - Figure 6.13 show the utility gain for all three solutions with varying number of component services. We can see that for every invocation pattern MclpFi performs better when the number of component service increase. This gain is primarily related to better prediction since the more data points are available to calculate the similarity of services. Cost, Execution Time and Throughput are the components that benefit more from the proposed solution. Reliability and Availability also show slight gains in the utility value.

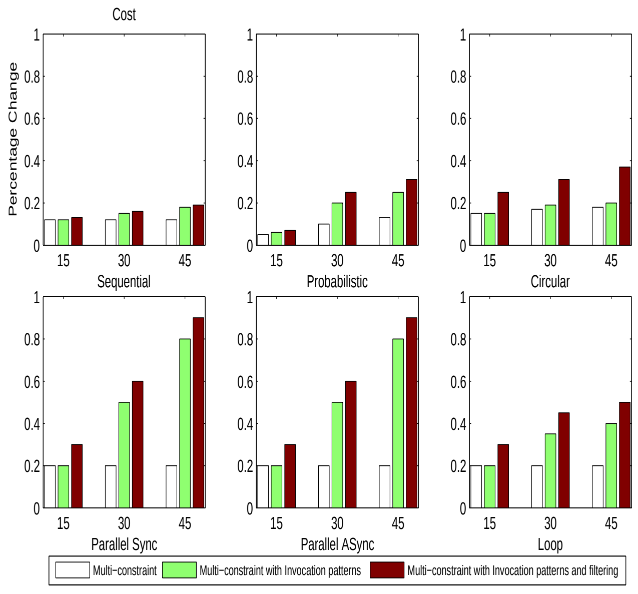


Figure 6.9: Utility comparison of Cost

6.7.5 Experiment 5 : Comparison with existing approaches

In this experiment we compare our solution(MclpFi) 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 Nitto et al. [2007] 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 consumer 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,

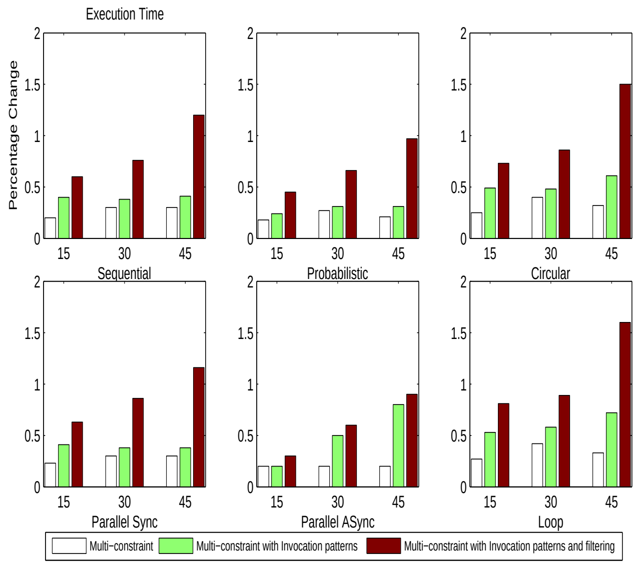


Figure 6.10: Utility comparison of Execution Time

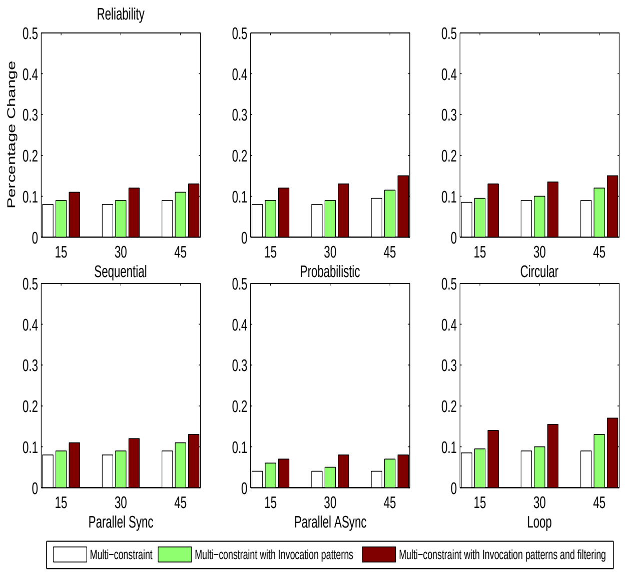


Figure 6.11: Utility comparison of Reliability

NBA Niu and Wang [2008] 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 Lecue [2009] also uses a GA based approach for the semantic composition of Web services. It uses the semantic equivalence in addition to the QoS

