Towards Understanding Negotiation Strategies: Analyzing the Dynamics of Strategy Components

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

A.S.Y. Dirkzwager
born in Hong Kong, China

Interactive Intelligence Group
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://ii.tudelft.nl/

Towards Understanding Negotiation Strategies: Analyzing the Dynamics of Strategy Components

Author: A.S.Y. Dirkzwager
Student id: 1397834
Email: A.S.Y.Dirkzwager@student.tudelft.nl

Abstract

With modern computers becoming more powerful, automated negotiation is coming more to the forefront of research. Software agents are able to aid human negotiators, help them achieve better outcomes and even negotiate on their behalf. An active topic in the field of automated negotiation concerns the development of negotiation strategies. While negotiation strategies and their components have been investigated before, there is to our knowledge no substantial work that analyzes the relations between the components. This thesis investigates these relationships, contributing to the field in two ways. First, it introduces the BOA architecture, which defines and isolates the key components of a negotiation strategy. To validate this architecture we have shown that many well-known and state-of-the-art negotiation strategies fit into the BOA architecture. Second, it investigates how the interactions of these key components influence an agent's performance. We examined the effects of combining different components and we determined as well which key component contributes the most to the agent's performance. We have reached the important conclusion that combining better performing components does lead to better performing negotiation strategies. Furthermore, we found a significant difference in the amount of contribution each key component makes to the performance of a negotiation agent. Throughout our research we have also found that the interactions between components play a large role as well in the performance of a negotiation strategy and that these interactions should not be overlooked.

Thesis Committee:

Chair:	Prof.dr. C.M. Jonker
Supervisor:	Dr. K.V. Hindriks
External supervisor:	Dr. M.V. Dignum
Daily supervisor:	Drs. T. Baarslag

Preface

This thesis was written for the masters program "Computer Science-Media Knowledge Engineering" at the Delft University of Technology.

After attending an artificial intelligence course, where we had to develop our own automated negotiation agent, as part of the course syllabus. I became interested in automated negotiations. I found it very interesting and challenging to devise a strategy which would allow a negotiation agent to negotiate effectively against other agents in different domains. After developing our own negotiation agent we submitted it to an international Automated Agent Competition in 2011 (ANAC 2011). Our agent performed well placing in the top five of the qualifying round allowing us to proceed to the final round. Due to our placement in the final round we were invited to attend the ACAN workshop which formed part of the 2011 Autonomous Agents and Multi-Agent Systems conference (AAMAS 2011) in Taipei, Taiwan. At this workshop we presented our negotiation strategy which finished 6^{th} in th competition. With my interest in negotiation strategies, I directed my thesis towards understanding them, looking at the different components and examining how they relate to each other. Next to that I also co-authored three papers which are related to my research.

I would like to thank Tim Baarslag, Koen Hindriks, and Catholijn Jonker for their feedback and support during my thesis. Tim Baarslag in particular has been very supportive when I was faced with daunting challenges while preparing my thesis. I also would like to thank Mark Hendrikx, for his efforts as co-author and joint developer of the BOA architecture and our ANAC 2011 and ANAC 2012 agents. In addition, I would like to thank the Universiteitsfonds Delft and the Interactive Intelligence Group of Delft University of Technology for sponsoring my trips to the AAMAS 2011 and AAMAS 2012. Furthermore, I thank Bart Vastenhouw and Ruud de Jong for their help in acquiring the large number of computers required to run all our experiments. Last but definitely not least, I thank my family for their help and support they gave me during those tough times.

A.S.Y. Dirkzwager
Delft, the Netherlands
October 17, 2013

Contents

Chapter 1

Introduction

Throughout our lives we are constantly faced with choices, making decisions about trivial matters such as: *what tie to wear*, more important issues such as *where to go on vacation* or matters of life changing importance, *should one work for this company or that company*. Making decisions is vital in our everyday lives.

There are times however, when decisions are not only up to one person but involve others. This makes the decision process more complex, as one needs to take the preferences of the other participants into account. When involving multiple participants there will be conflicting interests, because not all the participants will want to do the same things. To illustrate this we use the following example: Jack is planning a vacation and has asked Jill to come along. Now together they need to pick a suitable destination; Jack is interested in a destination which has plenty of outdoor activities, while Jill would prefer a location with indoor activities. Jill is also more interested in a vacation destination which is known for its local traditions. This conflicts with Jack's interests as he would rather go somewhere somewhat more touristic.

From this example it is clear that the interests of Jack and Jill are conflicting, making it at first impossible for them to get exactly what they want. To be able to resolve this Jack and Jill will need to make compromises. There are different ways to resolve conflicts and disputes, involving mediation or arbitration, but the one that we are going to focus on is *negotiation*.

1.1 Negotiation

Negotiation is the process by which different parties communicate with one another to try and come to a mutually acceptable agreement [87]. Negotiation happens all around us whether one realizes it or not. We negotiate with co-workers, partners, friends even with ourselves; we have been negotiating ever since we were big enough to disagree with our parents about eating vegetables [55]. A good example of where negotiation is used is in the business world. Businesses need resources whether it be in goods or services in order to be profitable. In a world where these resources are limited, businesses need to make deals and must overcome their conflicting interests to do so. In order for businesses to be

able to work together contracts have to be made, which establish how goods or services will be exchanged. These contracts represent a mutually acceptable agreement, usually made after both parties have negotiated with each other. If the negotiation leads to an agreement all parties will benefit to some degree as the agreement allows them to achieve their individual goals as well as their shared objectives. A negotiation process will begin when both, or all, parties accept that it is in their interest to cooperate, the only issue being whether the outcome is acceptable or not [37]. To be able to negotiate successfully one needs to take different factors into account; what is the objective of the negotiation, in what type of setting is the negotiation taking place, what are the interest of the opponent and is there any type of pressure or constraints (resources, time, etc.) which will affect the negotiation process. All these factors make negotiation a very complex process. Research in negotiation has been done in various academic disciplines such as economics [98, 105], electronic commerce [20, 72, 87, 93, 96], artificial intelligence [39, 43, 60, 73, 74, 78, 116], game theory [19, 21, 43, 60, 78, 98, 103, 110], and social psychology [109].

Negotiation can take place between two parties (bilateral negotiations), one against many (e.g. in an auction setting) or many against many (multilateral negotiation) for example at the United Nations. Sometimes to keep the negotiation process fair, a third party is involved to mediate or arbitrate the negotiation. The simplest negotiation process however, is where there are two parties negotiating on a single issue, for example the splitting of a pie [90, 110], where the only question to be settled is the size of the pie each will get. More complex negotiations involve multiple issues. Let us take the example of the vacation destination of Jack and Jill; there is a range of issues that they need to agree upon; what kind of atmosphere or amusement they want at the destination or the type of shopping or activities that can be done there. While a negotiation with multiple issues is more complex, it has a better chance of reaching a win-win outcome as the presence of more than one issue provides the parties with more potentially acceptable compromises. In a single issue negotiation on the other hand, a win for one party in most cases imply a loss for the other. With a multi-issue negotiation this does not have to be the case. One party can place particular value on issue x, while the other party is more interested in issue y. This allows the negotiators to make trade-offs and concessions that are beneficial for their opponents, but do not lower their own goals. For example let us say that a consumer places a lot of value on the quality of a product, while the seller places more value on the price. If the consumer is willing to pay more for quality goods both the parties will be happy, a win-win situation. This work will mainly focus on the process of bilateral negotiation.

1.2 Negotiation Software

Negotiations can be a very complex, time consuming and sometimes expensive process. Depending on the type and subject of the negotiation and the personalities of the negotiators, a negotiation can be quite lengthy and can range from a couple of days to a few years. During these periods many meetings will be held, where both parties come together to discuss and deliberate over the possible outcomes. This is not only very time consuming but can also be very expensive, as there may be a team of negotiators who might be required

to travel long distances to these meetings.

There is therefore a growing interest in negotiation software [17, 44, 60, 74], especially in the emerging market of e-commerce [20, 72, 87, 93]. With modern computers becoming more powerful and more capable of handling the complexities of negotiations, they are able to aid in the negotiation process. Computers can help human negotiators to negotiate in a more effective and efficient fashion by providing assistance in organizing information, suggesting perceived acceptable bids or possible counter offers. With modern day computers and the global communication networks we are able to bridge the physical gap between the two parties by allowing teleconferencing, in which both parties can negotiate with each other, without having to travel across the world to do so.

Negotiation Support Systems

Research has been done on *Negotiation Support Systems* (NSS) [52, 67, 68, 69, 101], which are systems that aid human negotiators to achieve better outcomes. Rangawamy and Shell distinguishes between two types of support that a NSS can provide [107]; one is to aid the negotiator in preparing for the negotiation, for example by eliciting preferences from the users, organize information and determine pre-negotiation strategies. The other type of support a NSS can give is mediation and interactive support, providing assistance during the negotiation, suggesting concessions or compromises or whether to accept the opponent's offer.

Research by Pommeranz et al. [101] has established certain guidelines in making an effective NSS. These guidelines were derived from in depth analysis of negotiations, not only examining theoretical research but also real world practices, involving general negotiation experts and job negotiation experts. Some of these guidelines are:

- Advice from a NSS should consider information about the context of the negotiation.
- A NSS should support the user by calculating bids and offering new options to negotiate on.
- It should have a data storing and management function that gives the user easy access to the information needed at a certain point in time.
- A NSS should generally provide the user with more generic advice that the user can apply to the situation he/she is in.

Over the years different NSS have been developed like the *INSPIRE* [67], which was the first web based negotiation platform. It was created to assist international negotiations, to teach and train negotiators in several countries and to provide support before (preparations), during (decision support) and after (analysis) the negotiation. Another example of a NSS is the *Pocket Negotiator* developed by Hindriks et al. [52]. The goal of the Pocket Negotiator is to provide valuable support to human negotiators on arbitrary domains and scenarios. Being able to run on portable devices (e.g. laptops, smartphones, tablets) makes it very versatile and allows it to be employed in multiple situations. A paper by Hendriks et al. [52], remark that to be able to resolve the fundamental problems faced by human negotiations, NSS

should be combined/integrated with negotiating software agents. *ASPIRE* developed by Kersten et al. implements this idea [68, 69], building upon INSPIRE to include negotiating software agents to further improve assistance during the negotiation by providing context-dependent support, especially about its tactics and strategies.

Negotiating Software Agents

Software agents can be very useful in a NSS providing added support to the human negotiator however, a software agent is also capable of negotiating on the behalf of the human negotiator, automating (part of) the negotiation process.

There is no clear universal definition for a software agent, however the definition which fits best in our context is: *a software agent is a computer program which is designed to achieve a certain objective whilst interacting with its environment* [59]. Nwana's paper [94] defines three attributes which make a smart software agent: *Collaboration, Autonomy* and *Learning* (figure 1.1). These three attributes are also important in automated negotiations; collaboration is a desired attribute in a negotiation since the essence of a negotiation is that both parties work together to reach a satisfying outcome. Autonomy is also an important attribute, since the negotiation agent will be negotiating on behalf of or aiding the human negotiator. The software agent should be free to control its behavior, making its own decisions and actions. If that were not the case, then the human negotiator would constantly need to monitor the agent which would defeat the purpose of an automated negotiation process. A negotiation agent also needs to be able learn from the opponent and adapt to its behavior in order to achieve the best possible results. Negotiation is about action and reaction, so if the agent has learned from its opponent it can predict its behavior and thus anticipate what the opponent will do.

Negotiating software agents can also be useful as a training tool to help improve human negotiation skills. Lin et al. [79, 80] compare the standard training techniques with new techniques involving automated negotiation agents. They show that there was significant improvement in people's negotiation skills, which was noted throughout their experiments. Other advantages of automated negotiators are that they are often rational, reducing the effects of outside influences like emotions. Human negotiators are not always objective and rational when negotiating; for example a negotiator might feel intimidated by the opponent and therefore accepts a sub-optimal agreement. This factor explains why automated negotiation agents can sometimes achieve better results than a human negotiator [20, 58, 87, 99].

