Chapter 18
Automated Negotiation Among Web services

Khayyam Hashmi, Amal Alhosban, Zaki Malik, Brahim Medjahed and Salima Benbernou

Abstract Automated negotiation among Web services not only provides an effective way for the services to bargain for their optimal customizations, but also allows the discovery of overlooked potential solutions. A number of negotiation supporting techniques have been used to find solutions that are acceptable to all parties in the negotiation. However, employing these solutions for automated negotiations among Web services has its own challenges. In this chapter, we present the design of a Negotiation Web service that would be used by both the consumers and providers of Web services for conducting negotiations. This negotiation service uses a genetic algorithm (GA) based approach for finding acceptable solutions in multi-party and multi-objective negotiations. In addition to the traditional genetic operators of crossover and mutation, the search is enhanced using a new operator called the *Norm*. *Norm* operator represents the cumulative knowledge of all the parties involved in the negotiation process. GA performance with the new *Norm* operator is compared to the traditional GA, hill-climber and random search techniques. Experimental results indicate the practicality of the approach in facilitating the negotiations involved in a Web service composition process. Specifically, the proposed GA with *Norm* operator performs better than other approaches.

K. Hashmi (✉) · A. Alhosban · Z. Malik
Wayne State University, Detroit, Michigan, USA
e-mail: eh2304@wayne.edu

Z. Malik
e-mail: zaki@wayne.edu

A. Alhosban
e-mail: ea1179@wayne.edu

B. Medjahed
The University of Michigan - Dearborn, Dearborn, Michigan, USA
e-mail: brahim@umd.umich.edu

S. Benbernou
Universite Paris Descartes, Paris, France
e-mail: salima.benbernou@parisdescartes.fr

A. Bouguettaya et al. (eds.), *Web Services Foundations*,
DOI: 10.1007/978-1-4614-7518-7_18,

18.1 Introduction

A Web service is defined as an autonomous and self-contained unit of application that is accessible over a network [83]. In recent years, the number of available Web services has increased, and it is believed that in the near future, we may find multiple services offering the same functionalities [57]. Moreover, with maturing standards (e.g., BPEL [79]) it is now possible to combine several services to formulate a composite solution (selecting the most suitable service for the composite solution from among a pool of competing services). However, this selection process is not straightforward as many inter-related variables of the different services may affect the performance of the service composition. To help facilitate this process we can use automated negotiation to provide an effective way for clients to bargain for an optimal customization of their required variables and to discover any overlooked potential solutions. In this chapter, the research problem of automated negotiation of Quality of Service (QoS) components among Web services is analyzed. The chapter is divided into four sections. The first section serves as an introduction to the service oriented paradigm and the concept of Web services, their underlying QoS specifications, and the process of negotiation. Section two focuses on the communication protocols for automated negotiation, while the third section discusses the different techniques/agents used in the negotiation process. In Sect. 18.4, we discuss the overall service negotiation requirements, show how existing solutions perform in the light of these requirements, and define an approach for solving the automated negotiation problem.

18.1.1 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* [57]. 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 [1]. Services in SOAs *share* schema and contracts. The message passing structures are specified by the schema, and message-exchange behaviors are specified by contracts (both implicit or explicit) for each SOA transaction.

Two major entities are involved in any SOA transaction: service customers and service providers. Figure 18.1 represents a typical Web service interaction model. Service providers offer their services by publishing their information (WSDL) in public registries (UDDI) [46, 54]. Customers then query these registries to find the

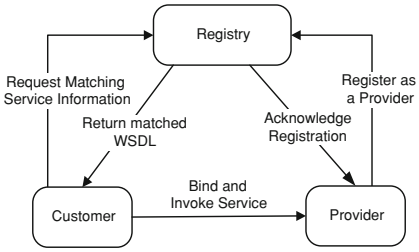

Fig. 18.1 Service-oriented interaction model

required services and then bind to the most suitable service, where input parameters are sent to the service provider and output is returned to the customer [2]. Web service registries thus serve as place holders and provide minimal functional information about a service. Service providers may use $tModels$ [21] to provide any additional information. Since $tModels$ are static place holders for information (service provider may provide a single value or a range for QoS attribute(s)), hence they have limited usability when it comes to negotiating non-functional requirements of the customers (e.g., availability, reliability etc.). A customer looking for a Web service could benefit from a service which if given both functional and non-functional requirements, could provide the most effective solution by simultaneously negotiating with multiple providers.

18.1.2 QoS Specification

The Quality of Service (QoS) is defined as a set of non-functional service attributes that indicate the service's ability to satisfy a stated or implied requirement in an end-to-end fashion [41]. This set of quality attributes not only characterize the service but also any entity used in the path between the service and its client. In a Service Oriented Architecture (SOA), service providers may characterize their services to define both the offered functionalities and the offered quality. Similarly the users may not only express their requirements by listing the desired functionalities, but also define a minimum level of quality that the service must ensure. The main issue here is the subjectiveness of 'quality': the quality of a service from the provider's perspective may be different than the quality experienced by the user. At the same time, the same quality level might be sufficient for a given user and not enough for another one. Hence, QoS parameters consist of both quantitative (availability 99.9 %) and qualitative (privacy, security) parts. Most of the quantitative attributes are not directly proportional in their cost/benefit curve e.g., 99.999 % uptime versus 99.0 % uptime. Hence this non-linear curve naturally generates a disparity among the provided values for these QoS attributes and opens them to negotiation.

18.1.3 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. A negotiation process enhances acquisition opportunities and enables flexible communication that can lead to a better solution [10, 92].

However, negotiation are usually uncertain (due to incomplete information of both parties) and knowledge intensive. Performing negotiations manually is thus ad-hoc and time-consuming. Automated component negotiations (e.g., on the web) are thus valuable not only for the customers 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.

An automated negotiation mechanism requires at least three components; a high-level protocol, objectives and strategies [47]. The high-level protocol controls the negotiation process depending on types of negotiations (e.g., auction). The objectives of all parties are based on a set of criteria, representing various parameters along with their respective domain values (e.g., price range). Negotiation strategies include mechanisms (rules and knowledge base) that the agent employs to generate and evaluate offers.

Figure 18.2 depicts the state diagram of an automated negotiation process. Typically, negotiation starts when the customer requests for proposals for a component service. After receiving the initial offers from service providers, it would then select some providers to engage in bi-lateral negotiations. This would start a round of offers and counter offers among the customer and selected service providers. Once both the provider and the customer agree on a certain attributes (e.g., price) for the services

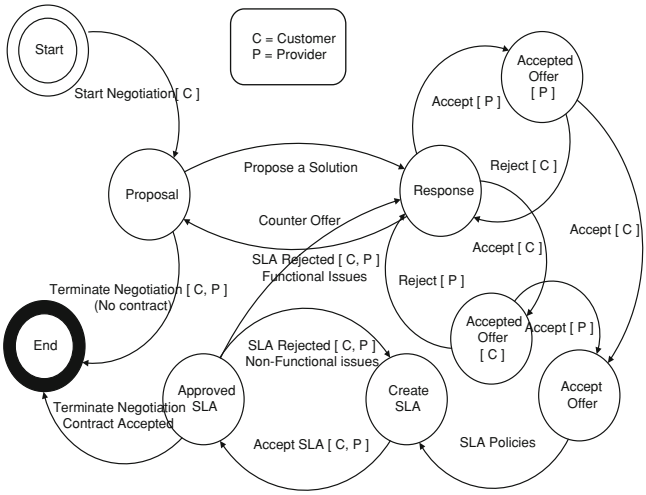

Fig. 18.2 Negotiation state chart

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

18.2 Communication Protocols for Negotiation

A communication protocol defines the syntax, semantics, rules and synchronization of messages exchanged between the partied involved. There are many communication protocols that have been defined to conduct negotiations. This section discusses some of the widely used negotiation protocols applied in the service's domain.