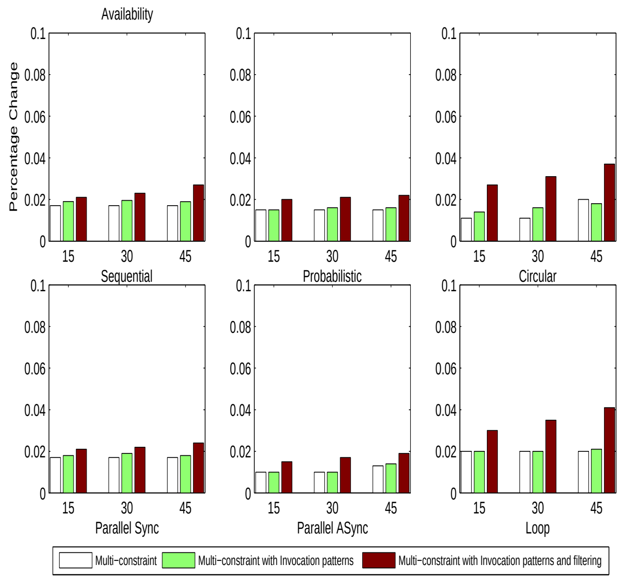


Figure 6.12: Utility comparison of Availability

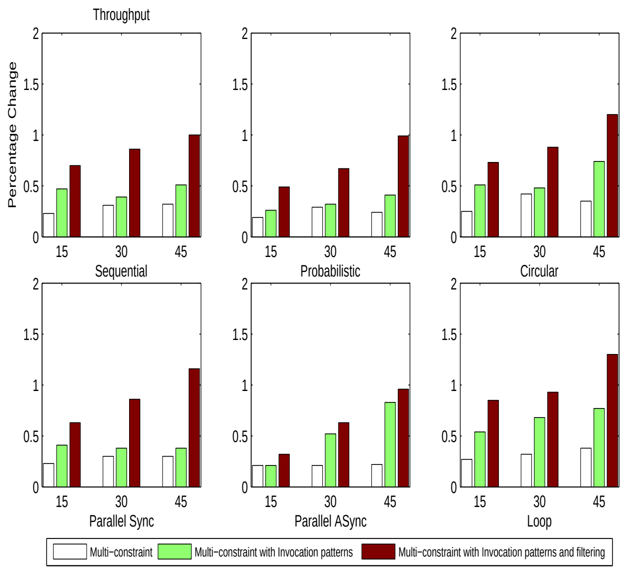


Figure 6.13: Utility comparison of Throughput

values to determine the best offering for the composition. BLGAN Sim et al. [2009] uses a Bayesian learning based approach with GA and incomplete information model to learn the reserve price of it opponent. GTFSN Figueroa et al. [2009] presents a game theoretical model of signaling games for Service Level Agreement negotiation.

The results are presented in Figure 6.14. We can see that our solution is the quickest in improving on the initial solution. This can be attributed to the use of invocation patterns, and the solution improves exponentially as the values stabilize (high jumps around time 27 and 49). We can also see that our solution finds a solution within the 97% utility range in about 66ms, while SWC (the second best) takes about 3 times more time. Other techniques fail to generate such a solution and NBA and SBA plateau around the 92% range while BLGAN and GTFSN only reach a solution of around 94% utility. The results suggest that our approach outperforms similar methods both in terms of finding the optimal solution and the amount of time it takes to find that solution

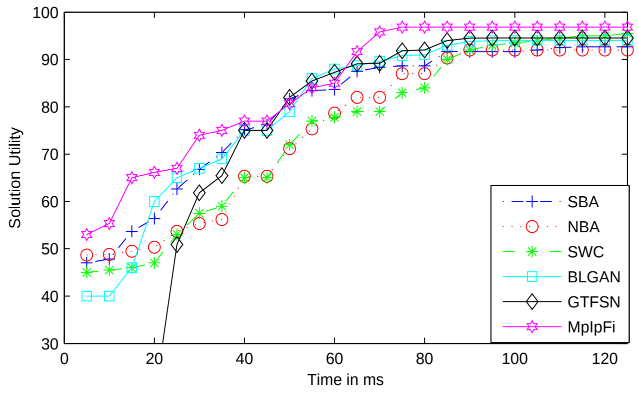


Figure 6.14: Performance comparison of proposed approach

CHAPTER 7: CONCLUSION AND FUTURE WORK

Automated negotiation among Web services is a challenging problem. So far the work has been focused on developing a comprehensive framework that could be used by both the service providers and service consumers to negotiate a better solution. In this regards we have looked into the GA based implementation for negotiating a solution. This solution is focused on using the information present in the negotiation context and learning from the environment in order to come up with a much desireable solution. This process takes advantage of the information shared by its participants for better convergence towards a mutually agreeable solution. Then we explored how social networks can provide added information that if utilized for this negotiation process can speed up the process while providing even better solution. We used semantic Web rules and ontologies to allow users to use different negotiation protocols to participate in a multi-party and multi-criteria automated negotiation process for Quality of Service components for a composite solution. This methodology allows more participants to get involved into the negotiation process and increase the search space in order to potentially find a better solution.

Currently I am exploring other pieces of information that could be used to formulate an even better solution. One of the very important aspect to analyze is the internal structure or orchestration of the individual services in a composition. This orchestration can be very useful when considering the dependency modeling among the Web services. We can utilize different invocation patterns to identify bottleneck and observed QoS parameters for the system and use them for our negotiation goals. Secondly the QoS values of Web services can be very uncertain. There could be discrepancies among the published and perceived QoS values of Web services. Different users can observe different QoS values for the same service i.e. response time. These observed values are often based on the environment and are out of the hands of the service providers. Hence, we need to figure out a mechanism to predict the observed QoS values for Web services in situations when invoking them or having direct observation is not possible. Thirdly reputation of a service plays a very prominent role in the selection phase of a composition. We want all the component service to be very reputable so that our composition could perform consistently. We need to look into how to extract the performance of an individual service from a rating that is given to the composition. Using the textual information provided with the reviews of the service along with the review score can provide very meaningful information in this regards.

REFERENCES

K. A, “openccs: Computing center software,” Aderborn Center for Parallel Computing, Technical report, 2007.

R. L. Ackoff, *Redesigning the future.* New York: Wiley, 1978.

R. Aggarwal, K. Verma, J. Miller, and W. Milnor, “Constraint driven web service composition in METEOR-S,” in *Services Computing, 2004.(SCC 2004). Proceedings. 2004 IEEE International Conference on.* IEEE, 2004, pp. 23–30.

L. Ai, M. Tang, and C. Fidge, “Partitioning composite web services for decentralized execution using a genetic algorithm,” *Future Generation Computer Systems*, vol. 27, no. 2, pp. 157–172, 2011.

K. Alexander and L. Heiko, “The wsla framework: Specifying and monitoring service level agreements for web services,” *J. Netw. Syst. Manage.*, vol. 11, no. 1, pp. 57–81, 2003.

G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web services: concepts, architectures and applications.* Springer, 2004.

M. Alrifai and T. Risse, “Combining global optimization with local selection for efficient qos-aware service composition,” in *Proceedings of the 18th international conference on World wide web.* ACM, 2009, pp. 881–890.

J. M. Andreoli and S. Castellani, “Towards a flexible middleware negotiation facility for distributed components,” *Database and Expert Systems Applications, International Workshop on*, vol. 0, p. 0732, 2001.