Negotiation Environment

To be able to support automated negotiations, the environment where these software agents negotiate in need to be formalized. A negotiation *scenario* is commonly used to describe the setting of a negotiation. It consists of a *domain*, an *outcome space* and two *preference profiles*. A negotiation *domain* specifies the *issues* that the negotiators need to agree upon, for example the "Travel" domain [13] has 7 issues: Atmosphere, Amusement, Culinary,

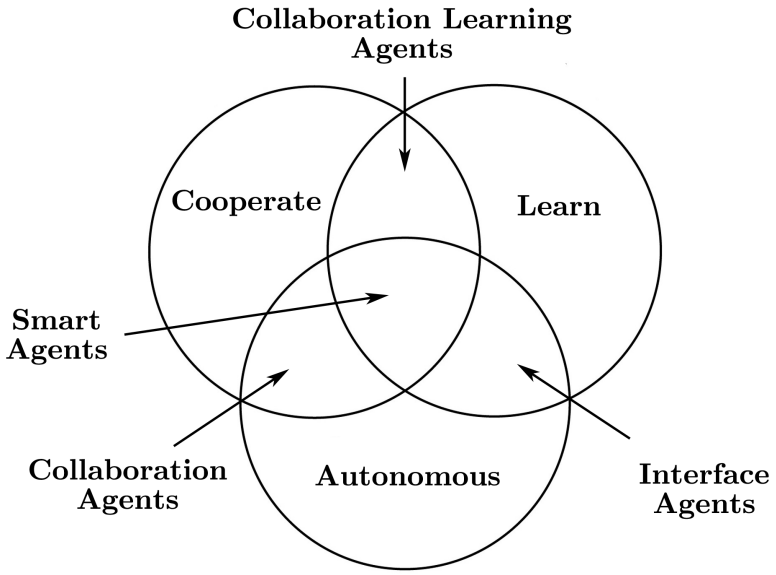

Figure 1.1: The three attributes make a software agent. Taken from [94].

Shopping, Culture, Sport, Environment, and for every issue there is an associated range of *values* that can be chosen. These value sets can either be discrete integer values, real values or nominal values. For the Travel domain the issues vary in the number of values (7, 7, 4, 4, 6, 5, 8 respectively), where all the issue values are of a nominal scale, e.g. the issue *Amusement* has the following values: Nightlife and entertainment, Nightclubs, Excursion, Casinos, Zoo, Festivals and Amusement park. However it could easily include issues which use real values for example the price of the vacation, or issues which use integer values, like the duration of the vacation (in days).

The negotiation *outcome space* is denoted as Ω, which represents all the possible outcomes achievable i.e. all possible issue value combinations within the domain. For example the Travel domain has 188160 possible outcomes ($7 \cdot 7 \cdot 4 \cdot 4 \cdot 6 \cdot 5 \cdot 8 = 188160$), out of which any can be offered to the opponent. The formal definition of a negotiation outcome (ω), also referred to as bid or offer, with n issues can be seen in equation 1.1

$$\omega = \{\omega_1, ..., \omega_n\} \tag{1.1}$$

where ω_i denotes a value associated with the i^{th} issue.

Each negotiator defines its preferences (preference profile). For each issue in the domain a weight is assigned, indicating how important that particular issue is to the negotiator (*issue weight*). For example, in the travel domain Jack could place more importance on amusement compared to shopping, which results in a higher issue weight. Using the preference profile of a negotiator it is possible to map any offer in the domain to a value (*Utility*), a rating indicating how well a bid satisfies the negotiator's preferences. The utility of a

multi-issue outcome is calculated by means of a *linear additive function* that evaluates each issue separately. The equation to calculate this can be found below:

$$U(\omega) = \sum_{i=1}^{n} w_i \cdot e_i(\omega_i) \tag{1.2}$$

where w_i are the normalized issue weights and $e_i(\omega_i)$ the evaluation functions for the i^{th} issue.

Other factors that are connected to the scenario are time pressures like *Deadlines* and *Discount Factor*. A deadline determines what the duration of the negotiation is. Once the deadline has passed, it means that the negotiation has ended [112]. If no agreement has been made both agents well receive their *Reservation Value* a predefined value which an agent will receive should no agreement be made. A discount factor specifies the rate the utility decreases over time. In other words, as the negotiation progresses the utility gained by an agreement will be reduced [86]. An example of this is when one is negotiating over perishable goods, as the duration of the negotiations decreases the value of the goods.

Figure 1.2 provides a visual representation of the negotiation, showing the components in the negotiation environment and how they interact with each other and the agents.

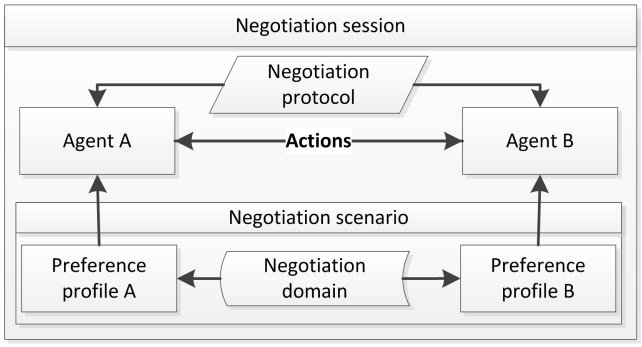

Figure 1.2: A diagram of the negotiation environment

By using the two preference profiles in conjunction with the domain we are able to analyze the scenario, presenting a visual representation of the domain with respect to the two preference profiles. Figure 1.3 shows such a visual representation for the domain Itex vs Cypress [70], a domain that contains 180 possible outcomes.

From figure 1.3 we are able to observe certain characteristics about the scenario. For example, how large the domain is, whether the bids are spread out over the domain or clustered together. Probably the most useful characteristics is the Pareto Frontier; its shape where it is located. The Pareto Frontier consists of Pareto optimal bids. These are bids where it is impossible to find a better bid which will benefit one individual without harming the other one's performance [124]. Pareto optimal or Pareto efficiency is also used as a measure for the joint performance of an outcome. Another important bid which is used to

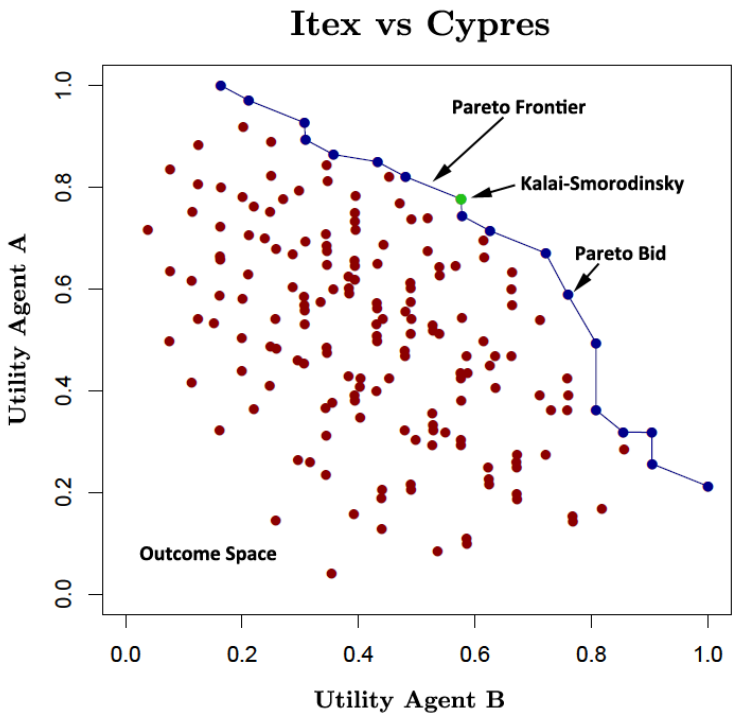

Figure 1.3: Visual representation of the Travel Domain

measure fairness is the Equal Proportion of Potential Point (EPP) also known as the Kalai-Smorodinski Point or Kalai point. The Kalai point is where the difference between both party's utilities are minimal.

Challenges in Automated Negotiation

For an agent to be able to perform well it needs to know what the preferences are of the user. In an automated negotiation environment these preferences need to be expressed in quantitative form (e.g. with issue weights) in order to determine the utility. Eliciting the user's preference can pose a challenge; depending on the domain there are several issues that need to be taken into account and as the issues increase, the number of alternative outcomes increases exponentially. For humans it is unnatural to express our preferences in a quantitative manner, making it challenging to present them accurately.

Once the agent has been able to obtain the users preferences there is still the question of what negotiation strategy to employ; how to choose counter bids to offer or whether it should accept the opponent's bid. There has already been a lot of research done in finding effective negotiation strategies [6, 12, 13, 28, 30, 111, 118] with two competitions to support this. The Trading Agent Competition [92], where agents bid for resources in an auction type setting and the Automated Negotiating Agent Competition (ANAC) [6, 13], which supports

bilateral negotiation, where two agents negotiate against each other. Despite these challenges automated negotiation is still quite promising. NSS has been shown to be a useful teaching and training tool [67, 79] and provides invaluable support to the negotiators, aiding them in the preparation as well as during the negotiation. With new advancements in computing power, automated negotiators are becoming more efficient and better at dealing with the complexities of a negotiation, allowing for better outcomes. This work will primarily focus on negotiation strategies, aiding in the creation of better strategies.

1.3 Contributions to Research

What our work contributes to the field of negotiation is the possibility to investigate the components of a negotiation strategy and examine their relationships. This work developed an architecture which allowed us to physically separate components from a strategy and analyze them individually.

The reason we developed this architecture was because in the field of automated negotiations, researchers are constantly trying to improve negotiation strategies, so that eventually they will be able to find the optimal one. Currently there are methods for determining the best negotiation strategy given a set of agents and scenarios [6, 13], however these are only valid for that specific set; given a different set of opponents or scenarios, the negotiation strategy can be less effective. These evaluation methods also look at the negotiation strategy as a whole, which makes it very hard to pinpoint what works well within a negotiation strategy and what does not. To be able to find an effective negotiation strategy researchers need to take a closer look at a strategy, studying its components in order to pinpoint what works well, as it is considered that well-performing components together constitute a well-performing agent. To date no efficient method exists to identify to which component the success of a negotiation strategy can be attributed and finding such a method would allow for the development of better negotiation strategies, resulting in better agreements.

To be able to create this architecture and to study a negotiation strategy's components, we first needed to find similar components within a negotiation strategy. With many negotiation strategies ranging from simple to very complex behavior, the question is: *whether it is possible to identify generic components within most, if not all, negotiation strategies*.

Studying the literature and existing negotiation strategies, three distinct components were identified: the ***B**idding Strategy*; which determines the concession and counter offers the agent makes, the ***O**pponent Model*; which estimates the opponent's preferences in order to offer more enticing bids, and the ***A**cceptance Condition*; which determines if the bid offered by the opponent is acceptable or not. These three *BOA Components* constitute the components of the *BOA Architecture*. A negotiation strategy is a result of complex interactions between the BOA components, of which the individual performances may vary significantly. By analyzing these components separately, it is possible to determine what works well in a negotiation strategy and what does not [8].

In this thesis, the term *Strategy* is used to describe the complete negotiation strategy, which consists of three *BOA Components* (Bidding Strategy, Opponent Model and Accep-

tance Condition). As each BOA component consists of a collection of *Components*, we make a distinction between a *BOA component* and a *Component*, where the former is abstract, describing the behavior of a component, while the latter is concrete, determining the behavior of the component and can be swapped for another.

The advantages of using the BOA architecture are threefold: first, it allows us to study the behavior and performance of the BOA components; second, it allows us to systematically explore the space of possible negotiation strategies; third, it aids the development of negotiation strategies by allowing researchers to focus on specific behaviors of a negotiation agent.

During the creation of the BOA architecture we have developed a large repository of components which permitted us to mix and match, swapping them at will. This way it was possible to systematically explore the vast space of negotiation strategies, comparing the performance of different component combinations. By doing this we were able to answer the following questions:

- If certain components perform better than others (with respect to some performance measure), does replacing a less effective component for a better one improve the agent's performance?
- Does combining the best (optimal) components in one strategy result in an optimal negotiation strategy?

To answer the first question we compared the performance of different component combinations, which was done in two ways. One approach was to determine the performance of a component by classifying the different components into three performance groups per BOA component. This allowed us to make and compare different strategy combinations based on the performance of the components. The other approach was to compare the performances of only two components. Both these approaches help to determine if swapping a component for a more effective one would improve the performance of the strategy and whether combining the best components would result in the best strategy. Knowing if combining components in a certain way helps improve negotiation strategy, we gain insight and a better understanding of how we can improve negotiation strategies. We also provided a systematic way of doing this, swapping less effective components with better performing ones.