18.2.1 WS-Agreement

WS-Agreement [28] is a protocol for establishing agreements between two parties, such as between a service provider and customer. It uses XML for specifying the nature of the agreement, and agreement templates to facilitate discovery of compatible agreement parties. The specification consists of three parts which may be used in a composable manner: 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. 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. Although WS-Agreement does not have any negotiation specific structure but there had been discussions for using it in negotiating agreements among parties [5, 87]. An implementation of WS-Agreement to negotiate SLA's for resource orchestration in grids have been presented in [66]. 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 WS-Agreement 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.

Cremona [48] is an agreement management architecture that facilitates (agreement-based) service binding for a variety of services. It uses WS-Agreement as the communication infrastructure. The Cremona architecture separates multiple layers of agreement management, orthogonal to the agreement management functions: the functions associated with an agreement protocol role, initiator or provider, is the Agreement Protocol Role Management (APRM). It comprises, on the agreement provider side, the agreement factory, the agreement instance implementations, the Web services container in which factory and instances are located and interfaces to an agreement template repository, decision-making functionality for *createAgreement* requests and the current state of terms. On the agreement customer side, it comprises proxy functions to interact with an agreement factory and created agreements, template processing functions to create agreement instance document from templates, and interfaces to components initiating agreement establishment, to functions deciding on how to fill an agreement template, and to guarantee monitors. The Agreement Service Role Management (ASRM) is the collection of functions that deals with a party's role in the service relationship, provider or customer, and connects it to the service system. On the service provider's side, this includes the mapping of agreements to provisioning specifications and other input to the service-implementing system—the agreement implementation plan [44].

OpenCCS [35], AgentScape [58] and VIOLA MetaScheduling Service MSS [87] also 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.

18.2.2 Contract Net

Contract Net [78] is a generic negotiation protocol. It is viewed as a task having four components (1) it is a local process that does not involve centralized control, (2) there is two-way exchange of information, (3) each party to the negotiation evaluates the information from its own perspective, and (4) the final agreement is achieved by mutual selection.

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.

A variation of Contract Net protocol for Semantic Web service composition is discussed in [43]. The issue of aligning data flow in semantic web service composition is 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.

Another variation of Iterative model of Contract Net Protocol (CNP) for negotiation is discussed in [65], where 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.

Multiple strategies are implemented using the Iterative model of Contract Net Protocol in [64]. The most basic of them i.e., truth telling strategy, relies on the fact that 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.

18.2.3 WS-Policy

WS-Policy [86] 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 WS-Policy 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.

A framework based on WS-Policy for negotiation of Quality of Service attributes between Web services is proposed in [20]. 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 customer, in the WSDL. This QoS registry could be stored along with WSDL

using WSOL, which is a WSDL-compatible language for specifying different service offerings for the same service identified by the different values or constraints on the service QoS attributes [82]. It can include different domain schemas on which the QoS could be defined.

18.2.4 WS-Negotiation

WS-Negotiation [30] is an independent declarative XML language for Web service's providers and customers. In general, WS-Negotiation contains three parts: Negotiation Message, Negotiation Protocol and Decision Making. The Negotiation Message part describes the format of the messages exchanged. Some suggested message types are: Offer, Counter-Offer, Rejected, Accepted, etc. This part of WS-Negotiation tackles the "Initial Contact" and "Offer and Counter-Offer" tasks. Negotiation Protocol describes the mechanism and the rules that the negotiation parties should follow to exchange messages. Messages contain offers and counter-offers and can be exchanged between customer and provider as well as a third-party negotiation service (Negotiation Support System-NSS). Negotiation primitives are also defined in order to coordinate and execute the tasks and events. A negotiation primitive sets the pre and post conditions that should hold as well as rules and constraints that should be applied during the negotiation. Example of negotiation primitives are the "Propose" primitive for proposing an offer/counter-offer to the other party, the "Modify" primitive to modify the sent offer/counter-offer before receiving the other party's reply etc. The Negotiation Decision-making component takes the decisions. It is private and is based on the negotiation strategy each party has chosen (e.g., cost-benefit strategy) and the agreement template. This part of WS-Negotiation tackles the "Evaluation". Negotiation issues vary from one business domain to another but there are some issues that are common or fixed in a domain. Hence, there are several Service Level Agreement (SLA) template models, with domain specific vocabularies, for supporting different types of business negotiations.

WS-AgreementNegotiation [89] describes the re-negotiation of agreements between two parties. It specifies a set of messages and resources that can be used to model several re-negotiation scenarios. WS-AgreementNegotiation sits on top of WS-Agreement, which makes it possible to switch between different negotiation protocol. However, this requires WS-AgreementNegotiation to express negotiation offers in terms of WS-Agreement constructs. This adds a dependency to WS-Agreement. Moreover, it only allows the re-negotiation of existing agreements among two parties and could be initiated by either the provider or the customer.

18.2.5 Xplore

Xplore [4] provides a lightweight co-ordination platform focused at multi-party, multi-attribute negotiation. It acts as a "middleware" which aims at addressing general, domain independent requirements on the interaction infrastructure to support negotiation. It is based on the negotiation mechanism which is an extension of the Contract Net protocol [78] with transactional facilities, enabling the coordinated execution of a collection of concurrent, interdependent Contract Nets. It exploits the coordination mechanism provided by CLFMekano [3], a coordination middleware platform designed to integrate negotiation and transaction aspects in distributed systems. CLF contains primitives enabling negotiation and transaction at the lowest level. The primitives are expressed as a set of eight "interaction verbs" a la KQML, similar to speech acts [38]. Xplore extends the unidirectional "announce/collect/decide" paradigm of CLF to incorporate counter offers by providing a multi-directional "announce/refine/decide" paradigm allowing flexible refinement of the negotiation terms. Xplore's protocol consists of the following negotiation verbs. *Open*, it creates a new node in the negotiation tree i.e., creating a new negotiation branch. *Close*, prevents any further development from the current negotiation branch, effectively closing the negotiation on the current options. *Request*, requests information on an aspect of the parameter passed at the current node of the negotiation tree, retrieving information for making informed proposals along a negotiation branch. *Assert*, is used the describe the aspects of a parameter i.e., refining the negotiation term. *Ready* states that the component is ready to enter in the enactment phase in the condition expressed by the passed node. The *Reserve*, *Confirm* verbs allow to first reserve and then consume a resource previously returned as an offer. Split in two phases, the operation of resource consumption (Reserve, then Confirm) allows the customer to perform atomic consumption of resources coming from different offers (possibly by different servers), thus realizing the most basic form of transaction. In addition, the *Cancel* verb allows to cancel a reservation, in case other resources in a transaction become unavailable. Finally, the *Insert* verb requests an extension of a service capability by insertion of a new resource.

An example of using Xplore to describe a NegotiAuction is presented in [12]. NegotiAuction [81] is an algorithmic Internet-based auction procedure. It combines various elements from negotiation and auction protocols, supports multiple attributes of the auctioned good and allows both fully automated negotiation as well as semi-automated negotiation process. Each NegotiAuction takes place in a one-to-many market environment. Auction owner sets up the auction and defines the form of auction i.e., reverse or forward and describes the goods and the quantity to be bought (or to be sold) and decides whether the potential bidders should be explicitly invited (closed format), or everybody could qualify for bidding (open format).

Another infrastructure based on Xplore for supporting negotiations in inter-organizational alliances in a flexible way with respect to the autonomy of the partners involved is defined as e-Alliance in [15]. 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. Similarly, the negotiation middleware CooF 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. This structure captures the dependencies between the negotiation interactions. CooF's job is to coordinate/synchronize these different copies of negotiation graphs. The negotiation process can be considered as a Distributed Constraint Satisfaction Problem [40]. The "distribution" part deals with constraint propagation between nodes, while the "satisfaction" part deals with constraint based reasoning and strategic reasoning at each node.