A. Andrieux, A. Dan, K. Keahy, H. Ludwig, and J. Rofrano, “From ws-agreement to sla negotiation,” 2004, http://www.mcs.anl.gov/ keahey/Meetings/GRAAP/WS-Agreement0Constraints.pdf.

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*, vol. 1, 2000, pp. 501 –508.

D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *Software Engineering, IEEE Transactions on*, vol. 33, no. 6, pp. 369–384, 2007.

R. Ashri, I. Rahwan, and M. Luck, "Architectures for negotiating agents," in *Proceedings of the 3rd Central and Eastern European conference on Multi-agent systems.* Springer-Verlag, 2003, pp. 136–146.

J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*, 1985, pp. 101–111.

——, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 1987, pp. 14–21.

E. Balas and M. W. Padberg, "Set partitioning: A survey," *SIAM Review*, vol. 18, no. 4, pp. 710–760, 1976.

C. Bartolini, C. Preist, and N. R. Jennings, "A software framework for automated negotiation," in *SELMAS*, ser. Lecture Notes in Computer Science, vol. 3390. Springer, 2004, pp. 213–235.

C. Beam and A. Segev, "Automated negotiations: A survey of the state of the art," *Wirtschaftsinformatik*, vol. 39, no. 3, pp. 263–268, 1997.

M. Benyoucef and M. H. Verrons, "Configurable enegotiation systems for large scale and transparent decision making," *Group Decision and Negotiation*, vol. 17, no. 3, pp. 211–224, 2008.

R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for QoS-aware web service composition," in *Web Services, 2006. ICWS'06. International Conference on.* IEEE, 2006, pp. 72–82.

P. Bonhard and M. A. Sasse, "'knowing me, knowing you' – using profiles and social networking to improve recommender systems," *BT Technology Journal*, vol. 24, no. 3, pp. 84–98, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1007/s10550-006-0080-3

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

D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, “Web services architecture,” 2004.

G. Bruns and M. Cortes, “A hierarchical approach to service negotiation,” *Web Services, IEEE International Conference on*, vol. 0, pp. 460–467, 2011.

T. X. Bui and M. F. Shakun, “Negotiation processes, evolutionary systems design, and negotiator,” *Group Decision and Negotiation*, vol. 5, no. 10, pp. 339–353, 1996.

T. Bui and A. Gachet, “Web services for negotiation and bargaining in electronic markets: Design requirements and implementation framework,” in *Proceedings of the 38th Hawaii International Conference on System Sciences*, 2005.

R. Burke, “Hybrid recommender systems: Survey and experiments,” *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, “A framework for qos-aware binding and re-binding of composite web services,” *Journal of Systems and Software*, vol. 81, no. 10, pp. 1754–1769, 2008.

C. Cappiello, M. Comuzzi, and P. Plebani, “On automated generation of web service level agreements,” *Advanced Information Systems Engineering*, vol. 4495, pp. 264–278, 2007.

H. E. Carter, A. Martin, D. Schofield, G. Duchesne, A. Haworth, C. Hornby, M. Sidhom, and M. Jackson, “A decision model to estimate the cost-effectiveness of intensity modulated radiation therapy (imrt) compared to three dimensional conformal radiation therapy (3dcrt) in patients receiving radiotherapy to the prostate bed,” *Radiotherapy and Oncology*, vol. 112, no. 2, pp. 187–193, 2014.

S. Castellani, J. M. Andreoli, M. Bratu, O. Boissier, I. Alloui, and K. Megzari, “E-alliance: a negotiation infrastructure for virtual alliances,” 2002.

A. Chavez and P. Maes, "Kasbah: An agent marketplace for buying and selling goods," in *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996, pp. 75–90.

R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, "Web services description language (wsdl) version 2.0 part 1: Core language," *W3C Recommendation*, vol. 26, 2007.

M. Claypool, P. Le, M. Wased, and D. Brown, "Implicit interest indicators," in *IN PROCEEDINGS OF IUI*, 2001, pp. 33–40.

C. A. C. Coello, B. G. Lamont, and D. A. V.Veldhuisen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, ser. Genetic and Evolutionary Computation Series. Springer, 2007. [Online]. Available: http://books.google.com/books?id=rXIuAMw3lGAC

M. Comuzzi and B. Pernici, "An architecture for flexible web service qos negotiation," in *EDOC '05: Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference*, 2005, pp. 70–82.

J. .Csirik and G. J. Woeginger, "Shelf algorithms for on-line strip packing." *Inf. Process. Lett.*, vol. 63, no. 4, pp. 171–175, 1997. [Online]. Available: http://dblp.uni-trier.de/db/journals/ipl/ipl63.html#CsirikW97

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*, vol. 6, no. 2, pp. 86 –93, 2002.

——, "Unraveling the web services web: an introduction to soap, wsdl, and uddi," *Internet Computing, IEEE*, vol. 6, no. 2, pp. 86–93, 2002.

K. Deb and J. Horn, "Introduction to the special issue: Multicriterion optimization," *Evol. Comput.*, vol. 8, no. 2, pp. 3–4, Jun. 2000. [Online]. Available: http://dx.doi.org/10.1162/106365600568149

N. S. K. Deb, “Muiltiobjective optimization using nondominated sorting in genetic algorithms,” *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, Sep. 1994. [Online]. Available: http://dx.doi.org/10.1162/evco.1994.2.3.221

G. Denaro, M. Pezzé, D. Tosi, and D. Schilling, “Towards self-adaptive service-oriented architectures,” in *TAV-WEB ’06: Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications.* New York, NY, USA: ACM, 2006, pp. 10–16.

D. A. D’Mello and V. S. Ananthanarayana, “A tree structure for web service compositions,” in *COMPUTE ’09: Proceedings of the 2nd Bangalore Annual Compute Conference.* New York, NY, USA: ACM, 2009, pp. 1–4.