The performance of an agent is very dependent on the component it employs, so changing these will have some impact, but it is unknown how much impact it will have. Does altering BOA component x have the same impact as doing so for BOA component y? This leads to the question *which of the three BOA components is the most important and contributes the most to the negotiation strategy?* In this work we looked at this from an engineering perspective: when designing an agent, to what BOA component should the engineering effort be directed to. We started by making this question precise, formulating it in quantifiable terms of predictability of variance. To determine the contribution of each BOA component we used the statistical measure of effect size. We performed an extensive analysis on a wide

array of components to determine which of the BOA components contributed the most variance to the overall performance. Knowing which BOA component contributed the most allows for a better understanding of what matters most in a negotiating agent design and agent designers can spend their development time more efficiently, allocating time on BOA components which produce the most gain.

The remainder of this thesis is as follows. Chapter 2 introduces related work that has already been done within the field of automated negotiation. Chapter 3 presents the BOA architecture [8], the aim of this chapter is to present the ideas behind the BOA architecture as well as taking a detailed look at its design and its implementation. Furthermore we provide empirical proofs that many state-of-the-art negotiation strategies can be converted into a BOA agent. Chapter 4 goes more in depth into the different ways to apply the BOA architecture, illustrating the advantages of the BOA architecture like improving state-of-the-art, building new agents, scaling and systematically exploring the negotiation space. Chapter 5 looks at the dynamics between the different BOA components and will answer key questions like: whether combining the best components will lead to the best negotiation strategy and which of the three BOA components is the most important to the net performance of a negotiation agent. We also discuss the interactions that occur between the different BOA components, which can influence the final outcome. Finally chapter 6 will present our conclusions and recommend future work.

Chapter 2

Related Work

This work introduces an architecture based on a theory of components as well as methods to answer complex questions about the components' performance, behavior and interaction. We have surveyed literature that investigates and evaluates such components. There are four categories of research approaches which are related to our work:

1. **Architecture of negotiation strategy**. Literature detailing the architecture of a negotiation strategy of an agent.

2. **Evaluation of negotiation strategy**. Work that investigates, discusses and compares the performance of a component of a negotiation strategy

3. **Negotiation strategy space exploration**. Literature that explores and combines a set of negotiation strategies to find an optimal strategy.

4. **Finding important components**. Methods that find the important variables/components within data.

2.1 Architecture of Negotiation Strategies

There are many different types of negotiations and for each type, there are specific architectures to support them. This is because each type of negotiation requires particular behaviors. For example, there are a couple of agent architectures for argumentation-based negotiation [42, 100, 104], trust negotiation [24] and auctions [102].

The primary scope of this work however is the focus of generic components of a negotiation strategy in a standard bilateral negotiation setting. To our knowledge, there is little work in literature describing, at a similar level of detail as our work, the generic components of a negotiation strategy architecture. For example, Bartolini et al. [16] and Dumas et al. [31] treat the negotiation strategy as a singular component. There are however some notable exceptions, a few papers recognize that there are different parts to a negotiation strategy, they distinguish two distinct strategy behaviors; the responding strategy (similar to the acceptance condition in our architecture) where the agent responds to a bid offered

by the opponent and the proposing strategy (similar to the bidding strategy in our architecture) where the agent offers a bid to the opponent [15, 60, 88]. Jennings et al. [60] go a step further, pointing out that an agent is selecting bids in the hope that it will eventually stumble upon a bid which is acceptable for both agents, hence showing the need for an extra component (an opponent model).

Lia et al. [76] have a similar architecture compared to what we propose with the BOA architecture. Lia et al. distinguish between three components: *Conceding*, *Proposing* and *Responding*. The first component is used to decide how much it will concede, in other words it determines what the threshold or target utility should be (to some extent this follows the line of a bidding strategy). The second component chooses which bid it will offer to the opponent, it is possible that there are multiple outcomes which satisfy the target utility and thus one of them will need to be chosen. There are multiple ways of doing this, as suggested in the paper there are methods like: presenting a few if not all of the bids, choose the bid with the shortest distance to the best bid the opponent made. Another option is to try and estimate what the preference of the opponent is and see which of the bids out of the list is most favorable from the view of the opponent (which is the function of our opponent model component). The final component (which resembles the acceptance condition within the BOA architecture) decides whether the agents should accept or reject the bid of the opponent.

Jonker et al. [62, 63] present an agent architecture for multi-attribute negotiation, where each component represents a specific process within the behavior of the agent, e.g.: attribute evaluation, bid utility determination, utility planning, and attribute planning. In contrast to our work, Jonker et al. focus on tactics for finding a counter offer and does not discuss acceptance conditions. Our architecture also considers a bid as a whole, so components like attribute evaluation and bid utility determination (which are present in the architecture by Jonker et al.) are not necessary. Figure 2.1 shows the internal components of the architecture. There are however some similarities between the two architectures. For example, the utility planning and attribute planning components correspond to the bidding strategy component in our architecture.

Ashri et al. [3] introduce a general architecture for negotiation agents, discussing components that resemble our architecture; however, the negotiation strategy is described from a *Belief, Desire and Intention* (BDI) agent perspective (in terms of motivation and mental attitudes). Components such as a proposal evaluator and response generator resemble an acceptance condition and bidding strategy respectively; however, the difference is that these components can be influenced by certain factors like mental models (motivation and attitudes), which is not part of our architecture. Figure 2.2 shows the agent architecture.

Hindriks et al. [51] introduce a generic architecture for negotiation agents in combination with a negotiation system architecture. Parts of the agent architecture correspond to the architecture presented in this paper; however, their focus is primarily on how the agent architecture can be integrated into a larger system architecture. In addition, Hindriks et al. treat the acceptance condition and bidding strategy as a singular component.

Many of these agent architectures recognize that on a higher level there are at least two distinct behaviors: one where the agent determines if the bid offered by the opponent

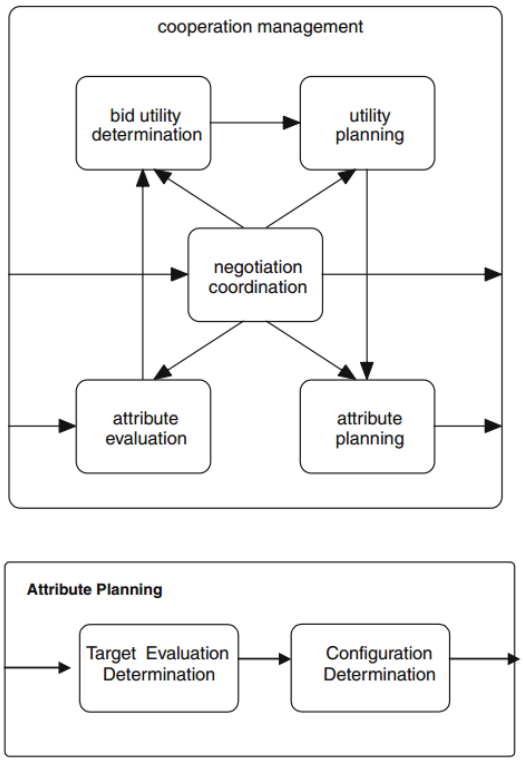

Figure 2.1: Internal composition of the Cooperation Management component. Taken from [63].

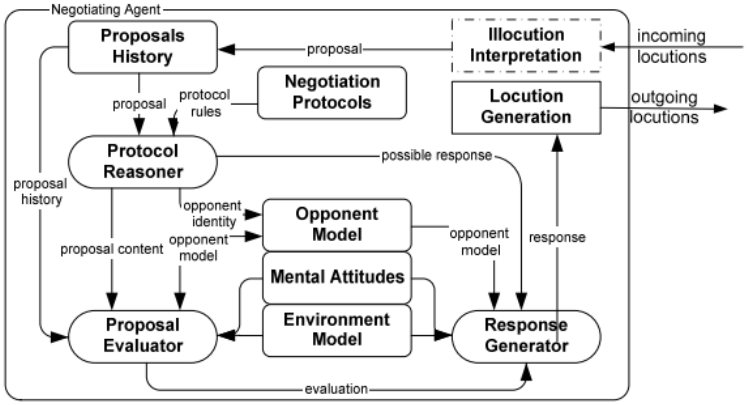

Figure 2.2: Ashri et al. Agent Architecture. Taken from [3].

is acceptable (based on a certain criteria) and the other behavior is to determine, if the opponent bid is rejected, what bid should be offered in return (also based on certain criteria).

It is in the manner of how these agent architectures determine and select these criteria that make them different, i.e. some use BDI, and others look at the utility threshold or the issues instead of considering a bid as a whole. There are also some architectures that implement or recognize other components to improve the chance of achieving a better outcome e.g. an opponent model.

2.2 Evaluation of Negotiation Strategies

To be able to determine if a negotiation strategy is effective or not, there has to be a way to evaluate them. There are different ways one goes about doing this; if one is considering a single negotiation session one can evaluate a negotiation strategy based on the *Intrinsic Agent Utility*, which is the final utility obtained at the end of a negotiation session. The intrinsic agent utility is independent of the time or the resources consume. Should no agreement be achieved by the end of the negotiation then the reservation value is assigned to the agents respectively [36]. To include the time and resource penalties, *Cost Adjusted Benefit* can be used, where the costs are subtracted from the intrinsic utility [36]. A similar method used by Henderson et al. [46] is to give each negotiator a set of goals (outcomes) which are known as examples. To evaluate the performance of the strategy, they compared the distance between the accepted bid with the set of examples. They score a strategy based on the inverse of the distance ($\frac{1}{distance}$) which results in a value between 0 and 1. The higher the score the closer the accepted bid is to the example and thus the better the agent performed. One can also evaluate a strategy by comparing the accepted bid with certain quality measures for example, how close it is to the Pareto Frontier [105, 125], the Nash Product [23, 91, 122], or the Kalai-Smorodinsky Point [50, 89].

These types of evaluation methods are based on one negotiation session which can produce anomalies. To reduce this and ensure a more representative spread of results, one can use a tournament approach, where the agent negotiates against multiple agents and in different scenarios. One such evaluation method is *Averaged Utility*. This method takes the score of the different sessions and averages them [36]. In ANAC [6, 13] they also re-normalize the utility using the maximum and minimum utility achieved by all other agents for that profile, this gives a normalized domain score [13].

With the BOA architecture we were able to now distinguish between the different components and thus evaluate them individually. There has already been some research focusing on specific components, for example the development of different bidding strategies, ranging from time-dependent (where the bidding behavior is influenced by time) to behavior-dependent tactics (where the bidding behavior is influenced by the opponents behavior), from simple to complex strategies. Papers by Faratin et al. [36], Sierra et al. [108] and Matos et al. [89] present an overview of these different types of tactic and Faratin et al. [36] go further by analyzing the performance of pure negotiation tactics on single issue domains in a bilateral negotiation setting. To date however, there is little research on how to evaluate a bidding strategy. A typical way to assess the performance of a bidding strategy is to look at aspects like, the fairness and quality of the outcome, who "won" the negotiation (received

the best deal), the distance to Nash or Kalai-Smorodinsky. The paper by Hindriks et al. [50] presents a metric to analyze the negotiation dynamics of a bidding strategy. They classify the type of moves an agent makes based on the location of the current bid relative to the location of the previous bid within a utility plot. In total there are six different types of steps an agent can make *Selfish, Fortunate, Silent, Nice, Concession or Unfortunate*, which can be seen in figure 2.3. With these move classifications, it is possible to create evaluation criteria to evaluate a bidding strategy.

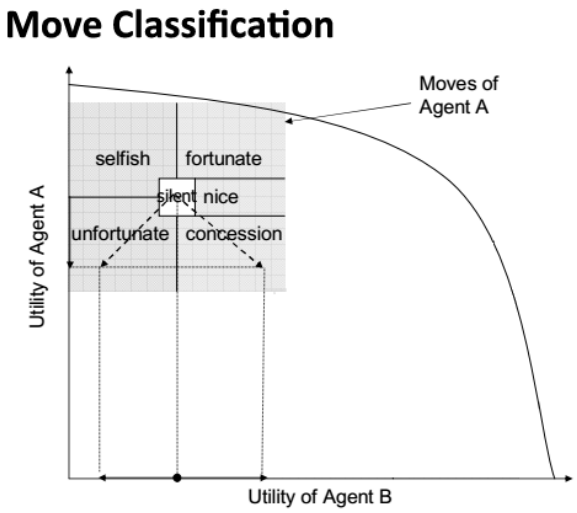

Figure 2.3: Classification of Agent Moves. Taken from [48]

There has been some research in the area of opponent modeling, it is believed that with an opponent model the quality and efficiency of the negotiation process will be enhanced [53, 134]. With an opponent model the negotiation agent can offer bids that are not only of higher utility for itself but also for the opponent, which results in a higher chance of acceptance. There are different types of opponent models in the literature; including opponent models that estimate the reservation value [134] the deadline [56], the (partial) preference profile [53], the opponent's chance of accepting an offer [99], or opponent models that predict the opponent's next move [22].