18.3 Negotiation Agents

A negotiation agent can be termed 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 types of negotiation (e.g., auction, reverse auction, bilateral negotiation). A brief overview of these follows.

18.3.1 Auction Based Agents

An auction can be described as the simplest form of negotiation where a customer bids on the price of an item and the provider has the option of either accepting the offer or rejecting it. There are multiple types of auctions such as English, Dutch, first-price and Vickery [52]. A service composition agent that both buys components and sells services through auctions has been discussed in [67]. 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 algorithms used 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 moves 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. However only the scenarios with English auction type of negotiation with no one-on-one negotiation are handled. It is assumed that the agent maintains a probabilistic model of expected outcomes of each auction based on past performance of similar auctions. The agent initially identifies 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 then places bids in these forward/reverse auctions and continues 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. Moreover, the problems of not committing and evaluating each option are solved by limiting the search space to promising offers only.

18.3.2 Trade-Off Based Negotiation Agents

In trade-off based negotiations the concerned parties make tradeoffs 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. A tradeoff based negotiation mechanism for web service procurement using a bilateral protocol to govern interactions between the negotiation parties is used in [63]. Each party can define its own set of evaluation function, utility function and offer generating algorithm. For simplicity 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.

A trade-offs based agent for multi-dimensional goods for the problem of distributed resource allocation has been presented in [25]. It uses a fuzzy similarity to approximate the preference structure of the other negotiator and then uses a hill-climbing technique to explore the space of possible trade-offs for the one that is most likely to be acceptable. Similar approaches have been discussed in [26] and [51]. Trade-off based agents have also been studied in [17, 50, 97].

18.3.3 Negotiation with Uncertain Data

Having as much information as possible about the other parties is important to strengthen one's negotiation capabilities [60]. Unfortunately, more often than not, we only have partial information about the negotiation context [50]. Hence it is very important to be able to manage different types of unknown parameters about the negotiation. An approach for bilateral negotiation under uncertainty, where a negotiator is uncertain as to what offer or counteroffer to make, at a particular step in the negotiation is presented in [59] and [93]. This uncertainty is resolved by making use of the negotiation experience of reputable parties in [93]. The idea is similar to the scenario where suppose, one has been offered a new employment and it is time to negotiate benefits, including salary. The negotiating parties are yourself and the hiring manager. The fact that mostly salaries are negotiable and often vary with specific job responsibilities, a new hire may not have all the information needed to make a good decision. In this case, a natural course of action, is to seek out others who are trustworthy and who may have negotiated salaries with this company in the past, for similar types of jobs and use their data to make an informed decision. So the main idea is of using the negotiation experiences of trusted parties with matching interests as aids in deciding which negotiating alternatives and offers should be employed.

A model for bilateral negotiations that considers the uncertain and dynamic outside options is defined in [45]. Outside options affect the negotiation strategies via their impact on the reservation price. The model is composed of three modules: single-threaded negotiations, synchronized multi-threaded negotiations, and dynamic multi-threaded negotiations. The single-threaded negotiation model provides negotiation strategies without specifically considering outside options. The model of synchronized multi-threaded negotiations builds on the single-threaded negotiation model and considers the presence of concurrently existing outside options. The model of dynamic multi-threaded negotiations expands the synchronized multi-threaded model by considering the uncertain outside options that may come dynamically in the future. A Poison Process is used to simulate the arrival process of uncertain (dynamic) options.

18.3.4 Genetic Algorithm Based Negotiations

Negotiations are a special class of group decision making problems. Multi-party and multi-objective negotiations add a lot of complexity to the already hard problem of negotiation. Such negotiation problems can thus be formulated as constrained multi-objective optimization problems. The main idea is to optimize a series of objectives simultaneously while considering constraints on the system. The Genetic Algorithm (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 popu-

lations of solutions, they are well suited for multi-criterion problems. A weighted sum based genetic algorithm to support multiple-party multiple-objective negotiations have been presented in [71]. The weighted sum approach is used to handle 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-Dimensional matrix, representing the participants and objectives. Similar approaches have been discussed in [22, 62, 96]. A genetic algorithms based approach that evolve Finite State Machines has been presented in [85]. Each individual in the population of FSMs represents a negotiation strategy that competes against other strategies and is modified over time using traditional operators of mutation and cross over. To mutate an FSM, several different operators are used which include changing the target or source of an edge, changing the output or input symbol of an edge and adding or deleting a state or an edge. A repair algorithm ensures that all the FSMs are valid after mutation or crossover operation. A GA based negotiation model using the traditional operators of mutilation and crossover has been presented in [18]. A special penalty based evaluation function is used that measures the prior concessionary behavior of the opponent agent. A negotiation time aware GA based approach has been presented in [9]. The pace of concession of the agent is proportional to the elapsed negotiation time while considering the opponent's payoff gains and the principal of Pareto optimality. Machine learning and bayesian learning have also been used in conjunction with genetic algorithms to achieve satisfactory results [56, 76].

18.3.5 Combinatorial Negotiations

A 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.

A Combinatorial Negotiation based decision-support service (iBundler) for highly constrained negotiation scenarios has been proposed in [69]. iBundler acts as a combinatorial negotiation solver for both multi-item, multi-unit negotiations and auctions. The service can be employed by both negotiating agents and auctioneers in combinatorial auctions. It consists of three main components. The *Manager* agent takes 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. The *Solver* component extends the *iBundler* with the offering

of an XML language for expressing offers, constraints and requirements. The winner determination is modeled as a mixed integer problem similar to the the binary multi-unit combinatorial reverse auction winner determination problem in [75] with side constraints in [73].

18.4 Discussion

As mentioned earlier 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 [33]. In existing literature, this has usually been accomplished in an ad-hoc manner, 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 SLA negotiation involves QoS attributes such as *reliability*, *availability*, *accessibility* and *response time* [94, 95]. These QoS attributes (and others) influence the negotiation protocol and the customer preferences articulation that the negotiation system must support. Hence there may be more than one combination of these attributes that may be suitable under a specified negotiation context. User preferences could be expressed in a variety of ways, e.g., utility functions [25], combination of attributes [23], or fuzzy constraints [50] etc. 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 (REQ 1).
- *Support for heterogeneous negotiation protocols*. In a service oriented system it is very much expected that all the participants using the system may not be similar. They 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 the negotiation protocol for cases where a participant supports multiple ones.
- *Heterogeneous decision model articulation*. 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 [24, 25, 39, 50]. 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 decommitment from previously created SLA [61]. 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*. Unlike traditional software environments, SOAs enable delivery of the same service to different customers with varied quality of service (QoS) requirements [23]. 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 (REQ 4).
- *Simultaneous negotiations*. Since services are not stored or downloadable, the market environment tends to be very dynamic [27]. The ability to create on-the-fly dynamic solutions emphasizes the need of conducting simultaneous negotiation (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 [27]. This entails that the participants are equipped to change their strategies/decisions at runtime (REQ 6), based on market dynamics and changing contexts [70]. 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 [60, 61]. Similarly, in some contexts if a more profitable offer is found, there should be a provision to decommit from the current agreement [74].