P. E. Dunne, M. Wooldridge, and M. Laurence, “The complexity of contract negotiation,” *Artif. Intell.*, vol. 164, no. 1-2, pp. 23–46, May 2005. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2005.01.006

A. Elfatatry and P. J. Layzell, “A negotiation description language,” *Software, Practice and Experience*, pp. 323–343, 2005.

P. Faratin, C. Sierra, and R. Jennings, “Negotiation decision functions for autonomous agents,” *Robotics and Autonomous Systems*, vol. 24, no. 3-4, pp. 159–182, 1998. [Online]. Available: http://eprints.ecs.soton.ac.uk/2117/

P. Faratin, C. Sierra, and N. R. Jennings, “Using similarity criteria to make issue trade-offs in automated negotiations,” *Artificial Intelligence*, vol. 142, pp. 205–237, 2002.

C. Figueroa, N. Figueroa, A. Jofre, A. Sahai, Y. Chen, and S. Iyer, “A Game Theoretic Framework for SLA Negotiation,” 2009. [Online]. Available: http://www.hpl.hp.com/techreports/2008/HPL-2008-5.pdf

C. Firan, W. Nejdl, and R. Paiu, “The benefit of using tag-based profiles,” *Web Congress, Latin American*, vol. 0, pp. 32–41, 2007.

D. B. Fogel and L. C. Stayton, "On the effectiveness of crossover in simulated evolutionary optimization," *Biosystems*, vol. 32, no. 3, pp. 171 – 182, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/030326479490040X

C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, no. 1, pp. 26 –37, jan 1998.

——, in *Proceedings of the 5th International Conference on Genetic Algorithms.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 416–423.

——, "Evolutionary computation," vol. 3, no. 1, 1995, pp. 1–16.

F. for Intelligent Physical Agents, "Fipa iterated contract net interaction protocol specification," 2002, http://www.fipa.org/specs/fipa00030/SC00030H.html.

R. A. Gallege., A. Monticelli, and R. Romero, "Transmission system expansion planning by an extended genetic algorithm," *Generation, Transmission and Distribution, IEE Proceedings-*, vol. 145, no. 3, pp. 329 –335, May 1998.

M. Gen and R. Cheng, *Genetic algorithms and engineering design*, ser. Wiley series in engineering design and automation. Wiley, 1997. [Online]. Available: http://books.google.com/books?id=32tRAAAAMAAJ

D. Ghosh, R. Sharman, H. Raghav Rao, and S. Upadhyaya, "Self-healing systems - survey and synthesis," *Decis. Support Syst.*, vol. 42, no. 4, pp. 2164–2185, 2007.

H. Gimpel, H. Ludwig, A. Dan, and B. Kearney, "Panda: Specifying policies for automated negotiations of service contracts," 2003, pp. 287-302.

J. Golbeck, "Generating predictive movie recommendations from trust in social networks," in *Proceedings of the 4th international conference on Trust Management*, ser. iTrust'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 93–104. [Online]. Available: http://dx.doi.org/10.1007/11755593_8

——, "Weaving a web of trust," *Science*, vol. 321, no. 5896, pp. 1640–1641, 2008.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, Dec. 1992. [Online]. Available: http://doi.acm.org/10.1145/138859.138867

G. R. A. A. P. (GRAAP), "Wsagreement," 2007, http://www.ogf.org/documents/GFD.107.pdf.

S. Guinea, "Self-healing web service compositions," in *ICSE '05: Proceedings of the 27th international conference on Software engineering.* New York, NY, USA: ACM, 2005, pp. 655–655.

H. Halpin, V. Robu, and H. Shepherd, "The complex dynamics of collaborative tagging," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 211–220. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242602

K. Hashmi, A. Alhosban, Z. Malik, and B. Medjahed, "Webneg: A genetic algorithm based approach for service negotiation," in *Web Services (ICWS), 2011 IEEE International Conference on*, july 2011, pp. 105 –112.

T. Hofmann, "Collaborative filtering via gaussian probabilistic latent semantic analysis," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval.* ACM, 2003, pp. 259–266.

——, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89–115, 2004.

T. Hofmann and D. Hartmann, "Collaborative filtering with privacy via factor analysis," in *Proceedings of the 2005 ACM Symposium on Applied Computing*, 2005, pp. 791–795.

J. Horn, N. Nafpliotis, and D. E. Goldberg, "Multiobjective optimization using the niched pareto genetic algorithm," Tech. Rep., 1993.

——, "A niched pareto genetic algorithm for multiobjective optimization," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, jun 1994, pp. 82 –87 vol.1.

H.Tung and R. Lin, "Automated contract negotiation using a mediation service," in *Seventh IEEE International Conference on E-Commerce Technology*, 2005, pp. 374 – 377.

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.

H. W. Huo, J. Xu, and Z. bao, "Solving 01 knapsack problem using genetic algorithm," in *J. Xidian Univ*, vol. 26, no. 4, 1999, pp. 493–497.

H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 28, no. 3, pp. 392–403, 1998.

D. Ivanovic, P. Kaowichakorn, and M. Carro, "Towards QoS prediction based on composition structure analysis and probabilistic environment models," in *Principles of Engineering Service-Oriented Systems (PESOS), 2013 ICSE Workshop on*. IEEE, 2013, pp. 11–20.

A. Jaiswal, Y. Kim, and M. L. Gini, "Design and implementation of a secure multi-agent marketplace," *Electronic Commerce Research and Applications*, vol. 3, no. 4, pp. 355–368, 2004.

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*, vol. 10, no. 2, pp. 199–215, 2001. [Online]. Available: http://eprints.ecs.soton.ac.uk/4231/

R. Jin, J. Y. Chai, and L. Si, "An automatic weighting scheme for collaborative filtering," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 2004, pp. 337–344.

C. Jonker, V. Robu, and J. Treur, "An agent architecture for multi-attribute negotiation using incomplete preference information," *Autonomous Agents and MultiAgent Systems*, vol. 15, pp. 221–252, October 2007.