Our work focused on opponent models which estimate the (partial) preference profile, because most existing implementations fit in this category. The BOA architecture provided a way to determine and compare the performance of different opponent models by separating the implementation of the opponent model from the rest of the negotiation agent.

To evaluate an opponent model, Hindriks et al. [54] have introduced different quality measures for learning, based on the estimated and the actual preference profile. Work by Baarslag et al. [7] place these quality measures into practice as well as comparing the state-of-the-art opponent models.

The acceptance condition has received little attention to date, however work by Baarslag et al. [10] analyzed the performance of a set of acceptance mechanisms. These acceptance

mechanisms depended on parameters such as the time, the utility of the previous or the next bid and the utility thresholds. This has given insights in how acceptance conditions work and which perform well. Baarslag et al. have continued their research in looking for the optimal stopping technique, developing an acceptance condition $AC_{OptimalStopping}$ [14], which predicts the opponent's strategy using a Gaussian process regression technique and determines the odds if a better bid will be offered in the future. They have shown favorable results, however to date there are not evaluation measures to determine if an acceptance condition performs well or not.

2.3 Negotiation Strategy Space Exploration

This work looks at how combining different components effects the net performance of a negotiation strategy. Within literature there are various authors who have done similar research in the form of combining strategies in the hope of finding an optimal negotiation strategy.

Faratin et al. [36] analyze the performance of "pure" negotiation tactics, considering them as a component around which full strategies are built. Faratin et al. further discuss the possibility of linearly combining these pure tactics, but they do not investigate this any further.

A paper by Sierra et al. [108] proposes a negotiation strategy which combines a concession based strategy (either time based or behavior based [36]) and a trade-off strategy [35]. The former tries to achieve an agreement, decreasing the utility threshold, while the latter searches for a satisfactory proposal, maintaining its own utility as much as possible. Sierra et al. combined them so that they complement each other; the agent follows the trade-off strategy until it perceives a deadlock, upon which it makes a rational concession based on its concession strategy and then continues with the trade-off strategy. From their experiments Sierra et al. saw that their strategy obtained better results than other combinations, e.g. selecting a strategy to execute during each turn at random.

Another negotiation strategy, proposed by Ilany and Gal approach this differently [57]; instead of combining different strategies during one negotiation session, they select the best strategy from a predefined set of agents, their choice is based on the characteristics of the domain. This is done through machine learning, whereby the agent is optimized to play well on different domains. In their experiments they have shown that their strategy significantly outperforms the tournament winner.

This work, combining existing strategies into one, is similar to our approach. However, we combined the different generic components of existing strategies: the bidding strategies, the opponent models and the acceptance conditions, instead of just the bidding behavior or strategies as a whole, as done by Gal and Ilany.

Rather than simply combining certain tactics or strategies, some authors chose a more complex approach, using genetic algorithm (GA). In the literature there are different ways to use GA in a negotiation strategy: one can use it to generate proposals at every round [77]

or track shifting tactics and changing behaviors [75]. However, the one most related to our work is using GA to evolve strategies to improve them.

Matos et al. [89] use a set of baseline negotiation strategies which consists of time dependent, resource dependent, and behavior dependent strategies [36]. The negotiation strategies are combined linearly and encoded as a type of chromosomes after which they are utilized by a GA. This approach is limited to the acceptance criteria that specify a utility interval of acceptable values, and hence does not take time into account; furthermore, the agents do not employ explicit opponent modeling. In our work we considered all generic components.

Eymann [32] also uses GA with more complex negotiating strategies, evolving six parameters that influence the bidding strategy. The GA uses the current negotiation strategy of the agent and the opponent strategy (with the highest average utility) to create a new strategy, which is similar to other GA approaches (see Beam and Segev [17] for a discussion of the application of GA in automated negotiations). The GA approach mainly treats the negotiation strategy optimization as a search problem in which the parameters of a small set of strategies are varied using GA. This is different to our approach as we did not tune parameters.

Oliver [95, 97] uses GA to improve self-interested negotiations strategies by applying simple sequential threshold rules, which will only accept an offer if it is greater than a threshold t_1 or else a counter bid will be offered. Should the opponent not accept and offer its own counter bid, the agent will then accept if it is higher than t_2 and so forth. His results show that after repeated training runs, the agents were able to learn more effective methods of negotiation, sometimes exceeding the performance of a human negotiator.

A paper by Tu et al. [126] have pointed out a shortcoming in Oliver's work. Using simple sequential threshold rules does not allow for past actions to be taken into account. Tu et al. propose using *Finite State Machines* (FSM) to model the strategy where there is a possibility to simulate "memory" of previous actions. Like Oliver's work each of these FSM represent strategies which are modified over time using GA. Their results show that this approach is promising and that FSM performs better than random strategies and that they are capable of reaching good negotiation results.

The GA approach combines certain strategies or parameters and tune them over time. Our approach differs in that there is no offline learning or alteration of the strategies over time required. Also our work combined different generic components instead of complete strategies.

2.3.1 Finding Important Components

In our work we determined the importance of a generic component based on its contribution on the negotiating agent's performance. However there are other methods for finding important generic components. *Principle Component Analysis* (PCA) is such a method; PCA is a multivariate technique to extract important information from observed data and expresses this as a new set of orthogonal variables called *principal components*, which are less in number than the original variables [1]. It is probably the most popular multivariate statistical technique, used in almost all scientific disciplines and applications like data com-

pression, image processing, exploratory data analysis and pattern recognition [123]. These principal components are a linear combination of the original variables, where the first principal component is required to have the largest possible variance. The second principal component has the constraint of having the second largest variance. This shows that the first principal component is the most important, followed by the second and the third, etc.

Another method is *Factor Analysis*, which is similar to PCA, having the same goals. Its underlying hypothesis is that a set of variables can be described by another set of variables, smaller in numbers than the original set. By extracting important information the data can be reduced into new variables, *factors* [61]. The difference between PCA and factor analysis is that the latter attempts to reduce the number of dimensions by using regression modeling techniques, while the former does not use any explicit model [113].

Both these methods have a primary goal of reducing the amount of variables by combining them linearly. In our research though, we have already defined our variables (BOA components) and are not interested in reducing the variable dimensions. Also, the aforementioned methods are able to determine which of the combined components are important, but they are unable to determine the importance of the original components, which is actually a different question. Another reason we were unable to use these methods is because our components were derived from negotiation strategies. When looking at a strategy (a single data point) the components were associated with the utility obtained by that strategy. For example, agent A received an average utility of x, this means that the bidding strategy, opponent model and the acceptance condition for that strategy will have the same value, x. If we were to plot our data points we would end up with a single line ($x = y = z$), where performing a PCA or factor analysis would result in a single principal component or factor, making the analysis trivial.

Chapter 3

BOA Architecture

There are methods to determine which is the best negotiation agent given a particular set of opponents and scenarios [6, 13], however it is still not known which is the best agent in general. In these evaluation methods the negotiation strategy is considered to be one entity, making it very hard to pinpoint what works well in a negotiating agent and what does not. For example; an agent can employ an effective method to estimate the opponent's preferences but still perform mediocre in a negotiation because, the bid that the agent accepts is sub-optimal. To get a better understanding we need to look at a negotiation strategy more closely, examining the components which constitute a negotiation strategy. The question is: *whether it is possible to identify generic components within most, if not all, negotiation strategies?*

With the International Automated Negotiating Agents Competition (ANAC) [6, 13] and the growing interest in automated negotiation, the search for an effective strategy has created a surge of newly developed strategies and tactics, ranging from simple to very sophisticated ones. From these strategies it is possible to distinguish different types of behavior, which we consider to be the basic components of a negotiation strategy [8]. The basic components are the ***B**idding Strategy*, the ***O**pponent Model* and the ***A**cceptance Condition*. These three *BOA Components* constitute the *BOA Architecture*, where a negotiation strategy is the result of complex interactions between these BOA components.

To date, no efficient method exists to identify the components to which the success of a negotiating agent can be attributed. Finding such a method would allow the development of better negotiation strategies, resulting in better agreements; the premise being that well-performing components together will constitute a well-performing agent.

To tackle this problem, we analyzed the BOA components separately, showing that most of the currently existing negotiating agents can be fitted into the *BOA architecture*. We further validated the BOA architecture by re-implementing, among others, the ANAC agents to fit into our architecture and showed that the BOA agents are equivalent to their original counterparts.

The advantages of fitting agents into the BOA architecture are threefold: first, it allows the study of the behavior and performance of the BOA components, for example, by re-implementing the ANAC agents in the BOA architecture it becomes possible to compare the accuracy of all ANAC opponent models and to pinpoint the best opponent model from

them [7]. Second, by only altering one of the components at a time, it allows the systematic exploration of the space of possible negotiation strategies and third, it aids the development of negotiation strategies by allowing researchers to focus on specific behaviors of a negotiation agent.

Section 3.1 goes into depth of the design of the BOA architecture [8] and describes the different behaviors and how combing them results in a negotiation strategy. Section 3.2 looks at the actual implementation of the BOA architecture; how it is built upon GENIUS (a well established negotiation platform) and how one can create and use a BOA agent or different components in a negotiation tournament. Finally section 3.3 answers the aforementioned question and shows that most of the existing negotiating agents can be decoupled to comply with the *BOA architecture*. This is done by re-implementing the ANAC agents as BOA agents, and show that the newly developed BOA agents are equivalent in behavior and performance to their original counterparts.

3.1 Design

When designing the BOA architecture we wanted to implement it in such a manner that it would support the decoupling of existing agents, while preserving its original behavior and performance. The BOA architecture also would have to provide support for the interchangeability of components, e.g. swapping the acceptance condition of one agent with the acceptance condition of another. There are two distinct categories within the BOA architecture: elements that are part of the agent's environment, and components that are part of the agent itself.

3.1.1 Negotiation Environment

We employed the same *negotiation environment* as in [6, 13]; that is, we considered bilateral automated negotiations, where the interaction between the two negotiating parties was regulated by the alternating-offers protocol [110]. The agents negotiated over a set of issues, as defined by the negotiation *domain*. The negotiation happened in real time where a deadline was present. Should no agreement be made before the deadline, both agents received nothing. Discount factors were also present, whereby the utility of an agreement may decrease over time.

In addition to the domain, both parties also had privately-known preferences described by their *preference profile*. While the domain is common knowledge, the preference profile of each player is private information, this means that each negotiator was unaware of the opponent's preferences. The negotiator could attempt to learn this during the negotiation encounter by analyzing the *bidding history*, using opponent modeling techniques.

3.1.2 The BOA Agent

Based on a survey of literature and the implementations of currently existing negotiation agents, we identified three main components of a general negotiation strategy: a *bidding*

strategy, an *opponent model*, and an *acceptance condition* (BOA components). The elements of a BOA agent are visualized in Figure 3.2. In order to convert an agent into the BOA agent it should be possible to distinguish these BOA components within the agent design, with no dependencies between them. An exposition of the agents we considered is given in the next section, which will further motivate the choices made below.

1. **Bidding strategy (BS)**. A bidding strategy is a mapping from a negotiation trace to a bid. The bidding strategy determines appropriate concessions to be made, depending on factors such as the opponent's negotiation trace, a target threshold, time, discount factor, etc. To determine if a bid to be offered is appropriate, the bidding strategy can consult with an opponent model, passing one or multiple bids and comparing them within the estimated opponent's utility space.
   Input: *opponent utility of bids, negotiation trace*.
   Output: *provisional upcoming bid.*

2. **Opponent model (OM)**. An opponent model uses learning techniques to construct a model of the opponent's preference profile, which allows the opponent model to estimate the opponent's utility of an arbitrary bid.
   Input: *set of possible bids, negotiation trace, outcome space or negotiation domain*.
   Output: *estimated opponent utility of a set of bids*.