Table 18.1 summarizes how the current negotiation systems in the literature perform on the above mentioned requirements. 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. The table 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. Moreover, none of the solutions provide any dependency modeling among different QoS components. Howsoever this is is an extended requirement in composite solutions that often have dependent QoS objectives. For example, if we were to have a composite solution consisting of *taskA* and *taskB* 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 *taskA* but could not find any service for *taskB* that could meet our current

Table 18.1 Summary of automated negotiation systems

Authors	REQ 1	REQ 2	REQ 3	REQ 4	REQ 5	REQ 6
Ashri et al. [6]	✓	✓	n/a	n/a	n/a	n/a
AuctionBot [91]	✗	✗	✓	✓	✗	✗
Bartolini et al. [8]	✓	✓	✗	✗	✗	✗
Benyoucef et al. [11]	n/a	✓	✓	n/a	✗	✗
Bruns et al. [13]	✓	✗	✓	n/a	✓	✗
Comuzzi et al. [19]	✗	✓	✓	✗	✗	✗
Cremona [48]	✓	✗	✓	✓	✗	n/a
DynamiCS [84]	✓	✓	✓	✗	✗	✗
Inspire [36]	✓	✗	✓	✗	n/a	✗
Jonker et al. [34]	✓	n/a	✓	✓	✗	✗
Kasbah [16]	✓	✓	✓	✗	✗	✗
Kim et al. [37]	✓	✓	✗	✗	✗	✗
Lecue et al. [43]	✗	✓	✗	✗	✗	✓
Ludwig et al. [49]	✓	✓	✓	✗	✗	✗
MAGNET [31]	✓	✗	✓	✓	✗	✗
Marco et al. [20]	✓	✓	✗	✓	✗	✗
NegoPlan [72]	✓	✗	✓	✗	✗	✗
Negotiator [14]	✓	✗	✓	✗	✗	✗
PANDA [27]	✓	✓	✓	✓	✗	✗
Paurobally et al. [65]	✓	✗	✗	✗	✗	✗
Rinderle et al. [68]	✓	✓	✗	✗	✗	✗
Skogsrud et al. [77]	✗	✓	✓	✓	✗	✓
Strobel [80]	✓	✓	✗	✗	✗	✗
Tete-a-Tete [42]	✓	✓	✓	✗	✗	✗

objective. It would then be more economical for the composite solution to downgrade *taskA* to the level of *taskB'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 [4, 28, 43, 78, 86] lack such capabilities, and a new standard language that could be used to pass on all these constraints and decision model to the negotiation system is required. 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.

18.4.1 A framework for Web Service Negotiation

In this section, we provide an overview of our solution for the service negotiation problem. Figure 18.3 presents a high level architecture of a negotiation system

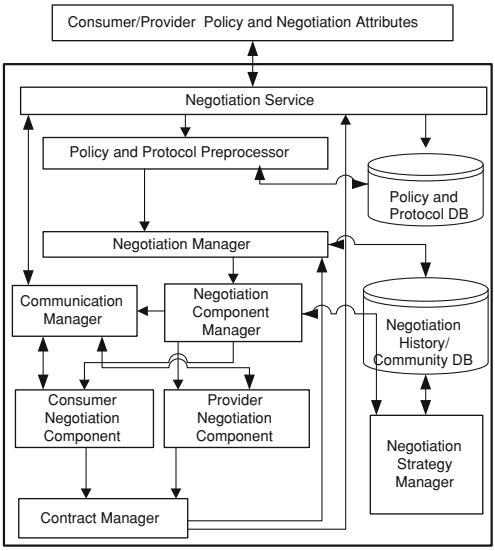

Fig. 18.3 WebNeg system architecture

(defined as *WebNeg*) that is very flexible in terms of its functionality and the services provided. It is primarily targeted to be invoked by the customer searching for a compatible service from a list of service providers providing similar functionalities. The client does not need to implement any negotiation specific component to use the proposed service. The WebNeg architecture is compatible with both the negotiation scenarios i.e., the negotiating participants could either provide their own negotiation component or send all the necessary information to the service, that would handle all the negotiation process. A brief overview of the major modules of the proposed negotiation architecture is as follows:

18.4.1.1 Negotiation Service

The negotiation service layer acts as the interface of the whole system. This layer is responsible for any and all external communication of the system. All the internal components use this service to communicate with both the customer and provider as well as with the community (to be discussed later). Customer invokes the service providing its negotiation attributes, negotiation policy as well its decision model. The service then communicates with potential providers and request their decision model and policy attributes for the negotiation process.

18.4.1.2 Policy and Protocol Preprocessor

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

18.4.1.3 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 then ranks these providers based on their ratings, trust and reputation values.

It uses the trust model based on the concept of community [53] where the reputation represents the perception of the users in that community regarding a service. So, the rating of a service represents the average of all the rating provided to the community for that individual service. For the newly starting service that does not have any history, it uses the reputation bootstrapping mechanism defined in [55]. Community is a centralized knowledge base that would be responsible for storing all the data 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.

18.4.1.4 Negotiation Component Manager

Since negotiation is a multi-party mechanism, the *WebNeg* system needs to spawn separate components for each customer and provider. In the most basic scenario at any given instance, the systems would have one customer and multiple provider

components. These components operate in their separate context and communicate with their original service through the communication manager. The communication manager is responsible for creating and manging these components.

18.4.1.5 Negotiation Strategy Manager

There are multiple strategies available for conducting efficient negotiations. One such strategy is defined below in Sect. 18.4.2 (defined as *WebNeg*). Our 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 could pass on this information as a policy to the system. 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.

18.4.1.6 Communication Manager

All the external pre-contract communications are handled by this manager. The component may communicate with their respective services for any decision model or policy/guidance queries. Communication manager ensures that all the communication is related to the current negotiation and adheres to the negotiation service's policies.

18.4.1.7 Contract Manager

Once the system identifies perspective negotiation solution(s), it is presented to the respective services, if they agree, contract manager then handles all the formal SLA creation process. If the current selected provider does not agree on the solution, the system would then try the next best available solution, until either an agreement is achieved or the system has ran out of options. If the system could not find a mutually agreeable solution, then the process would be termed as a failure and the customer would be asked to revise its negotiation model.

18.4.2 WebNeg

We present a GA based approach to solving the Web service negotiation problem [29]. We enhance the traditional GA with a new operator called *Norm*. Our proposed approach complements the proposed negotiation framework that is designed towards

Table 18.2 Definition of symbols

Symbol	Definition
f_j	Fitness of the solution s for participant j
F_s	Fitness of the solution s (for all participants)
C_j	The value of jth component of Customer's vector
$C_{j(min)}$	The minimum allowed value of jth component of customer's vector as provided by the customer
$C_{j(max)}$	The maximum allowed value of jth component of customer's vector as provided by the customer
WC_j	The weight of jth component of customer's vector as provided by the customer
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 as provided by the provider
$P_{ij(max)}$	The maximum allowed value of jth component of ith Provider's vector as provided by the provider
WP_{ij}	The weight of jth component of ith Provider's vector as provided by the 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

a scenario where a customer is involved in simultaneous negotiations with multiple providers. Each instance of communication among the customer and service provider is private and holds a lot of information. The proposed *Norm* operator makes it possible to share this private information among all the participants without revealing the source of any of such information. This in turn helps all the agents to adapt quickly and significantly reduces the search space by guiding the negotiation process toward a mutually agreeable solution.

We propose a weighted sum genetic algorithm to support multi-party multi-objective negotiation. All the Web services provide their respective QoS parameters to be negotiated. These are called the component vector of a Web service. Each vector is accompanied by a decision model, i.e., ranges of all the QoS parameters as well as their respective priorities also known as the weights. We assume that all the participating Web services are able to articulate their objectives and prioritize them. Table 18.2 lists the definition of symbols used henceforth.