A. Jøsang and S. Pope, "Semantic constraints for trust transitivity," in *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling - Volume 43*, ser. APCCM '05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 59–68. [Online]. Available: http://dl.acm.org/citation.cfm?id=1082276.1082284

A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Proceedings of the 29th Australasian Computer Science Conference - Volume 48*, ser. ACSC '06. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2006, pp. 85–94. [Online]. Available: http://dl.acm.org/citation.cfm?id=1151699.1151710

K. Karta, "An investigation on personalized collaborative filtering for web service selection," *Honours Programme thesis, University of Western Australia, Brisbane*, 2005.

M. J. Katchabaw, H. L. Lutfiyya, A. D. Marshall, and M. A. Bauer, "Policy-driven fault management in distributed systems," in *ISSRE '96: Proceedings of the The Seventh International Symposium on Software Reliability Engineering.* Washington, DC, USA: IEEE Computer Society, 1996, p. 236.

H. Kautz, B. Selman, and M. Shah, "Referral web: combining social networks and collaborative filtering," *Commun. ACM*, vol. 40, no. 3, pp. 63–65, Mar. 1997. [Online]. Available: http://doi.acm.org/10.1145/245108.245123

D. Kelly and J. Teevan, “Implicit feedback for inferring user preference: a bibliography,” *SIGIR Forum*, vol. 37, no. 2, pp. 18–28, Sep. 2003. [Online]. Available: http://doi.acm.org/10.1145/959258.959260

A. D. Keromytis, “Characterizing self-healing software systems,” in *Proceedings of the 4th International Conference on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS*, 2007.

G. E. Kersten, “Nego group decision support system,” *Inf. Manage.*, vol. 8, pp. 237–246, May 1985.

G. E. Kersten and S. J. Noronha, “Wwwbased negotiation support: Design, implementation and use,” *Decision Support Systems*, vol. 25, no. 2, pp. 135–154, 1999.

E. Kim, “Oasis quality model for web services. version 2.0,” 2005.

J. Kim and A. Segev, “A web services-enabled marketplace architecture for negotiation process management,” *Decision Support System*, vol. 40, pp. 71–87, 2005.

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

I. Konstas, V. Stathopoulos, and J. J. Jose, “On social networks and collaborative recommendation,” in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR ’09. New York, NY, USA: ACM, 2009, pp. 195–202. [Online]. Available: http://doi.acm.org/10.1145/1571941.1571977

R. Kowalczyk, “Fuzzy e-negotiation agents,” *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 6, pp. 337–347, 2002, 10.1007/s00500-002-0187-5.

S. Kraus, “Mutli-agents systems and applications,” J. G. Carbonell and J. Siekmann, Eds. New York, NY, USA: Springer-Verlag New York, Inc., 2001, ch. Automated negotiation and decision making in multiagent environments, pp. 150–172. [Online]. Available: http://dl.acm.org/citation.cfm?id=567248.567255

A. Krzywicki, W. Wobcke, Y. Kim, X. Cai, M. Bain, A. Mahidadia, and P. Compton, "Collaborative filtering for people-to-people recommendation in online dating: Data analysis and user trial," *International Journal of Human-Computer Studies*, vol. 76, no. 0, pp. 50 – 66, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581914001669

M. M. Lab, "Teteatete," 2000, online: ecommerce.media.mit.edu.

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

F. Lecue, "Optimizing qos-aware semantic web service composition," in *8th International Semantic Web Conference (ISWC2009)*, October 2009. [Online]. Available: http://data.semanticweb.org/conference/iswc/2009/paper/research/305

K. Lerman, "Social networks and social information filtering on digg," *CoRR*, vol. abs/cs/0612046, 2006.

C. Li, J. Giampapa, and K. Sycara, "Bilateral negotiation decisions with uncertain dynamic outside options," *IEEE Transactions on System, Man, and Cybernetics Part C: Application and Reviews*, vol. 36, no. 1, 2006.

H. Li, "A new category of business negotiation primitives for bilateral negotiation agents and associated algorithm to find pareto optimal solutions," in *4th International Workshop on Advanced Issues of ECommerce and Web-based Information Systems(WECWIS)*, 2002.

X. Li, L. Guo, and Y. E. Zhao, "Tag-based social interest discovery," in *Proceedings of the 17th international conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 675–684. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367589

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*, vol. 4, 1996, pp. 2677 –2682.

C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, and F. Fotouhi, "Service-oriented architecture for view: A visual scientific workflow management system," in *SCC '08: Proceedings of the 2008 IEEE International Conference on Services Computing*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 335–342.

G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.

A. R. Lomuscio, M. Wooldridge, and N. R. Jennings, "A classification scheme for negotiation in electronic commerce," *Group Decision and Negotiation*, vol. 12, no. 1, pp. 31–56, 2004.

A. Ludwig, P. Braun, R. Kowalczyk, and B. Franczyk, "A framework for automated negotiation of service level agreements in services grids," in *Business Process Management Workshops, BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, France, September 5, 2005, Revised Selected Papers*, C. Bussler and A. Haller, Eds., vol. 3812, 2005, pp. 89–101.

H. Ludwig, A. Dan, and R. Kearney, "Cremona: an architecture and library for creation and monitoring of ws-agreements," in *2nd international conference on Service oriented computing*, 2004.

C. Lueg, "Social filtering and social reality," in *In Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering*. ERCIM Press, 1997, pp. 10–12.

X. Luo, N. R. Jennings, N. Shadbolt, H. Leung, and J. Lee, "A fuzzy constraint based model for bilateral multi-issue negotiations in semi-competitive environments," *Artificial Intelligence Journal*, vol. 148, no. 1-2, pp. 53–102, 2003.

H. .Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR

'09. New York, NY, USA: ACM, 2009, pp. 203–210. [Online]. Available: http://doi.acm.org/10.1145/1571941.1571978

H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 39–46.

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*. Springer-Verlag, 2007, pp. 38–49.