3. **Acceptance condition (AC)**. The acceptance condition determines whether the bid offered by the opponent is acceptable, depending on certain factors, which are similar to those used by the bidding strategy.
   Input: *provisional upcoming bid, negotiation trace*.
   Output: *send accept, or send out the upcoming bid.*

To better understand how the different BOA components work, let us view a negotiation process as a search problem, where the negotiation strategy is a filter to find an outcome which both parties are willing to agree upon. The BS controls the rate of concession and the target utility (range), which determines the general location of the agreement in the outcome space according to the agent's own utility. This concept can be seen in Figure 3.1, where the target utility range (B) is shown in the outcome space. The AC is able to extend this area of possible acceptable bids, depending on whether it is prepared to make a large jump or concession towards the opponent in order to reach an agreement. Should the AC concede a lot to reach an agreement, the number of acceptable outcomes increase. This is illustrated in Figure 3.1, showing the extended area and the acceptable outcomes (A). The OM can filter this area, refining the possible agreement to bids that are near the Pareto frontier. This is because OM's that perform well generally offer bids that are close to the Pareto frontier. This idea is presented in Figure 3.1, where the target utility ranged is filtered by the OM (O).

To illustrate the BOA components within a strategy, we use IAMhaggler2011 [133] as an example, which finished third in the ANAC 2011 competition [6]. In order to choose a counter bid to offer to the opponent, IAMhaggler2011 needs to have a target utility, so that it can filter and offer bids with similar utility. This is the job of the BS, where it chooses a

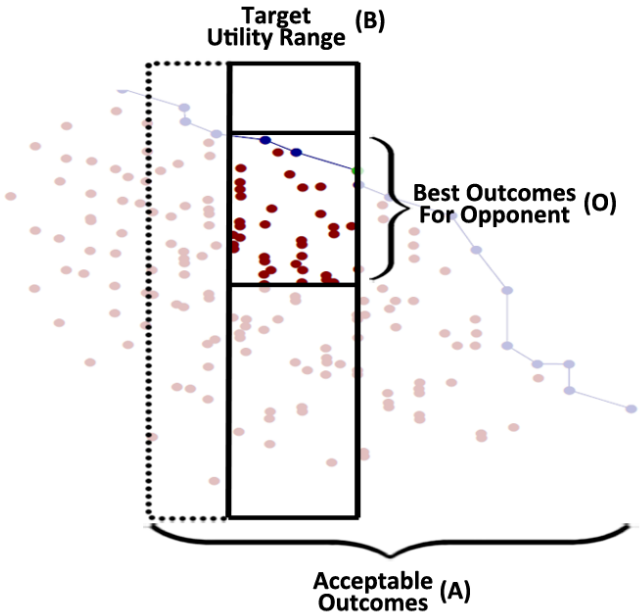

Figure 3.1: Outcomes filtered by opponent model

target utility based on the rate at which it will concede, i.e. the concession rate. The way IAMhaggler2011 determines its concession rate is by first predicting the concession rate of the opponent, with this information it will be able to sets its own concession rate so that the expected utility is optimized. IAMHaggler2011 uses an OM in order to approximate the opponent's concession rate by taking the past offered bids and inspecting them within the estimated opponent's utility space in order to determine the opponent's concession rate. The AC of IAMhaggler2011 uses a combination of acceptance mechanisms defined in [11], for example; $AC_{constant}$ which will only accept bids that have a utility higher than a certain value.

The BOA components interact in the following way (the full process is visualized in Figure 3.2): when receiving the opponent's bid, the BOA agent first updates the *bidding history* and *opponent model* to make sure that up-to-date data are being used, which maximizes the information known about the environment and opponent. Given the opponent bid, the BS determines the counter offer by first generating a set of bids which satisfies certain criteria. The BS uses the OM (if present) to select a bid from this set by taking the opponent's utility into account. Finally, the AC decides whether the opponent's offer should be accepted. If the opponent's bid is not accepted by the AC, then the bid generated by the BS is offered instead.

At first glance, it may seem counter-intuitive to make this decision *at the end* of the agent's deliberation cycle. Clearly, deciding upon acceptance *at the beginning* would have the advantage of not wasting resources on generating an offer that might never be sent out.

However, generating an offer first allows us to employ AC's that depend on the utility of the counter bid that is ready to be offered. This method is widely used in existing agents [10]. Such acceptance mechanisms can make a more informed decision by waiting until the last step, when more information is available. Given our aim to incorporate as many agent designs as possible, we therefore adopted this approach in our architecture.

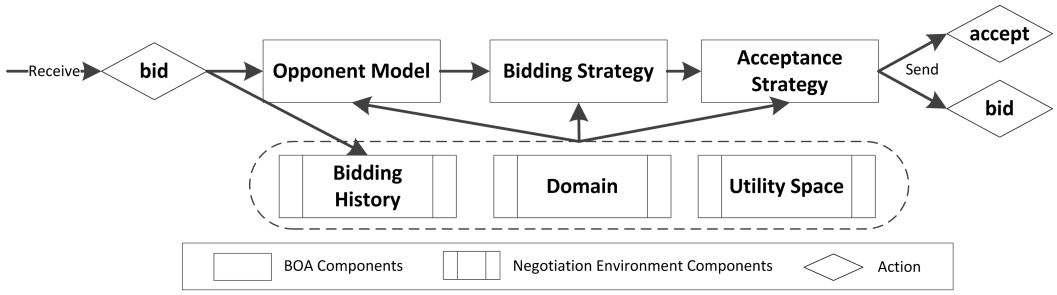

Figure 3.2: The BOA architecture negotiation flow.

3.2 Implementation

We have implemented the BOA architecture as an extension of the GENIUS framework [49, 85], which stands for **G**eneric **E**nvironment for **N**egotiation with **I**ntelligent multi-purpose **U**sage **S**imulation, and is a negotiation platform that implements an open architecture supporting heterogeneous agent negotiations. This framework was developed as a research tool to facilitate the design of negotiation strategies and to aid in their evaluation. It provides a flexible and easy to use environment for implementing negotiation agents as well as running negotiations. GENIUS allows the agent designer to focus on the negotiation strategies, by providing a structure to manage the negotiation process according to the rules of the negotiation protocol and include functionality whereby the agents are able to access information about the negotiation scenario. GENIUS can further aid the development of a negotiation agent by acting as an analytical toolbox, providing a variety of tools to analyze the negotiation agent's performance, which is based on the outcome and dynamics of the negotiation. The advantage of GENIUS is that it presents the negotiation results in such a manner that it becomes easy to compare results between different setups, which is much harder to do between ad hoc negotiation platforms. It also allows researchers to easily run simulations (see figure 3.3). All a researcher needs to do is specify the characteristics of the tournament or single negotiation session, for example the domains the agents need to negotiate on and what the two agent pools are.

The BOA architecture has been integrated seamlessly into the GENIUS framework, offering the user the ability to create and apply newly developed components using a graphical user interface, as depicted in Figure 3.6. From the perspective of GENIUS, a negotiation agent is identical to a BOA agent, and therefore both types of agents can participate in the same tournament.

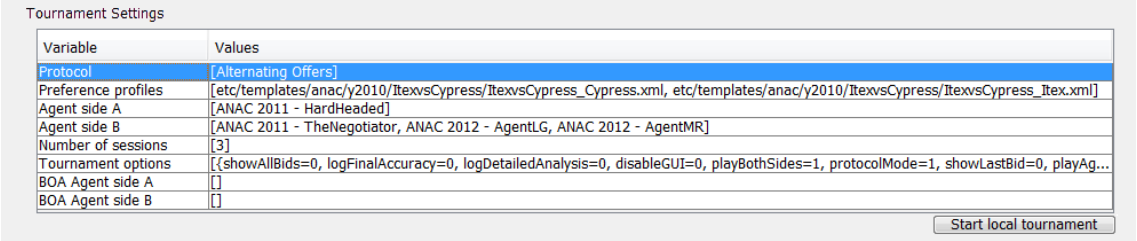

Figure 3.3: GENIUS interface to run a tournament or single negotiation session.

3.2.1 Main BOA classes

The following sections briefly describe the main classes of the BOA architecture, which consists of the *NegotiationSession, BOAagent, Bidding Strategy, Opponent Model*, and *Acceptance Condition*. For these classes we provide a reduced view of the implementation showing the important variables and methods only. For the classes pertaining to the BOA agent, we show how to construct these components in such a manner that they can be properly used by a BOA agent. Appendix A contains a class diagram depicting the main classes in the BOA architecture and the supporting classes used in a BOA agent.

Negotiation Session

The Negotiation Session, though not part of a BOA agent, plays a vital role in making sure that the agent has up to date information about the negotiation state. The Negotiation Session keeps track of the scenario the agent is negotiating in, i.e. the domain, the outcome space as well as the agents utility space. The Negotiation Session also keeps track of the negotiation itself, how much time has passed (via a timeline), as well as the negotiation trace of both agents (the bidding histories).

```
public class NegotiationSession {
   protected OutcomeSpace outcomeSpace;
   protected BidHistory opponentBidHistory;
   protected BidHistory ownBidHistory;
   protected Domain domain;
   protected UtilitySpace utilitySpace;
   protected Timeline timeline;
}
```

BOAagent

The BOAagent is the BOA version of the GENIUS's Agent class (which it extends). This is an abstract class containing a BS, an OM and an AC. It manages the incoming actions (via the receiveMessage method) and outgoing actions (via the chooseAction method) of

the BOA agent. It also controls the order in which the BOA components are run as well as the interaction with their environment (updating the NegotiationSession).

To use a BOA agent within GENIUS one needs to extend this class and specify the BS, the OM and the AC in the agentSetup method.

```
public abstract class BOAagent extends Agent {
   protected BiddingStrategy biddingStrategy;
   protected OpponentModel opponentModel;
   protected AcceptanceCondition acceptConditions;

   // Loads and initializes the BOA components of the agent.
   public void agentSetup() {
      ...
   }

   //Handles actions of the opponent
   @Override
   public void receiveMessage(Action opponentAction) {
      ...
   }

   //Handle the actions of the agent itself
   @Override
   public Action chooseAction() {
      ...
   }
}
```

Bidding Strategy

The Bidding strategy is the first BOA component, it is an abstract class which determines what counter bid to offer to the opponent. The BS contains three variables: nextBid, negotiationSession and an OM. The nextBid is used to store the bid that the BS is planning to offer to the opponent. Thereby the AC can take this into account when making decisions. The purpose of the negotiationSession is to keep the information about the negotiation state up to date. This allows the BS to make an informed decision. Once a set of bids have been selected, which satisfies certain criteria, the OM can choose the best bid with respect to the opponent's utility.

When creating a BS one needs to only extend the bidding strategy class and override some key methods. The two most important methods in the bidding strategy class are *determineOpeningBid* and *determineNextBid*. Both these methods determine what counter bid to offer to the opponent either, when it is offering the first bid or subsequent bids there after.

```
public abstract class BiddingStrategy {
   protected BidDetails nextBid;
```

```
    protected NegotiationSession negotiationSession;
    protected OpponentModel opponentModel;

    // Determines the first bid to be offered
    public abstract BidDetails determineOpeningBid();

    // Determines the bid to be offered after the first bid
    public abstract BidDetails determineNextBid();
}
```

Opponent Model

The opponent model is the second BOA component. It is an abstract class which models the opponent's preference profile. It makes use of a NegotiationSession and saves a UtilitySpace object, which is the estimated utility space of the opponent.

To create an OM one needs to extend the abstract opponent model class and override certain important methods. The three important methods are *updateModel, getBidEvaluation* and *getDiscountedBidEvaluation*. The *updateModel* method updates the OM with the newest bid offered by the opponent. The *getBidEvaluation* and the *getDiscountedBidEvaluation* method allow the agent to determine the utility (or discounted utility) of a particular bid with respect to the estimated opponent preference.

```
public abstract class OpponentModel {
    protected NegotiationSession negotiationSession;
    protected UtilitySpace opponentUtilitySpace;

    // Updates the opponent model
    public abstract void updateModel(Bid bid, double time);

    // Returns the estimated utility of the given bid
    public double getBidEvaluation(Bid bid){
        ...
    }

    // Returns the estimated discounted utility of the given bid
    public double getDiscountedBidEvaluation(Bid bid){
        ...
    }
}
```

Acceptance Condition

The acceptance condition is the third and final BOA component. It is an abstract class which determines if the bid offered by the opponent is acceptable or not. This class uses the NegotiationSession and the BS, to make sure it has up to date information. The nego-

tiationSession is used to get information about the negotiation state and the BS is used to determine the bid being offered to the opponent should it choose not to accept.