Since all the Web services (participants) start negotiation from a different position, they have different preferences for those objectives, and are described by how far their current position is from the customer's objective. All the Web services conform to some constraints in the solution. For instance, any QoS vector cannot have a negative value (as shown by Eq. 18.1). The QoS values lie between the maximum and minimum allowable values set by the Web service (as shown by Eq. 18.2). A

repair algorithm is applied to GA after each operator, to ensure all these constraints are met.

$$C_j \geq 0\ ,\ P_{ij} \geq 0 \tag{18.1}$$

$$C_{j(min)} \leq C_j \leq C_{j(max)} \text{ and } P_{ij(min)} \leq P_{ij} \leq P_{ij(max)} \tag{18.2}$$

Each gene is a combination of customer and provider chromosomes. If we have n objectives to be negotiated then each gene will have $2n$ chromosomes. The fitness function is a multi-step calculation that evaluates the level of disagreement between the negotiating Web services. A weighted sum approach is used to combine these multiple QoS parameters (objectives). We use a distance function to measure the difference among the proposed solutions of both the customer and provider Web services. Thus, lower fitness values are desired as they translate to lesser disagreement among the participants. Similarly, lower values translate to higher ranks for the solutions among the solution space. Ranks are then used for selection of subsequent steps of the GA [88]. Each solution represents a probable distribution of values that may be agreed upon by the other Web service in the negotiation. The fitness value of a solution is calculated as follows.

$$\Delta_{ij} = \frac{|C_j - P_{ij}|}{C_j} \tag{18.3}$$

$$f_j = \sum_{j=0}^{n} (WC_j * \Delta_{ij} + WP_{ij} * \Delta_{ij}) \tag{18.4}$$

$$F_s = min \sum_{j=0}^{G} (f_j) \tag{18.5}$$

Pareto optimality is not enforced after each generation as it is possible for a Web service to accept a less favorable solution for the time being (in the negotiation process) for a better solution in the long run. However, a secondary population of solutions is kept which is updated after each iteration. This secondary population or *Elitism* is a an important concept in genetic searches [7, 98]. The probabilistic nature of GA does not guarantee that the best solutions would be preserved in the final generation. Hence a secondary population of best solutions is kept through all generations. Below is the algorithm used to determine the optimal solution. Details follow.

Set generation number g equal to zero ($g = 0$)
Generate initial population
Calculate fitness for each member
Store the most fit solution in the secondary population
Rank the solutions
Apply Norm
Select members for crossover using Roulette-Wheel selection method
Perform crossover
Perform random mutations
IF $g = G$ (last generation)
    Ensure Pareto optimality
    exit
ELSE
Set $g = g + 1$
Set Go to step 3

End Algorithm

18.4.2.1 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 transfered through generations and different people follow different norms. Often people are inclined to abandon or follow a new norm on the basis of the facts if they think they are being better off following or deviating from them. Most helpful norms tend to accumulate more followers, which in turn re-enforces that norm. 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 what works the best for them. These norms serve as a guide for achieving their desired goals. Figure 18.4 shows a scenario that depicts the concept of norm. Assume we have n norms (information sources) in the society and k population subsets. Set 1 may follow *Norm 1*, Set 2 may follow *Norm n* and Set m may choose to follow *Norm 2* while others may not choose to follow any *Norm*. The selection of subsets and *Norm* selections are random. Population in Set 1 is effected by the values of *Norm 1* and they in turn effect the values of the *Norm*. This cycle makes sure that beneficial values are prevailed in the *Norms*.

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 criteria is represented as a norm and certain members of the population follow a certain norm. After each generation, the followers update the impact factor of their respective norm. If increasing the value of the norm resulted in a better overall fitness value for the member of population, it would influence the norm into increasing its

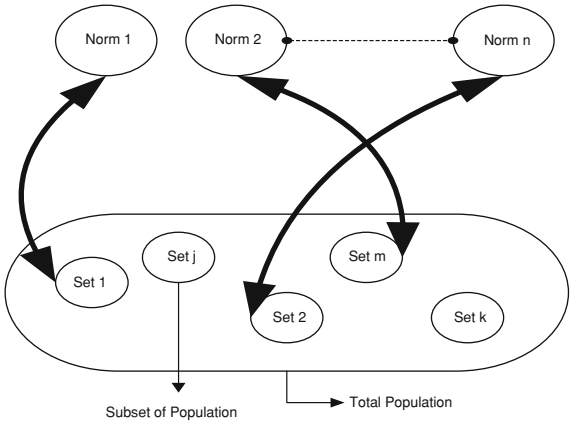

Fig. 18.4 Norm operator in relation to population sets

value. The increase is dependent on the difference of current and previous values of that objective of the reporting individual and the current absolute value of that objective. Both customers and providers share the same influence values of norms. This is an indirect information source for the customer about providers decision model and vice versa. Ideally, we will have one customer and n providers, hence sharing these impact factors does not reveal any trade secrets. These values have the bias of $n+1$ agents and are averaged out.

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

$$E_{ij} = |\frac{C_i}{N_m}| \tag{18.6}$$

and the amount exchanged would be

$$A_{ij} = (1 - WC_i)|1 - \frac{C_i}{N_m}| \tag{18.7}$$

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

$$E_{ij} = |\frac{C_i}{P_{ij}}| \tag{18.8}$$

and the amount exchanged would be

$$A_{ij} = (1 - WC_i)|1 - \frac{C_i}{P_{ij}}| \tag{18.9}$$

18.4.2.2 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 an individual is selected is proportional to its fitness [32].

Solution rankings are used to implement selection. The population is augmented so that solutions with better ranks are more prevalent in the population. We use both ranks and fitness values for our selection technique because ranking indicates the performance of solutions relative to others in the population and minimizes the effect of large disparities in fitness values within the population [88]. Augmentation of the population for roulette-wheel selection is performed as follows:

$$CrossP_j = 1 - \frac{1}{R_j}(R_j - 1) \tag{18.10}$$

Crossover rate is used to determine if crossover will actually occur or if the selected solution will simply be copied over to the next generation. If it is determined that crossover will occur, uniform crossover is implemented on the pair. It has been proved that custom operators provide superior performance for real-valued problems [90].

Mutation is the last operator to act on the population of solutions and is also applied randomly to the elements of the solution, in accordance with the experimentally predetermined mutation rate. A random number is generated for each member in the population and compared to the mutation rate. If the random number is less than or equal to the mutation rate, mutation will occur in that solution. Mutation here involves arbitrarily changing one element of the negotiation vector and then implementing a repair algorithm to ensure that objective values lies within the valid range for that agent.

18.4.2.3 Study and Results

To determine the efficiency to GA with the *Norm* operator we 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.

Traditional GA: A traditional GA was implemented by removing the *Norm* operator. It only uses the simple GA operators of crossover and mutation. All the GA parameters are same as that of 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 customer's Web service gene or provider's Web service gene parameters are augmented. 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. Then all the numbers are aggregated by subtracting their respective minimum values. This summation is then averaged out and either added or subtracted randomly to all the parameters. Then the repair algorithm is applied to ensure that all the constraints from Eqs. 18.1–18.3 are held. Then we add this new solution of our population. The population is then ranked according to their fitness values and members with higher fitness values are taken to the next generation.

Hill-Climber: Hill-climber uses the concept of randomly exchanging the QoS values. It is somewhat similar to the *Norm* operator as both use Eq. 18.10 to determine the amount of the objective to be exchanged. However, the GA with *Norm* uses either Eq. 18.7 or Eq. 18.9 to determine if the exchange will occur, while in hill-climber it is done randomly. Once a gene is randomly selected, the exchange takes place. However, it is guaranteed that only one objective per gene is exchanged and that once selected, that gene does not participate in any other exchange for that generation. We create the initial population and rank them according to their fitness values. We then perform crossover using Roulette-Wheel selection method. Then we apply the mutation operator. After we are done with the basic GA operators we apply the Hill Climbing operator on the population. The repair algorithm ensures that all the constraints from Eqs. 18.1–18.3 are held. All the GA parameters are same as that of GA with *Norm* operator.

Experiment Environment

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. A large number of similar providers are chosen to show the applicability/scalability of the proposed solution. The client negotiated four QoS components of reliability, availability, throughput and accessibility with the providers. We performed 200 iterations consisting of 500 generations each, for all the four algorithms and analyzed the results for efficiency and completeness.