——, "Reputation bootstrapping for trust establishment among web services," *Internet Computing, IEEE*, vol. 13, no. 1, pp. 40 –47, jan. 2009.

——, "Rateweb: Reputation assessment for trust establishment among web services," *VLDB J.*, vol. 18, no. 4, pp. 885–911, 2009.

——, "Rater credibility assessment in web services interactions," *World Wide Web*, vol. 12, no. 1, pp. 3–25, 2009.

——, "Rateweb: Reputation assessment for trust establishment among web services," *The VLDB Journal*, vol. 18, no. 4, pp. 885–911, 2009.

——, "Rateweb: Reputation assessment for trust establishment among web services," *The VLDB Journal*, vol. 18, no. 4, pp. 885–911, 2009.

D. Margaris, C. Vassilakis, and P. Georgiadis, "An integrated framework for adapting ws-bpel scenario execution using qos and collaborative filtering techniques," *Science of Computer Programming*, vol. 98, Part 4, no. 0, pp. 707 – 734, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642314004778

P. .Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of the 2007 ACM conference on Recommender systems*, ser. RecSys '07. New York, NY, USA: ACM, 2007, pp. 17–24. [Online]. Available: http://doi.acm.org/10.1145/1297231.1297235

N. Mastronarde, K. Kanoun, D. Atienza, P. Frossard, and M. van der Schaar, “Markov decision process based energy-efficient on-line scheduling for slice-parallel video decoders on multicore systems,” *Multimedia, IEEE Transactions on*, vol. 15, no. 2, pp. 268–278, 2013.

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*, ser. ICMAS ’98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 182–. [Online]. Available: http://dl.acm.org/citation.cfm?id=551984.852238

S. Matwin, T. Szapiro, and K. Haigh, “Genetic algorithms approach to a negotiation support system,” *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 21, no. 1, pp. 102 –114, jan/feb 1991.

M. R. McLaughlin and J. L. Herlocker, “A collaborative filtering algorithm and evaluation metric that accurately model the user experience,” in *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004*, 2004, pp. 329–336. [Online]. Available: http://doi.acm.org/10.1145/1008992.1009050

B. Medjahed, A. Bouguettaya, and A. Elmagarmid, “Composing Web Services on the Semantic Web,” *The VLDB Journal*, vol. 12, no. 4, November 2003.

D. A. Menasce, “Composing web services: A qos view,” *IEEE Internet Computing*, vol. 8, pp. 88–90, 2004.

Z. Michalewicz, “Genetic algorithms, numerical optimization, and constraints.” Morgan Kaufmann, 1995, pp. 151–158.

Z. Michalewicz and N. Attia, “Evolutionary optimization of constrained problems,” in *Proceedings of the Third Annual Conf. Evolutionary Programming.* World Scientific River Edge, 1994, pp. 98–108.

D. Mobach, B. Overeinder, and F. Brazier, "A ws-agreement based resource negotiation framework for mobile agents," in *Scalable Computing: Practice and Experience*, 2006, pp. 23–26.

R. J. Mooney and L. Roy, "Content based book recommending using learning for text categorization," 2000, 606001.

F. P. Mora and C. Y. Wang, "Computer-supported collaborative negotiation methodology," in *J. Comput. Civil Eng*, 1998, pp. 64–81.

M. Morita and Y. Shinoda, "Information filtering based on user behavior analysis and best match text retrieval," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '94. New York, NY, USA: Springer-Verlag New York, Inc., 1994, pp. 272–281. [Online]. Available: http://dl.acm.org/citation.cfm?id=188490.188583

T. Murata, H. Ishibuchi, and H. Tanaka, "Multi-objective genetic algorithm and its applications to flowshop scheduling," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 957–968, Sep. 1996. [Online]. Available: http://dx.doi.org/10.1016/0360-8352(96)00045-9

T. D. Nguyen and N. R. Jennings, "Managing commitments in multiple concurrent negotiations," *ELECTRONIC COMMERCE RESEARCH AND APPLICATIONS*, vol. 4, no. 4, pp. 362–376, 2005.

——, "Bayesian learning in negotiation," *Int. J. Hum.-Comput. Stud.*, pp. 125–141, 1998.

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

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.

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*, june 2008, pp. 3834 –3838.

J. O'Donovan and B. Smyth, "Trust in recommender systems," in *Proceedings of the 10th international conference on Intelligent user interfaces*, ser. IUI '05. New York, NY, USA: ACM, 2005, pp. 167–174. [Online]. Available: http://doi.acm.org/10.1145/1040830.1040870

A. Osyczka and S. Kundu, "A modified distance method for multicriteria optimization, using genetic algorithms," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 871–882, Sep. 1996. [Online]. Available: http://dx.doi.org/10.1016/0360-8352(96)00038-1

M. P. Papazoglou and W. Heuvel, "Service oriented architectures: approaches, technologies and research issues," *The VLDB Journal*, vol. 16, pp. 389–415, July 2007. [Online]. Available: http://dx.doi.org/10.1007/s00778-007-0044-3

M. Papazoglou and P. Hall, *Web Services: Principles and Technology*, 2008, vol. ISBN: 978-0-321-15555-9.

V. Patankar and R. Hewett, "Automated negotiation in web service procurement," in *Proceedings of the Third International Conference on Internet and Web Applications and Services*, 2008.

J. Patrick, "A markov decision model for determining optimal outpatient scheduling," *Health care management science*, vol. 15, no. 2, pp. 91–102, 2012.

S. Paurobally, V. Tamma, and M. Wooldrdige, "A framework for web service negotiation," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 2, no. 4, 2007.

S. Paurobally, C. van Aart, V. Tamma, M. Wooldridge, and P. van Hapert, "Web services negotiation in an insurance grid," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 2007.