The AcceptanceCondition class needs to be extended if one wants to make an AC. One must also override the method *determineAcceptability*, which determines if the offer made by the opponent is acceptable or not based certain criteria.

```
public abstract class AcceptanceCondition {
   protected NegotiationSession negotiationSession;
   protected BiddingStrategy offeringStrategy;

   //Determines if opponent bid is acceptable
   public abstract Actions determineAcceptability();
```

3.2.2 Support Classes

There are three important support classes, the *OutcomeSpace*, the *BidDetails* and the *BidHistory*. The OutcomeSpace is a class which calculates all the possible issue-value combinations, and thus provides all the possible bids that can be offered. It also allows the agent to search through these outcomes based on a particular target utility. All bids are stored in a BidDetail; this is a container class which saves the bid with its undiscounted utility and the time that it was offered, allowing agents to determine the (discounted) utility at a later point in time. The last support class is a BidHistory which is a collection of BidDetails and allows for specific search functions through this collection of BidDetails.

3.2.3 Adding and Using Components in GENIUS

Adding a component to the GENIUS repository or using it in a BOA agent is a very easy process due to the simple user interface provided by GENIUS and the BOA architecture.

Adding a Component to the BOA Repository

To add a new component created by a user, the component needs to be added to the *BOA repository*. This can be done in two ways: by adding it directly to the BOA repository file (an xml file) or via the GENIUS interface, which can be done simply by opening and right clicking on the BOA component tab in the components window as shown in Figure 3.4 and select “Add new component”.

By selecting “Add new component” a window will appear as shown in Figure 3.5. In this window the user should specify the name of the component, for example “ANAC2012 - AgentLG”, specify the path to the class by selecting “Open” and optionally add parameters. By clicking “Add component” the component is added to the repository.

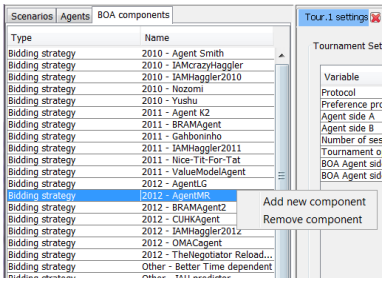

Figure 3.4: The BOA components window.

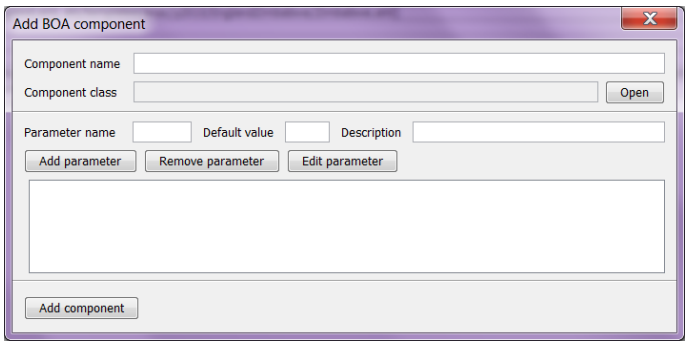

Figure 3.5: Loading a BOA agent.

Using Components

Using existing or newly created components (which have been added to the BOA repository) is very easy with the user interface of GENIUS. When setting up a negotiation tournament in GENIUS, the user is able to select a BOA agent by double clicking the *Values* section next to the *BOA Agent side A* or *BOA Agent side B*. This will bring up the BOA user interface (see Figure 3.6), where the user is able to construct multiple BOA agents by choosing its components.

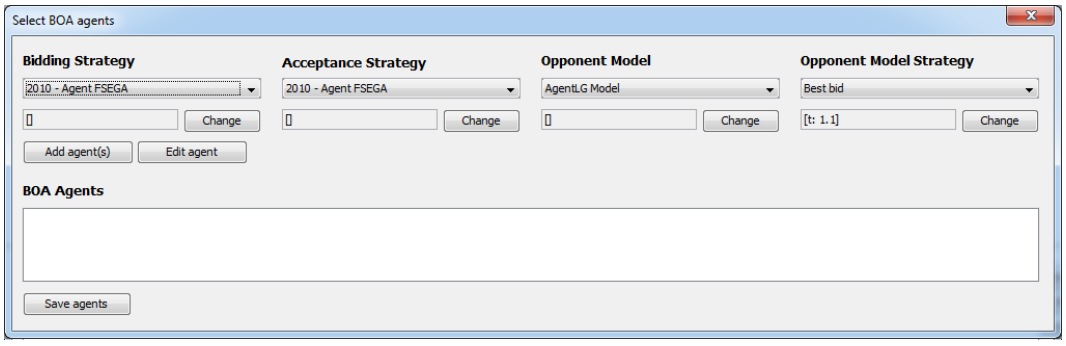

Figure 3.6: The BOA framework GUI.

Many of the components have parameters which can slightly alter the behavior of the component. When selecting a component from the BOA interface, the default parameters appear in the textbox below (Figure 3.6).

To change these parameters the user can click *change* next to the textbox, which will bring up a user interface allowing the user to edit the parameters. Figure 3.7 shows an example where the parameters of AC_{next} are displayed. The user has the option to create one BOA agent using AC_{next} with the current parameters or multiple BOA agents where the parameters of AC_{next} differ.

In Figure 3.7 we see three agents are being made of which the BS and the OM are the same, however the AC's have different parameters. The figure shows that the lower bound of parameter a is 1.0 and it will make steps of 0.1 till it reaches the upper bound, which is 1.2, resulting in three different AC's with parameters (a = 1.0, 1.1 and 1.2). Note that in this example we only varied a single parameter of a single component. If we vary more parameters of different components, then all possible combinations are generated.

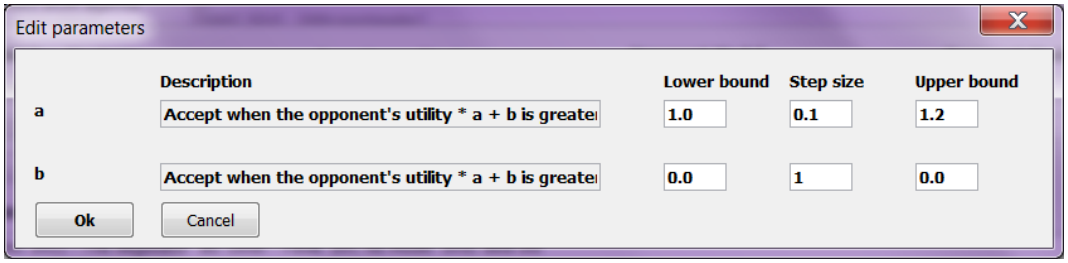

Figure 3.7: Adding a parameter.

3.3 Decoupling Existing Agents

In this section we answer the question *whether it is possible to identify generic components within most, if not all, negotiation strategies?* As a validation of our architecture, we provide empirical evidence that many of the currently existing agents can be decoupled, by separating a set of state of the art agents into its components. This section serves three goals: first, we discuss how existing agents can be decoupled into a BOA agent, second, we argue that the BOA architecture design is appropriate, as most agents will turn out to fit in our architecture and third, we discuss and apply a method to determine if the sum of the components – the BOA agent – is equal in behavior to the original agent.

3.3.1 Identifying the Components

In this section we identify the components of 21 negotiating agents, taken from the ANAC competitions of 2010 [13], 2011 [12] and 2012. We selected these agents as they represent the current state of the art in automated negotiation, having been implemented by various negotiation experts.

Since the agents were not designed with decoupling in mind, all agents had to be re-implemented to be supported by the BOA architecture. Our decoupling methodology was to adapt an agent's algorithm to enable it to switch its components, without changing the agent's functionality. A method call to specific functionality, such as code specifying when to accept, was replaced by a more generic call to an AC, which can then be swapped at will. The contract of the generic calls are defined by the expected input and output of every component, as outlined in Section 3.1.2.

The first step in decoupling an agent is to determine which components can be identified. For example, in the ANAC 2010 agent *FSEGA* [114], an AC, a BS, and an OM can all be identified. The AC combines simple, utility–based criteria (called $\mathbf{AC}_{\text{const}}$ and $\mathbf{AC}_{\text{prev}}$ in [10]), and can be easily decoupled in our architecture. The OM is a variant of the *Bayesian opponent model* [7, 9, 53], which is used to optimize the opponent utility of a bid. Since this is consistent with our architecture (i.e. the OM provides opponent utility information), the model can be replaced by a call to the generic opponent model interface. The final step is to change the BS so that it can use the generic OM's and AC's instead of its own specific implementation. In addition to this, the OM and the AC need to be altered to allow other BS's to use it. Other agents can be decoupled using a similar process.

Unfortunately, some agent implementations contain slight dependencies between different components. These dependencies needed to be resolved before an agent could be separated into singular components. For example, the AC and the BS of the ANAC 2011 agent *The Negotiator* [30] rely on a shared target utility. In such cases, the agent can be decoupled by introducing Shared Agent State (SAS) classes. A SAS class avoids code duplication and thus performance loss by sharing the code between the components. Where one of the components uses the SAS to calculate the values of the required parameters and saves the results, while the other component simply asks for the saved results instead of repeating the calculation. In case the first component is replaced by another, which does not use the SAS, then the second component will calculate the values of the required parameters and the component can thus be used as a normal one.

Table 3.1 provides an overview of all agents that we re-implemented in our architecture and more specifically, which BOA components we were able to decouple. In fact, we were able to decouple all ANAC 2010, and most ANAC 2011 and ANAC 2012 agents. For a complete overview of all the components that were decoupled or created, see appendix B.

Complications of Decoupling Agents

Two agents (*ValueModelAgent* [41] and *Meta-Agent* [57]) were not decoupled due to practical reasons, even though theoretically that would have been possible. The *ValueModelAgent* was not decoupled because there were unusually strong dependencies between its components. Decoupling the strategy would have resulted in computationally heavy components when trying to combine them with other components, making them impractical to use. The ANAC 2012 *Meta-Agent* chooses to imitate one of the 17 agents from the ANAC 2011 qualifying round. This agent was not decoupled because it would have required the decoupling of all 17 agents, of which only 8 optimized versions actually entered the finals.

Agents	**Components**		
ANAC 2010	**B**	**O**	**A**
FSEGA [114]	✓	✓	✓
Agent K [65]	✓	∅	✓
Agent Smith [127]	✓	✓	✓
IAMcrazyHaggler [130]	✓	∅	✓
IAMhaggler [130]	✓	✓	✓
Nozomi	✓	∅	✓
Yushu [2]	✓	∅	✓
ANAC 2011			
Agent K2 [66]	✓	∅	✓
BRAMAgent [40]	✓	–	✓
Gahboninho [18]	✓	–	✓
HardHeaded [128]	✓	✓	✓
IAMhaggler2011 [133]	✓	∅	✓
Nice Tit for Tat [12]	✓	✓	✓
The Negotiator [30]	✓	∅	✓
ANAC 2012			
AgentLG	✓	∅	✓
AgentMR	✓	∅	✓
BRAMAgent2	✓	–	✓
CUHKAgent [45]	✓	–	–
IAMhagger2012	✓	∅	✓
OMAC Agent [27]	✓	∅	✓
The Negotiator Rel.	✓	✓	✓

Table 3.1: Overview of components identified in every agent. ✓: original has component that can be decoupled. ∅: original has no component, but it can be added. – : no support for such a component.

The *CUHKAgent*, like the *ValueModelAgent*, is heavily coupled with multiple variables that are shared between the BS and the AC. This makes it very hard to decouple and can make the components impractical to use with other components. However, since the *CUHKAgent* was placed first in the ANAC 2012 competition, we decided to decouple its BS, allowing it to work with other AC's.