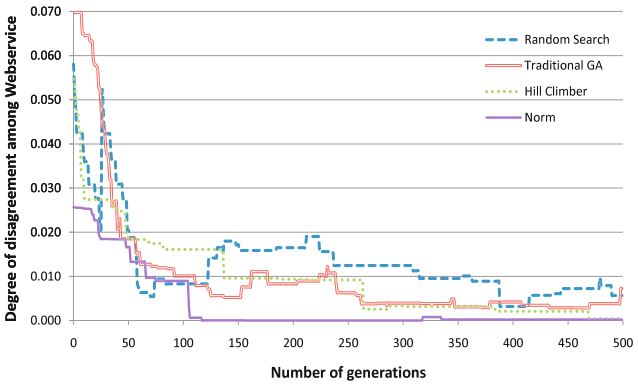

Fig. 18.5 Sample representation of multi-party negotiation

Results

Figure 18.5 shows results of a representative run of the four algorithms after each generation. Note that these are the actual output values without *Elitism* [7, 98]. We have plotted the output of 500 generations (X-axis) against the degree of disagreement (Y-axis) among the client and provider Web services. Lower values of degree of disagreement are desired as they show a higher chance of reaching an agreement e.g., Assume that Web service *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 *A* and *B* is small and hence they are more likely to reach a solution. Note that both the customer and provider must have some overlapping search space values for the algorithm to identify a solution. If both the customer and provider have mutually exclusive ranges of QoS parameters, the algorithm fails and no solution is returned.

The graph confirms the assumption that the probabilistic nature of 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. Our proposed technique (GA with *Norm*) takes almost 1/4th the time to reach an agreeable solution. We can see in the graph 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 find the solution faster with our proposed approach. Similarly our proposed approach finds a lot better solution than any of the other techniques.

The probabilistic nature of GA does not guarantee the same solution every time. Hence, it is appropriate to analyze the performance of GA over multiple rounds. Table 18.3 shows that average of 200 runs for all four algorithms.

We can see that the best solution of 0.00002 returned by *Norm* is far better that best solution returned by any other technique. Similarly *Traditional GA* performed better than the *Hill Climber* in finding a more agreeable solution. As far as the worst solution is concerned, *Norm* still performed better than any of the other techniques.

Table 18.3 Average results over 200 iterations

	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

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* shows a remarkable improvement from the next best i.e., *Hill Climber* technique, depicting that the *Norm* also has the best average case performance among the compared techniques. Similarly our proposed technique has the lowest standard deviation of 0.01157. Lowest mean value combined with the lowest standard deviation indicates that our technique performs consistently better than other techniques.

These results suggest that our approach outperforms other compared methods in terms of finding the optimal solution in the amount of time it takes to find that solution.

18.5 Conclusion and Future Directions

Designing an automated, flexible and efficient negotiation system that facilitates the Web service selection process, is challenging. None of the existing solutions meet all the requirements for a completely automated solution for Web service negotiation. One of the limitations of the presented techniques involve 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. Secondly, most of the solutions use a priori decision model articulation, which requires that all the negotiating participants can identify and share their preferences at the beginning of the negotiation. However, some of these limitations involved with the static environment assumption could be overcome, if participants decide to provide their own negotiating component rather than only articulating their preferences. However, this limits the effectiveness of sharing private information. Therefore, we need to design a negotiation system that can support multiple communication protocols for enabling interactions among different customers and providers as well as supporting multiple negotiation strategies for an optimized solution. The solution should support multiple simultaneous negotiations and provide mechanisms to model the dependency relationships among different component services to achieve an optimal solution.

In this chapter we have presented the framework for Web services negotiation to enable customers and providers negotiating QoS parameters in SLA's. The presented architecture uses a GA based approach to conduct multi-party multi-objective negotiations. Our approach 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*. This operator is based on the concept of cumulative knowledge of the society over a period of time. This accumulated knowledge influences the decision making process of negotiating participants. Furthermore, *Norm* provides a platform for sharing private information of all the participants of the negotiation in such a manner that allows for using this shared knowledge for the overall gain of the society, without revealing the identity of information providers. We have compared *Norm's* performance with similar optimization techniques i.e., Traditional GA, Hill-Climbing and Random Search. The results show that our proposed technique performs better than any of the above mentioned techniques, and that applying a genetic algorithm based approach to complex negotiation for Web service composition problems is a viable option.

We are currently investigating on enhancing the effectiveness of private information sharing by exploring the possibilities of having people follow multiple information sources rather than following just one source. This is motivated by the fact that composite solutions often have dependent objectives. We want to further extend our approach to incorporate these dependencies among the different QoS parameters of multiple services to formulate optimized solutions. We need to be able to use the information sources of *Norm* operator to share such information. We need the negotiation service to be able to simultaneously negotiate multiple service having multiple objectives with multiple providers. Existing communication protocols [4, 28, 43, 78, 86] 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 [30] and WS-AgreementNegotiation [89] by adding the support of complex logical functions for articulating these and similar complex decision models. We are also working on a solution that moves away from the centralized approach in the favor of a more adaptive distributed model.

References

1. Alhosban, A., Hashmi, K., Malik, Z., Medjahed, B.: Assessing fault occurrence likelihood for service-oriented systems. In: Proceedings of the 11th International Conference on Web, Engineering, pp. 59–73 (2011)
2. Alhosban, A., Hashmi, K., Malik, Z., Medjahed, B.: S2r: a semantic web service similarity and ranking approach. Int. J. Next-Gener. Comput. **3**(2) (2012). http://perpetualinnovation.net/ojs/index.php/ijngc/article/view/145
3. Andreoli, J., Arregui, D., Pacull, F., Rivire, M., Vion-dury, J., Willamowski, J.: Clfmekano: a framework for building virtual-enterprise applications. In: Proceedings of the EDOC'99 (1999)
4. Andreoli, J.M., Castellani, S.: Towards a flexible middleware negotiation facility for distributed components. In: International Workshop on Database and Expert Systems Applications 0732 (2001)
5. Andrieux, A., Dan, A., Keahy, K., Ludwig, H., Rofrano, J.: From ws-agreement to sla negotiation (2004). http://www.mcs.anl.gov/keahey/Meetings/GRAAP/WS-AgreementNegotiabilityConstrains.pdf
6. Ashri, R., Rahwan, I., Luck, M.: Architectures for negotiating agents. In: Proceedings of the 3rd Central and Eastern European conference on Multi-agent systems, pp. 136–146 (2003)

7. Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms, pp. 101–111 (1985)
8. Bartolini, C., Preist, C., Jennings, N.R.: A software framework for automated negotiation. In: SELMAS, Lecture Notes in Computer Science, vol. 3390, pp. 213–235. Springer (2004)
9. Beheshti, R., Rahmani, A.T.: A multi-objective genetic algorithm method to support multi-agent negotiations. In: Second International Conference on Future Information Technology and Management Engineering, 2009. FITME '09, pp. 596–599 (2009). doi:10.1109/FITME.2009.154
10. Benbernou, S., Brandic, I., Cappiello, C., Carro, M., Comuzzi, M., Kertész, A., Kritikos, K., Parkin, M., Pernici, B., Plebani, P.: Modeling and negotiating service quality, in service research challenges and solutions for the future internet—s-cube—towards engineering, managing and adapting service-based systems. In: Papazoglou, M.P., Pohl, K., Parkin, M., Metzger A. (eds.) S-CUBE Book, Lecture Notes in Computer Science, vol. 6500, pp. 157–208. Springer (2010)
11. Benyoucef, M., Verrons, M.H.: Configurable enegotiation systems for large scale and transparent decision making. Group Decis Negot **17**(3), 211–224 (2008)
12. Brandl, R., Andreoli, J., Castellani, S.: Ubiquitous negotiation games: a case study. In: Proceedings of the DEXA e-negotiations, Workshop (2003)
13. Bruns, G., Cortes, M.: A hierarchical approach to service negotiation. In: IEEE International Conference on Web Services, pp. 460–467 (2011)
14. Bui, T.X., Shakun, M.F.: Negotiation processes, evolutionary systems design, and negotiator. Group Decis Negot **5**(10), 339–353 (1996)
15. Castellani, S., Andreoli, J., Bratu, M., Boissier, O., Alloui, I., Megzari, K.: E-alliance: a negotiation infrastructure for virtual alliances (2002)
16. Chavez, A., Maes, P.: 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, pp. 75–90 (1996)
17. Cheung, S.C., Hung, P.C.K., Chiu, D.K.: On the e-negotiation of unmatched logrolling views. In: Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36) (2003)
18. Choi, S.P.M., Liu, J., Chan, S.: A genetic agent-based negotiation system. Comput. Netw. **37**(2), 195–204 (2001)
19. Comuzzi, M., Pernici, B.: Negotiation support for web service selection. In: TES (2004)
20. Comuzzi, M., Pernici, B.: An architecture for flexible web service qos negotiation. In: Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference, pp. 70–82 (2005)
21. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the web services web: an introduction to soap, wsdl, and uddi. Internet Comput. IEEE **6**(2), 86–93 (2002)
22. Deng, M.D., Li, J.: An agent negotiation system based on adaptive genetic algorithm. In: 2009 5th International Conference on Wireless Communications Networking and Mobile Computing, vol. 18, pp. 5307–5310 (2009)
23. Elfatatry, A., Layzell, P.J.: A negotiation description language. Softw. Pract. Exp. **35**(4), 323–343 (2005)
24. Faratin, P., Sierra, C., Jennings, R.: Negotiation decision functions for autonomous agents. Robot. Auton. Syst. **24**(3–4), 159–182 (1998). http://eprints.ecs.soton.ac.uk/2117/
25. Faratin, P., Sierra, C., Jennings, N.R.: Using similarity criteria to make issue trade-offs in automated negotiations. Artif. Intell. **142**, 205–237 (2002)
26. Freuder, E.C., O'Sullivan, B.: Modeling and generating tradeoffs for constraint-based configuration (2001)
27. Gimpel, H., Ludwig, H., Dan, A., Kearney, B.: Panda: specifying policies for automated negotiations of service contracts, pp. 287–302 (2003)
28. (GRAAP) G.R.A.A.P.: Wsagreement (2007). http://www.ogf.org/documents/GFD.107.pdf
29. Hashmi, K., Alhosban, A., Malik, Z., Medjahed, B.: Webneg: A genetic algorithm based approach for service negotiation. In: Proceedings of the 2011 IEEE International Conference

on Web Services, ICWS '11, pp. 105–112. IEEE Computer Society, Washington, DC, USA (2011). doi:10.1109/ICWS.2011.55.http://dx.doi.org/10.1109/ICWS.2011.55
30. Hung, P.C.K., Li, H., Jeng, J.: Ws-negotiation: an overview of research issues. In: Proceedings of the 37th Hawaii International Conference on System Sciences (2004)
31. Jaiswal, A., Kim, Y., Gini, M.L.: Design and implementation of a secure multi-agent marketplace. Electron. Commer. Res. Appl. **3**(4), 355–368 (2004)
32. James, E.B.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the Second International Conference on Genetic Algorithms and their application, pp. 14–21 (1987)
33. Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra, C., Wooldridge, M.: Automated negotiation: prospects, methods and challenges. Int. J. Group Decis. Negot. **10**(2), 199–215 (2001). http://eprints.ecs.soton.ac.uk/4231/
34. Jonker, C., Robu, V., Treur, J.: An agent architecture for multi-attribute negotiation using incomplete preference information. Auton. Agents MultiAgent Syst. **15**, 221–252 (2007)
35. Keller, A.: openccs: Computing center software. Technical report, Aderborn Center for Parallel Computing (2007)
36. Kersten, G.E., Noronha, S.J.: Www based negotiation support: design, implementation and use. Decis. Support Syst. **25**(2), 135–154 (1999)
37. Kim, J., Segev, A.: A web services-enabled marketplace architecture for negotiation process management. Decis. Support Syst. **40**, 71–87 (2005)
38. Kit, C.M., Woo, C.C.: A speech-act-based negotiation protocol: design, implementation, and test use. ACM Trans. Inf. Syst. **12**(4), 360–382 (1994)
39. Kowalczyk, R.: Fuzzy e-negotiation agents. Soft Computing—a fusion of foundations, methodologies and applications **6**, 337–347 (2002). doi:10.1007/s00500-002-0187-5
40. Kowalczyk, R., Bui, V.: Jfsolver: a tool for modeling and solving fuzzy constraint satisfaction problems. In: FUZZ-IEEE, pp. 304–307 (2001)
41. Kritikos, K., Plexousakis, D.: Requirements for qos-based web service description and discovery. IEEE Trans. Serv. Comput. **2**(4), 320–337 (2009). doi:10.1109/TSC.2009.26
42. Lab, M.M.: Teteatete (2000). Online: ecommerce.media.mit.edu.
43. Lecue, F., Wajid, U., Mehandjiev, N.: Negotiating robustness in semanticweb service composition. In: Seventh IEEE European Conference on Web Services (2009)
44. Levy, R., Nagarajarao, J., Pacifici, G., Spreitzer, M., N.Tantawi, A., Youssef, A.: Performance management for cluster based web services. In: IFIP/IEEE 8th International Symposium on Integrated Network Management (2003)
45. Li, C., Giampapa, J., Sycara, K.: Bilateral negotiation decisions with uncertain dynamic outside options. IEEE Trans. Syst. Man Cybern. **36**(1), 45–55 (2006)
46. Lin, C., Lu, S., Lai, Z., Chebotko, A., Fei, X., Hua, J., Fotouhi, F.: Service-oriented architecture for view: a visual scientific workflow management system. In: SCC '08, Proceedings of the 2008 IEEE International Conference on Services Computing, pp. 335–342. IEEE Computer Society, Washington, DC, USA (2008). http://dx.doi.org/10.1109/SCC.2008.118
47. Lomuscio, A.R., Wooldridge, M., Jennings, N.R.: A classification scheme for negotiation in electronic commerce. Group Decis. Negot. **12**(1), 31–56 (2004)
48. Ludwig, H., Dan, A., Kearney, R.: Cremona: an architecture and library for creation and monitoring of ws-agreements. In: 2nd International Conference on Service Oriented Computing (2004)
49. Ludwig, A., Braun, P., Kowalczyk, R., Franczyk, B.: A framework for automated negotiation of service level agreements in services grids. In: Bussler, C., Haller, A. (eds.) Business Process Management Workshops 2005, vol. 3812, pp. 89–101 (2005)
50. Luo, X., Jennings, N.R., Shadbolt, N., Leung, H., Lee, J.: A fuzzy constraint based model for bilateral multi-issue negotiations in semi-competitive environments. Artif. Intell. J. **148**(1–2), 53–102 (2003)
51. Luo, X., Jennings, N.R., Shadbolt, N.: Acquiring user strategies and preferences for negotiating agents: a default then adjust method. Int. J. Human Comput. Stud. **64**(4), 304–321 (2006)