S. Perugini, M. A. Gonçalves, and E. A. Fox, "Recommender systems research: A connection-centric survey," *J. Intell. Inf. Syst.*, vol. 23, no. 2, pp. 107–143, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:JIIS.0000039532.05533.99

M. Pesce, D. Munaretto, and M. Zorzi, "A markov decision model for source video rate allocation and scheduling policies in mobile networks," in *Ad Hoc Networking Workshop (MED-HOC-NET), 2014 13th Annual Mediterranean.* IEEE, 2014, pp. 119–125.

A. Pichot, OliverWaldrich, W. Ziegler, and P. Wieder, "Towards dynamic service level agreement negotiation:an approach based on ws-agreement," in *Web Information Systems and Technologies 4th International Conference, WEBIST 2008, Funchal, Madeira, Portugal*, 2008.

G. Pitsilis and S. Knapskog, "Social trust as a solution to address sparsity-inherent problems of recommender systems," *arXiv preprint arXiv:1208.1004*, 2012.

C. Preist, C. Bartolini, and A. Byde, "Agentbased service composition through simultaneous negotiation in forward and reverse auctions," in *Proceedings of the 4th ACM conference on Electronic commerce.* ACM, 2003, pp. 55 – 63.

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*, vol. 16, no. 8, pp. 609–15, 2000.

M. Resinas, P. Fernndez, and R. Corchuelo, "A bargaining-specific architecture for supporting automated service agreement negotiation systems," *Science of Computer Programming*, vol. 77, no. 1, pp. 4 – 28, 2012, ¡ce:title¿System and Software Solution Oriented Architectures¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642310001851

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work.* ACM, 1994, pp. 175–186.

S. Rinderle and M. Benyoucef, "Towards the automation of e-negotiation processes based on web services a modeling approach." in *WISE 05*, 2005, pp. 443–453.

J. A. Rodrguez-Aguilar, A. Giovanucci, A. Reyes-Moro, F. X. Noria, and J. Cerquides, "Agent-based decision support for actual-world procurement scenarios," in *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology*, 2003.

R. Ros and C. Sierra, "A negotiation meta strategy combining trade-off and concession moves," *Journal of Autonomous Agent and Multiagent Systems*, vol. 12, pp. 163–181, 2006.

M. H. Rothkopf, A. Peke, and R. M. Harstad, "Computationally manageable combinational auctions," *MANAGEMENT SCIENCE*, vol. 44, no. 8, pp. 1131–1147, 1998.

B. Rubenstein-Montano and R. A. Malaga, "A weighted sum genetic algorithm to support multiple-party multiple-objective negotiations," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 4, pp. 366–377, 2002.

S. S. S. Matwin and Z. Koperczak, "Negoplan: An expert system shell for negotiation support," *IEEE Expert*, vol. 4, no. 4, pp. 50–62, 1996.

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*, ser. EC '00. New York, NY, USA: ACM, 2000, pp. 30–37. [Online]. Available: http://doi.acm.org/10.1145/352871.352875

T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artif. Intell.*, vol. 135, no. 1-2, pp. 1–54, Feb. 2002. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(01)00159-X

T. Sandholm and S. Suri, "Side constraints and non-price attributes in markets," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

T. Sandholm, S. Suri, A. Gilpin, and D. Levine, "Winner determination in combinatorial auction generalizations," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, 2002.

T. W. Sandholm and V. R. Lesser, “Leveled commitment contracts and strategic breach,” *Games and Economic Behavior*, vol. 35, pp. 212–270, 2001.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th international conference on World Wide Web*, ser. WWW ’01. New York, NY, USA: ACM, 2001, pp. 285–295. [Online]. Available: http://doi.acm.org/10.1145/371920.372071

J. D. Schaffer, “Multiple objective optimization with vector evaluated genetic algorithms,” in *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 93–100. [Online]. Available: http://dl.acm.org/citation.cfm?id=645511.657079

A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, “Methods and metrics for cold-start recommendations,” in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR ’02. New York, NY, USA: ACM, 2002, pp. 253–260. [Online]. Available: http://doi.acm.org/10.1145/564376.564421

Y. .Shang and B. W. Wah, “A discrete lagrangian-based global-searchmethod for solving satisfiability problems,” *J. of Global Optimization*, vol. 12, no. 1, pp. 61–99, Jan. 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1008287028851

U. Shardanand and P. Maes, “Social information filtering: algorithms for automating word of mouth,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’95. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217. [Online]. Available: http://dx.doi.org/10.1145/223904.223931

Y. Shimizu, “Multi-objective optimization of mixed-integer programming problems through a hybrid genetic algorithm with repair operation,” pp. 395–404, 1999.

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, ser. SC '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1188455.1188563

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*, sept. 2007, pp. 1007 –1014.

——, "Blgan: Bayesian learning and genetic algorithm for supporting negotiation with incomplete information," *Trans. Sys. Man Cyber. Part B*, vol. 39, no. 1, pp. 198–211, Feb. 2009. [Online]. Available: http://dx.doi.org/10.1109/TSMCB.2008.2004501

R. R. Sinha and K. Swearingen, "Comparing Recommendations Made by Online Systems and Friends," in *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

R. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions Computers*, vol. C-29, no. 12, 1980.

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*, vol. 79, no. 2, pp. 131 – 143, 2001.

O. Standard, "Wsbpel," 2005, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.

W. Standard, "Soap version 1.2," 2007, http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.

M. Strobel, "Design of roles and protocols for electronic negotiations," vol. 1, pp. 335–353, July 2001.

V. Tamma, M. Wooldridge, and I. Dickinson, "An ontology for automated negotiation," in *First Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS02)*, 2002.

M. Tennenholtz, “Tractable combinatorial auctions and bmatching,” *Artificial Intelligence*, vol. 140, pp. 231–243, 2002.

D. S. Todd and P. Sen, “A multiple criteria genetic algorithm for containership loading,” in *Proc. Seventh Int. Conf. Genetic Algorithms*, 1997, pp. 674–681.

V. Tosic, P. Bernard, P. Kruti, E. Babak, and M. Wei, “Management applications of the web service offerings language (wsol),” *Inf. Syst.*, vol. 30, no. 7, pp. 564–586, 2005.