Four additional agents were only partially decoupled: *AgentLG*, *BRAMAgent*, *BRAMAgent2*, and *Gahbininho*. As is evident from Table 3.1, the only obstacle in decoupling these agents fully is how they use their OM, as that can be employed in many different ways, for example agents such as *Nice Tit for Tat*, attempt to estimate the Nash point on the Pareto frontier. Other common applications include: ranking a set of bids according to the opponent utility, reciprocating the opponent's utility and estimating the opponent utility. The generic opponent model interface needs to sufficiently accommodate such requirements

from the BS to make interchangeability possible. For this reason we require the opponent model interface to be able to produce the estimated opponent utility of an arbitrary negotiation outcome.

Depending on how the OM is used there are three groups of agents: first, there are agents such as *FSEGA* [114], which use an OM that can be freely interchanged, second, there are agents such as the ANAC 2010 winner *Agent K* [65], which do not have an OM themselves but can be extended to use one. Such agents typically employ a BS that first decides upon a specific target utility range, and then selects a *random* bid within that range. These agents can easily be fitted with an OM, by passing bids in the utility range through the OM before sending out the bid. Third, there are agents, whose OM are not compatible with our generic OM. For example *Gahboninho* uses its OM to measure how strong an opponent concedes, or like *CUHKAgent* and *AgentLG* who both use a "similarity heuristic" which is not compatible with our architecture, as their OM's do not yield enough information to compute the opponent utility of a bid. For these type of agents, we consider their "opponent model" as part of the BS. These strategies are also extended to be able to work with other OM's like the previous group.

When decoupling the agents, we can distinguish different sub-types within each BOA component except for the BS, which varies greatly between different agents. For instance, there are only three main types of OM's being used: *Bayesian models*, *Frequency models*, and *Value models*. Bayesian models are an implementation of a (scalable) model of the opponent preferences that is updated using Bayesian learning [53, 134]. The main characteristic of frequency based models is that they track the frequency of occurrence of issues and values in the opponent's bids and use this information to estimate the opponent's preferences. Value models take this approach a step further and solely focus on the frequency of the issue values. In practice Bayesian models are computationally intensive, whereas frequency and value models are relatively light-weight.

Similar to the OM's, there are different types of acceptance mechanisms, where most agents use variations or combinations. Specifically, many agents use simple thresholds for deciding when to accept (called $\mathbf{AC}_{\text{const}}$ in [10]) and linear functions that depend on the utility of the bid under consideration ($\mathbf{AC}_{\text{next}}(\alpha, \beta)$ [10]).

3.3.2 Testing Equivalence of BOA Agents

A BOA agent should behave identically to the agent from which its components were derived. The equivalence of an agent with the BOA version was verified in two ways: first, given the same negotiation environment and the same state both agents should behave in identical ways (Section 3.3.2) and second, the performance in a real time negotiation of both agents should be similar (Section 3.3.2).

Identical behavior does not necessarily imply similar performance, since we adapted the agents' code to follow the BOA architecture so that it would be compatible with all types of components, we introduced redundancies which can slow down an agent. For example, while the BOA architecture keeps a record of the bidding history of the opponent, it is possible that a BS also records the opponents' offers in its own manner. This would cause

the agent to keep two records which can slow down the agent and cause a decrease in its performance.

Identical Behavior Test

Two deterministic agents can be considered equivalent if they perform the same action given the same negotiation trace. There are two main problems in determining equivalence: first, most agents are non-deterministic as they behave randomly in certain circumstances, for example when selecting from a set of bids of similar utility and second, the default protocol in GENIUS uses real time [85, 49], which is highly influenced by CPU performance. This means that in practice two runs of the same negotiation are never exactly equivalent.

To be able to run an equivalence test despite agents choosing actions at random, we fixed the seeds of the random functions of the agents. The challenge of working in real time was dealt with by changing the real time deadline to a maximum amount of rounds. Since time does not pass within a round, cpu performance does not play a role.

All agents were evaluated on the ANAC 2011 domains (see [6] for a domain analysis). The ANAC 2011 domains vary widely in characteristics: the number of issues ranges from 1 to 8, the size from 3 to 390625 possible outcomes, and the discount from none (1.0) to strong (0.424). Some ANAC 2010 agents, specifically *Agent Smith* and *Yushu*, were not designed for large domains and were therefore run on a subset of these domains.

The *opponent strategies* used in the identical behavior test should satisfy two properties: the opponent strategy should be deterministic and the opponent strategy should not be the first to accept, to avoid masking errors in the agent's AC. Given these two criteria, we used the standard time-dependent tactics [36, 38] for the opponent bidding strategy. Specifically, *Hardliner* ($e = 0$), *Linear Conceder* ($e = 1$) and *Conceder* ($e = 2$). In addition, we used the *Offer Decreasing* agent, which offers all the possible bids one after the other in decreasing order of utility.

All original and BOA agents were evaluated against these four opponents, using both preference profiles defined on all eight ANAC 2011 domains. Both strategies were run in parallel, making sure that the moves made by both agents were equivalent at each moment. After the experiments were performed, the results indicated that all BOA agents were identical to their original counterparts except for *AgentMR* and *AgentLG*. Both these agents do not have identical behavior with its BOA counter-part because of the order in which the components are called. Their implementation requires that they first test if the opponent's bid is acceptable, and then determine the bid to offer. As discussed above, this is exactly the opposite of what the BOA agent does, which can cause inconsistencies between the two agents.

Similar Performance Test

Two agents can perform the same action given the same input, but may still achieve different results because of differences in their real time performance. When decoupling agents, there is a trade-off between the performance and interchangeability of components. For example, some agents record only a partial negotiation history, while other strategies require a

Agent	Amsterdam Trip	Camera	Car	Energy	Grocery	Company Acquisition	Laptop	Nice or Die	Mean utility
HardHeaded	0.891	**0.818**	**0.961**	0.664	0.725	0.747	0.683	**0.571**	**0.757**
Gahboninho	**0.912**	0.659	0.928	**0.681**	0.667	0.744	0.726	**0.571**	0.736
Agent K2	0.759	0.719	0.922	0.467	0.705	0.777	0.703	0.429	0.685
IAMhaggler 2011	0.769	0.724	0.873	0.522	0.725	**0.814**	**0.749**	0.300	0.685
BRAMAgent	0.793	0.737	0.815	0.420	0.724	0.744	0.661	**0.571**	0.683
The Negotiator	0.792	0.744	0.913	0.524	0.716	0.748	0.674	0.320	0.679
Nice Tit for Tat	0.733	0.765	0.796	0.508	0.759	0.767	0.660	0.420	0.676
Value Model Agent	0.839	0.778	0.935	0.012	**0.767**	0.762	0.661	0.137	0.611

Table 3.2: ANAC 2011 reference results of the original agents using our hardware (n = 10). Best results are marked bold.

full bid history of the agent and/or its opponent. We elected for the most universal approach (recording the full bid history), even though this can have a negative influence on the performance. We demonstrated however, that while there is some performance difference when decoupling existing agents, this does not significantly impact the negotiation outcome.

The performance of the BOA agents was tested by letting them participate in the ANAC 2011 tournament (using the same setup [6]). The decoupled ANAC 2011 agents replaced the original agents, resulting in a tournament with eight participants. For the other BOA agents this was not possible, as their original counterparts did not participate in the ANAC 2011 competition. For each of these agents we therefore ran a modified tournament in which we added the original agent to the pool of ANAC 2011 agents, resulting in a tournament with nine participants. We repeated this process for the BOA agents and evaluated the similarity of the results.

For our experimental setup we used computers that were slower compared to the IRIDIS high-performance computing cluster that was used to run ANAC 2011. As we were therefore unable to reproduce exactly the same data, we first recreated our own ANAC 2011 tournament data as depicted in Table 3.2, which we used as our baseline to benchmark the decoupled agents. The difference in computer performance caused small changes compared to the official ANAC 2011 ranking, as *Agent K2* moved up from 5th to 3rd place.

Table 3.3 provides an overview of the results. We evaluated the performance in terms of the difference in the overall utility as well as the difference in time of agreement between the original and the BOA agents. The table does not list the agents that were not decoupled, and we also omitted *The Negotiator Reloaded* from the test set, as this agent was already submitted as a fully decoupled BOA agent.

From the results, we can conclude that the variation between the original and the BOA version is minimal; the majority of the standard deviations for both the difference in overall utility and time of agreement are close to zero. The largest difference between the original and decoupled agents with regard to the average time of agreement is 0.010 (*Agent Smith*);

and for the average utility, the largest difference is 0.015 (*BRAMAgent2*). Hence, in all cases the BOA agents and their original counterparts show comparable performance.

	Avg. time of agr.	SD time of agr.	Avg. utility	SD utility
Agent FSEGA (Org.)	0.425	0.0013	0.721	0.0009
Agent FSEGA (Dec.)	0.426	0.0041	0.721	0.0024
Agent K (Org.)	0.713	0.0057	0.666	0.0035
Agent K (Dec.)	0.714	0.0061	0.672	0.0045
Agent Smith (Org.)	0.469	0.0083	0.703	0.0041
Agent Smith (Dec.)	0.479	0.0053	0.707	0.0041
IAMcrazyHaggler (Org.)	0.591	0.0103	0.699	0.0078
IAMcrazyHaggler (Dec.)	0.587	0.0069	0.702	0.0099
IAMhaggler2010 (Org.)	0.633	0.0110	0.682	0.0093
IAMhaggler2010 (Dec.)	0.636	0.0101	0.684	0.0066
Nozomi (Org.)	0.663	0.0071	0.704	0.0063
Nozomi (Dec.)	0.666	0.0062	0.708	0.0053
Yushu (Org.)	0.798	0.0030	0.715	0.0035
Yushu (Dec.)	0.800	0.0026	0.717	0.0037
Agent K2 (Org.)	0.619	0.0046	0.685	0.0040
Agent K2 (Dec.)	0.621	0.0050	0.686	0.0034
BRAMAgent (Org.)	0.683	0.0089	0.683	0.0054
BRAMAgent (Dec.)	0.687	0.0060	0.683	0.0033
Gahboninho (Org.)	0.667	0.0055	0.736	0.0044
Gahboninho (Dec.)	0.668	0.0053	0.742	0.0015
HardHeaded (Org.)	0.738	0.0009	0.758	0.0024
HardHeaded (Dec.)	0.735	0.0028	0.749	0.0034
IAMhaggler2011 (Org.)	0.494	0.0102	0.685	0.0023
IAMhaggler2011 (Dec.)	0.493	0.0078	0.683	0.0024
Nice Tit for Tat (Org.)	0.677	0.0078	0.676	0.0043
Nice Tit for Tat (Dec.)	0.683	0.0070	0.668	0.0025
TheNegotiator (Org.)	0.716	0.0016	0.679	0.0027
TheNegotiator (Dec.)	0.716	0.0014	0.679	0.0023
BRAMAgent 2 (Org.)	0.713	0.0087	0.750	0.0093
BRAMAgent 2 (Dec.)	0.715	0.0064	0.735	0.0069
IAMhaggler2012 (Org.)	0.497	0.0046	0.734	0.0017
IAMhaggler2012 (Dec.)	0.492	0.0041	0.721	0.0024
OMAC Agent (Org.)	0.819	0.0028	0.767	0.0124
OMAC Agent (Dec.)	0.822	0.0016	0.779	0.0077

Table 3.3: The table shows performance (with standard deviation) of agents in an ANAC 2011 tournament before and after being decoupled.

Chapter 4

Uses of the BOA Architecture

The previous chapter presented details of the BOA architecture, what its underlying ideas are and why it is useful in the development of negotiation strategies. It also showed that the BOA architecture is not just a theoretical concept but also an applicable one, as we were able to decouple many existing agents. This chapter builds upon this, looking at how the BOA architecture can be used and applied in the field of automated negotiation. Being integrated with GENIUS, it is used internationally by various academic institutes, which we discuss in section 4.1. Section 4.2 illustrates that by switching components we can systematically explore the space of possible negotiation strategies for effective ones. The BOA architecture also makes it easy to create and tune agent strategies with its ability to reuse and focus on specific components. Section 4.3 presents *The Negotiator Reloaded*, which is a negotiation strategy that was built and tuned using the BOA architecture. The Negotiator Reloaded was the first BOA agent to compete in the ANAC competitions, reaching third place in ANAC 2012.