52. Maasland, E., Onderstal, S.: Going, going, gone! a swift tour of auction theory and its applications. De Economist **154**, 197–249 (2006). http://dx.doi.org/10.1007/s10645-006-9002-5. doi:10.1007/s10645-006-9002-5
53. Malik, Z., Bouguettaya, A.: Evaluating rater credibility for reputation assessment of web services. In: WISE'07: Proceedings of the 8th International Conference on Web Information Systems Engineering, pp. 38–49. Springer (2007)
54. Malik, Z., Bouguettaya, A.: Rateweb: reputation assessment for trust establishment among web services. VLDB J. **18**(4), 885–911 (2009). doi:dx.doi.org/10.1007/s00778-009-0138-1
55. Malik, Z., Bouguettaya, A.: Reputation bootstrapping for trust establishment among web services. Internet Comput. IEEE **13**(1), 40–47 (2009)
56. Matwin, S., Szapiro, T., Haigh, K.: Genetic algorithms approach to a negotiation support system. IEEE Trans. Syst. Man Cybern **21**(1), 102–114 (1991)
57. Michlmayr, A., Rosenberg, F., Leitner, P., Dustdar, S.: End-to-end support for qos-aware service selection, binding, and mediation in vresco. IEEE Trans. Serv. Comput. **3**(3), 193–205 (2010)
58. Mobach, D., Overeinder, B., Brazier, F.: A ws-agreement based resource negotiation framework for mobile agents. Scalable Comput. Pract. Exp. **7**(1), pp. 23–26 (2006)
59. Mudgal, C., Vassileva, J.: Bilateral negotiation with incomplete and uncertain information: a decision-theoretic approach using a model of the opponent. In: Klusch, M., Kerschberg, L. (Eds.) Cooperative Information Agents IV, LNAI, pp. 107–118. Springer-Verlag (2000)
60. Nguyen, T.D., Jennings, N.R.: Bayesian learning in negotiation. Int. J. Hum.-Comput. Stud. **48**(1), pp. 125–141 (1998)
61. Nguyen, T.D., Jennings, N.R.: Managing commitments in multiple concurrent negotiations. Electron. Commer. Res. Appl. **4**(4), 362–376 (2005)
62. Niu, X., Wang, S.: Genetic algorithm for automatic negotiation based on agent. In: 7th World Congress on Intelligent Control and Automation, 2008. WCICA 2008, pp. 3834–3838 (2008)
63. Patankar, V., Hewett, R.: Automated negotiation in web service procurement. In: Proceedings of the Third International Conference on Internet and Web Applications and Services (2008)
64. Paurobally, S., Aart, C.V., Tamma, V., Wooldridge, M., Hapert, P.V.: Web services negotiation in an insurance grid. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (2007)
65. Paurobally, S., Tamma, V., Wooldrdige, M.: A framework for web service negotiation. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 2(4) (2007)
66. Pichot, A., Waldrich, O., Ziegler, W., Wieder, P.: Towards dynamic service level agreement negotiation: an approach based on ws-agreement. In: 4th International Conference on Web Information Systems and Technologies, WEBIST 2008, Funchal, Madeira, Portugal (2008)
67. Preist, C., Bartolini, C., Byde, A.: Agentbased service composition through simultaneous negotiation in forward and reverse auctions. In: Proceedings of the 4th ACM conference on Electronic commerce, pp. 55–63. ACM (2003)
68. Rinderle, S., Benyoucef, M.: Towards the automation of e-negotiation processes based on web services a modeling approach. In: WISE 05, pp. 443–453 (2005)
69. Rodrguez-Aguilar, J.A., Giovanucci, A., Reyes-Moro, A., Noria, F.X., Cerquides, J.: Agent-based decision support for actual-world procurement scenarios. In: Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (2003)
70. Ros, R., Sierra, C.: A negotiation meta strategy combining trade-off and concession moves. J. Auton. Agent Multiagent Syst. **12**, 163–181 (2006)
71. Rubenstein-Montano, B., Malaga, R.A.: A weighted sum genetic algorithm to support multiple-party multiple-objective negotiations. IEEE Trans. Evol. Comput. **6**(4), 366–377 (2002)
72. Matwin, S., Szpakowicz, S., Koperczak, Z.: Negoplan: an expert system shell for negotiation support. IEEE Expert **4**(4), 50–62 (1996)
73. Sandholm, T., Suri, S.: Side constraints and non-price attributes in markets. In: International Joint Conference on Artificial Intelligence (IJCAI), (2001)
74. Sandholm, T.W., Lesser, V.R.: Leveled commitment contracts and strategic breach. Games Econ. Behav. **35**, 212–270 (2001)

75. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: Winner determination in combinatorial auction generalizations. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (2002)
76. Sim, K.M., Guo, Y., Shi, B.: Blgan: Bayesian learning and genetic algorithm for supporting negotiation with incomplete information. IEEE Trans. Syst. Man Cybern. B **39**(1), 198–211 (2009)
77. Skogsrud, H., Motahari-Nezhad, H., Benatallah, B., Casati, F.: Modeling trust negotiation for web services. Computer **42**(2), 54–61 (2009). doi:10.1109/MC.2009.56
78. Smith, R.G.: The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Trans. Comput. **C-29**(12), 1104-1113 (1980)
79. Standard, O.: Wsbpel (2005). http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.
80. Strobel, M.: Design of roles and protocols for electronic negotiations. Electron. Commer. Res. **1**, 335–353 (2001)
81. Teich, J., Wallenius, H., Wallenius, J., Zaitsev, A.: An internet-based procedure for reverse auctions combining aspects of negotiations and auctions. In: DEXA '00: Proceedings of the 11th International Workshop on Database and Expert Systems Applications (2000)
82. Tosic, V., Bernard, P., Kruti, P., Babak, E., Wei, M.: Management applications of the web service offerings language (wsol). Inf. Syst. **30**(7), 564–586 (2005)
83. Treiber, M., Andrikopoulos, V., Dustdar, S.: Calculating service fitness in service networks. In: ICSOC/ServiceWave Workshops, pp. 283–292 (2009)
84. Tu, M., Seebode, C., Griffel, F., Lamersdorf, W.: Dynamics: an actor-based framework for negotiating mobile agents **1**, 101–117 (2001)
85. Tu, M.T., Wolff, E., Lamersdorf, W.: Genetic algorithms for automated negotiations: a fsm-based application approach. In: Proceedings of the 11th International Workshop on Database and Expert Systems Applications, pp. 1029–1033 (2000)
86. (W3C) W.W.W.C.: Wspolicy (2006). http://www.w3.org/Submission/WS-Policy/.
87. Waldrich, O., Wieder, P., Ziegler, W.: A meta-scheduling service for co-allocating arbitrary types of resources. In: Parallel Processing and Applied Mathematics. Lecture Notes in Computer Science, vol. 3911/2006. Springer, Berlin (2006)
88. Whitley, D.: 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, pp. 116–121. Morgan Kaufmann Publishers Inc., San Francisco (1989)
89. Wieder, P.: Ws-agreementnegotiation (2010). http://forge.gridforum.org/sf/go/doc15831
90. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
91. Wurman, P.R., Wellman, M.P., Walsh, W.E.: The michigan internet auctionbot: a configurable auction. In: Second International Conference On Autonomous Agents, pp. 301–308 (1998)
92. Yao, Y., Yang, F., Su, S.: Evaluating proposals in web services negotiation. In: Computer and Information Sciences ISCIS 2006, pp. 613–621. Springer, Berlin (2006)
93. Yee, G., Korba, L.: Bilateral e-services negotiation under uncertainty. In: Proceedings of the 2003 Symposium on Applications and the Internet (2003)
94. Yu, Q., Liu, X., Bouguettaya, A., Medjahed, B.: Deploying and managing web services: issues, solutions, and directions. VLDB J. **17**(3), 537–572 (2008). doi:dx.doi.org/10.1007/s00778-006-0020-3
95. Zarras, A., Vassiliadis, P., Issarny, V.: Model-driven dependability analysis of webservices. In: Web Services, International Symposium on Distributed Objects and Applications, pp. 69–79 (2004)
96. Zhai, D., Wu, Y., Lu, J., Yan, F.: A fuzzy negotiation model with genetic algorithms. In: I3E (1)'07, pp. 35–43 (2007)
97. Zhu, J.: A buyer-seller game model for selection and negotiation of purchasing bids: extensions and new models. Eur. J. Oper. Res. **154**(1), 150–156 (2004). http://EconPapers.repec.org/RePEc:eee:ejores:v:154:y:2004:i:1:p:150--156
98. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**, 173–195 (2000)