M. Tu, C. Seebode, F. Griffel, and W. Lamersdorf, “Dynamics: An actor-based framework for negotiating mobile agents,” vol. 1, pp. 101–117, February 2001.

J. Vanhatalo, H. Völzer, and J. Koehler, “The refined process structure tree,” *Data & Knowledge Engineering*, vol. 68, no. 9, pp. 793–818, 2009.

P. K. Vatturi, W. Geyer, C. Dugan, M. Muller, and B. Brownholtz, “Tag-based filtering for personalized bookmark recommendations,” in *Proceedings of the 17th ACM conference on Information and knowledge management*, ser. CIKM ’08. New York, NY, USA: ACM, 2008, pp. 1395–1396. [Online]. Available: http://doi.acm.org/10.1145/1458082.1458297

D. A. V. Veldhuizen and G. B. Lamont, “Multiobjective evolutionary algorithms: Analyzing the state-of-the-art,” *Evol. Comput.*, vol. 8, pp. 125 – 147, June 2000.

V. Veldhuizen and D. Allen, “Multiobjective evolutionary algorithms: classifications, analyses, and new innovations,” Ph.D. dissertation, Wright Patterson AFB, OH, USA, 1999, aAI9928483.

R. Viennet, C. C. Fonteix, and I. Marc, “Multicriteria optimization using a genetic algorithm for determining a pareto set,” *International Journal of Systems Science*, vol. 27, no. 2, pp. 255–260, 1996. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/00207729608929211

W. W. W. C. (W3C), “Wspolicy,” 2006, http://www.w3.org/Submission/WS-Policy/.

O. Waldrich, P. Wieder, and W. Ziegler, *A Meta-scheduling Service for Co-allocating Arbitrary Types of Resources*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. Volume 3911/2006, ch. Parallel Processing and Applied Mathematics.

F. E. Walter, S. Battiston, and F. Schweitzer, "A Model of a Trust-based Recommendation System on a Social Network," *eprint arXiv:nlin/0611054*, Nov. 2006.

H. Wang, X. Zhou, X. Zhou, W. Liu, W. Li, and A. Bouguettaya, "Adaptive service composition based on reinforcement learning," in *Service-Oriented Computing*. Springer, 2010, pp. 92–107.

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*. Morgan Kaufmann Publishers Inc., 1989, pp. 116–121.

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

D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 67 –82, Apr. 1997.

P. R. Wurman, M. P. Wellman, and W. E. Walsh, "The michigan internet auctionbot: A configurable auction . . ." in *IN SECOND INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS*, 1998, pp. 301–308.

G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005, pp. 114–121.

Y. Yao, F. Yang, and S. Su, "Evaluating proposals in web services negotiation," in *Computer and Information Sciences ISCIS 2006*. Springer Berlin / Heidelberg, 2006, pp. 613–621.

G. Yee and L. Korba, "Bilateral e-services negotiation under uncertainty," in *Proceedings of the 2003 Symposium on Applications and the Internet*, 2003.

B. Yu and M. P. Singh, "Detecting deception in reputation management," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS '03. New York, NY, USA: ACM, 2003, pp. 73–80. [Online]. Available: http://doi.acm.org/10.1145/860575.860588

Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and managing web services: issues, solutions, and directions," *The VLDB Journal*, vol. 17, no. 3, pp. 537–572, 2008.

T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Trans. Web*, vol. 1, no. 1, p. 6, 2007.

——, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, p. 6, 2007.

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*, 2000, pp. 656–660.

A. Zarras, P. Vassiliadis, and V. Issarny, "Model-driven dependability analysis of webservices," in *Web Services, International Symposium on Distributed Objects and Applications*, 2004, pp. 69–79.

M. A. Zemni, S. Benbernou, and M. Carro, "A soft constraint-based approach to qos-aware service selection," in *Service-Oriented Computing*. Springer, 2010, pp. 596–602.

L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, May 2004.

L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, “Qos-aware middleware for web services composition,” *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, 2004.

Y. Zhang, Z. Zheng, and M. R. Lyu, “Wsexpress: A qos-aware search engine for web services,” in *Proc. IEEE Int’l Conf. Web Services (ICWS’10)*, 2010, pp. 83–90.

E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach,” pp. 257–271, 1999.

E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results,” *Evol. Comput.*, vol. 8, pp. 173–195, June 2000.

ABSTRACT

AUTOMATED NEGOTIATION AMONG WEB SERVICES

by

Khayyam Hashmi

May 2016

Advisor: Dr. Zaki Malik

Major: Computer Science

Degree: Doctor of Philosophy

Software as a service is well accepted software deployment and distribution model that is grown exponentially in the last few years. One of the biggest benefits of SaaS is the automated composition of these services in a composite system. It allows users to automatically find and bind these services, as to maximize the productivity of their composed systems, meeting both functional and non-functional requirements. In this dissertation we present an automated negotiation framework for Service systems that can be used by both the parties for conducting negotiations. We proposed multiple algorithms for finding acceptable solutions in multi-party and multi-objective scenarios. We incorporates the dependencies of different quality attributes of independently developed component services for the system composition, considering the different invocation patterns for optimum values of these QoS parameters. We evaluate our approached on multiple data sets and lastly outline the future direction of the proposed techniques.

AUTOBIOGRAPHICAL STATEMENT

Khayyam Hashmi is a Ph.D. candidate in the Department of Computer Science at Wayne State University, received his Bachelors degree in Computer Science from National University and received his Masters degree in Computer Science from National University. He is a member of the Services COmputing REsearch (SCORE) Laboratory. Since starting her doctoral studies at Wayne State University he has served under various capacities (e.g. Graduate Research / Teaching Assistant, Student Assistant, and Part Time Faculty) in the Department of Computer Science. He is currently working as a Lecturer in the Computer Science Department at Wayne State University. His research interests include service computing, distributed systems, semantic web and software process engineering. He has published several research papers on these topics, and is an active member of ACM and IEEE.