4.1 International Adoption

The BOA architecture has been integrated into the well-established negotiation framework GENIUS [49, 85], which is the simulation platform used by various researchers within the field of automated negotiation. GENIUS supports research that investigates the process of automated negotiations [120], comparing the negotiation process between traditional and automated negotiations [115] or looking at negotiation settings where agents can negotiate concurrently with multiple opponents [132]. A number of researchers spend their time investigating the negotiation process related to specific domains [29, 119], while others develop negotiation environments in order to test and observe negotiations [33, 85]. Some research look at the negotiation process from a higher level, looking at negotiation protocols [71, 121], while others focus on certain aspects, like ways of learning and presenting preferences [4, 25, 26] or the use of emotional-social agents [106]. GENIUS also aids research that investigates human negotiators within the automated negotiation process [34, 64, 115]. Lin et al. has done extensive research on this subject [82, 84], look-

ing at how automated agents can help improve the ability of humans to negotiate [79, 80]. Lin et al. even developed an automated mediator in order to aid negotiations involving humans [81, 83]. Much work concentrates on negotiation strategies and their performance [6, 28, 111, 117, 118, 129], where GENIUS provides the means to test their abilities as well as the underlying theories and hypotheses. The largest source of negotiation strategies however, comes from ANAC competitions [6, 13], where various strategies from ANAC 2010 [2, 65, 114, 127, 130], ANAC 2011 [12, 30, 40, 41, 128] and ANAC 2012 [27, 45] have been tested.

With the implementation of the BOA architecture, more research focuses on the separate components, for example research that investigates acceptance conditions [5, 11, 14] and opponent modeling techniques [7, 47].

The following sections show how the BOA architecture has been adopted alongside the GENIUS framework, where section 4.1.1 explains what the Automated Negotiation Agent Competition is and how in the past two competitions BOA agents performed well in it. Aside from being used in the ANAC competition and various research, it has also found its way into various academic courses (section 4.1.2).

4.1.1 Automated Negotiation Agent Competition (ANAC)

The Automated Negotiation Agent Competition (ANAC) is a yearly international negotiation competition where automated agents negotiate against each other in a negotiation setting, which is a bilateral, multi-issue and uses the alternating offers protocol [110]. The ANAC competitions use GENIUS as their negotiation platform and since the integration of the BOA architecture, BOA agents have been participating in the ANAC competitions. The aim of the competition is to advance the state-of-the-art in the area of automated negotiations. The goals of ANAC include:

1. encourage and stimulate the design of new and innovative negotiation agents that can negotiate proficiently in a variety of settings and circumstances,
2. evaluate the different negotiation strategies objectively,
3. explore different learning and adaptation strategies and opponent models,
4. collect state-of-the-art negotiating agents and negotiation scenarios, and make them available and accessible as benchmarks for the negotiation research community.

The first ANAC competition was held in 2010 and was based on the alternating offers protocol [110] which was extended to include a real time deadline of 3 minutes. In the following years the protocol's scope was extended to allow for more complex and realistic settings. Time-based discount factors were added to ANAC 2011 [6] and the protocol supported reservation values in ANAC 2012. In ANAC 2013 the negotiation process was altered to allow for repeated negotiations, where negotiation agents were able to use the results of previous negotiation sessions.

Since its implementation in 2011, the BOA architecture has been used in subsequent ANAC competitions. The first BOA agent, *The Negotiator Reloaded*, participated in the ANAC 2012 competition, making it the first ANAC competition to use the BOA architecture. In the qualifying round, *The Negotiator Reloaded* was placed in the top five, securing a place in the final round. In the final round, TNR took the award for the best performing agent in non-discounted domains, which was one of the two parts of the competition. The agent also achieved third place in the overall standing of the competition.

The ANAC 2013 competition also supported the development and the use of BOA agents. Several BOA agents were submitted to the competition, of which two went through to the finals (*Fawkes* and *Inox*). In the final round *The Fawkes* agent won the 2013 competition while *Inox* finished fourth.

The BOA architecture aided the design of these agents by providing the foundation on which these negotiation strategies were developed, using a "Plug and Play" system. By recombining components to create their strategy, developers are inadvertently exploring the space of negotiation strategies, trying to find an effective strategy. Section 4.2 discusses the search for effective strategies in more detail. The BOA architecture offers the developers two ways in which they can design their negotiation agents. One approach is to create a new and innovative strategy. The other approach is to reuse already existing components and ideas. The advantage of using already existing component is that developers are building upon an already tried and tested negotiation strategy, where they can focus on tuning and optimizing these components.

The developers of Inox opted to create its own BS, while the developers of Fawkes chose to base their BS on the ANAC 2012 agent OMAC [27]. To improve the BS of OMAC, the Fawkes developers used a dynamic target utility range, making the agent more generous as the time passes, instead of using a fixed target utility range. Both Fawkes and Inox used a simple frequency model in order to offer more enticing bids to the opponent without sacrificing its own utility, thereby increasing the chances of reaching an agreement. Inox made modifications to its OM by scaling the value frequencies (based on a scaling function) in order to prevent bids that were offered multiple times to skew the estimated opponent utility space. When we looked at the AC's of these two agents, Inox created its own AC, which will accept only if the opponent's bid is higher or equal to our worst bid it has already offered to the opponent. Fawkes chose to reuse already existing acceptance mechanisms presented in [11] to construct its AC.

4.1.2 Scholarly Use

GENIUS and the BOA architecture has also been found in the classroom, having been integrated into artificial intelligence courses at academic institutes such as *Bar-Ilan University*, *Ben-Gurion University of the Negev*, *Maastricht University*, and *Delft University of Technology*, where part of the syllabus covers automated negotiation and the creation of negotiation strategies.[1] The BOA architecture offers the students an easier and more structured way to

[1]Educational material for the BOA architecture can be freely downloaded from `ii.tudelft.nl/genius/#Education`

develop a negotiation strategy, and causes them to think more critically about the components they design themselves, which in turn helps them understand the inner workings of a negotiation strategy.

4.2 In the Search of an Effective Negotiation Strategy

The BOA architecture enabled us to find an effective negotiation strategy within the large space of possible strategies by changing its components [7, 9]. By having a negotiation strategy divided into different components we can change one component at a time and test its performance. This allowed us to systematically test different negotiation strategies to find improved ones. Section 4.2.1 describes a technique integrated into the BOA architecture, which aids the search process by reducing the number of negotiations and thus the time needed to run a set of negotiation strategies. We were able to achieve this by running multiple AC's in parallel. Section 4.2.2 illustrates the ability of the BOA architecture to assist in the exploration of the negotiation strategy space and look for a more effective (ANAC) strategy, by way of switching components.

4.2.1 Multi Acceptance Condition (MAC)

Searching through the large space of possible negotiation strategies is an immense task. To scale down the number of negotiations, we can use a Multi Acceptance Condition (MAC) mechanism, which runs multiple AC's simultaneously and thereby reduces the computational cost of running a set of negotiation strategies.

To illustrate this, assume that two negotiating BOA agents A and B have identical bidding behavior and the same opponent modeling techniques, so that only their AC's differ. Furthermore, suppose agent A accepted halfway through the negotiation, while agent B accepted somewhere towards the end. The agents accepted at different times during the negotiation, but their bidding behavior will be identical up to the point of the first acceptance. In other words, the only difference between the two traces is that the trace of agent A stops earlier.

The BOA architecture exploits this property by running multiple AC's in parallel while recording, when each of them accept the opponent bid. This drastically reduced the amount of component combinations, from combining three components to combining only two, as the number of AC's is reduced to 1. This technique can be used to easily optimize and tune AC's by running many of them with varying parameters in the same negotiation thread. It should be noted that this approach assumes that checking additional AC's do not introduce large computational overhead. In practice, more than 50 variants of a simple AC, for example $\mathbf{AC}_{next}$ [11] can be tested in the same negotiation with negligible computational overhead.

To create a multi-acceptance condition component, one first needs to extend the class that we implemented, *Mulit Acceptance Condition*. This gives access to the ACList which is a list of AC's that will be tested in parallel. One needs to add the AC to the ACList to include it in the negotiation as shown below.

```
public class AC_MAC extends Multi_AcceptanceCondition {

  @Override
  public void init(NegotiationSession negoSession,
     OfferingStrategy strat,
     HashMap<String, Double> parameters) throws Exception {
        this.negotiationSession = negoSession;
        this.offeringStrategy = strat;
        outcomes = new ArrayList<OutcomeTuple> ();
        ACList = new ArrayList<AcceptanceStrategy>();
        for (int e = 0; e < 5; e++) {
             ACList.add(new AC_Next(negotiationSession,
                              offeringStrategy, 1, e * 0.01));
       }
     }
}
```

4.2.2 Improving the State-of-the-Art

Using the scaling methods discussed in the previous section, we demonstrate a practical application of the BOA architecture; showing how it can be employed to explore the negotiation strategy space for an effective strategy. To be able to do this, we took the original BS of every ANAC agent and attempted to find a better accompanying OM and AC.

Searching the Negotiation Space

We used the following combinations of BOA components:

(B) For the BS, we used all ANAC agents that we were able to successfully decouple (see Table 3.1).

(O) For our OM set, we selected the best Bayesian model (*IAMhaggler Bayesian Model* [131]), frequency model (*Smith Frequency Model* [127], and the best value model (*CUHKAgent Value Model* [45]) as identified in [9].

(A) All the AC's of the top four agents of ANAC 2012 were used, except for *CUHKAgent* as it was not be decoupled. In addition, we used a set of baseline AC's, such as $\mathbf{AC}_{\text{combi}}(T, \text{MAX}^W)$ [11], and an optimal stopping acceptance condition $\mathbf{AC}_{\text{opt.stop}}$ which is based on a Gaussian process strategy prediction as discussed in [14]. Table 4.1 provides an overview of all 15 tested AC's.

For each BS, we ran a tournament on a subset of the ANAC 2012 domains against the eight ANAC 2012 agents. Even if MAC is applied, the space to be explored can still be impractically large. This is already problematic for a limited amount of domains and agents. To illustrate, ANAC 2011 consists of 448 negotiation sessions [6] which may all last 3 minutes. In the worst case, it requires 22 hours to run a single tournament, and almost four weeks to run it 28 times, as we did for the similarity test discussed in Section 3.3.2.

Acceptance Condition	Range	Increments
$\mathbf{AC}_{\text{combi}}(T, \text{MAX}^W)$	T$\in$ [0.95, 0.99]	0.01
$\mathbf{AC}_{\text{next}}(\alpha, \beta)$	$\alpha \in$ [1.0, 1.05]	0.05
	$\beta \in$ [0.0, 0.1]	0.05
$\mathbf{AC}_{\text{opt.stop}}$	–	–
$\mathbf{AC}_{\text{AgentLG}}$	–	–
$\mathbf{AC}_{\text{OMAC}}$	–	–
$\mathbf{AC}_{\text{TheNegotiatorReloaded}}$	–	–

Table 4.1: All acceptance conditions that were used in the experiment to search the negotiation strategy space.

We opted to use a representative subset of the domains from ANAC 2012 to improve scalability. The following domains were used: *Barter*, *IS BT Acquisition*, *Barbecue*, *Phone*, *Energy (small)* and *Supermarket*. Since the ANAC 2010 agents are not compatible with discounts and reservation values, these were removed from the domains. To further improve scalability, a rounds-based protocol was used with a deadline of 3000 rounds and we used the scalability optimization techniques (MAC) as discussed in Section 4.2.1. The complete tournament is repeated five times to improve the reliability of the results.

Experimental Results

From the 19 ANAC agents considered in this work, we were able to considerably improve 16, as depicted in Table 4.2. This table shows the optimal AC and OM for each agent, as well as their scores in the tournament. Due to scalability issues, some agents were only run on the four smallest domains instead of all six domains.

Besides the utility gain, the overview also indicates the agent's ranking before and after the optimization of the components. As is evident from the results, most agents were significantly improved by swapping their components with the optimized versions. To illustrate: *IAMcrazyHaggler*'s ranking improved from the twelfth place to the fourth when it employed *IAMhaggler*'s OM, and optimal stopping as its AC.

The only agents we were not able to improve were *Yushu*, *The Negotiator* and *BRAMAgent2*. There are two main reasons for this: the first reason is that some of these agents do not use an OM optimally, therefore the BS does not benefit from an improved OM. The second reason is that these agents employ AC's that already performs well.

An interesting pattern in the results is that nearly all agents were improved by using the acceptance condition $\mathbf{AC}_{\text{opt.stop}}$ [14]. For the OM the *IAMhaggler Bayesian Model* is often best, however the results indicate that the differences between the OM's are minimal. This shows that a better AC often results in a larger gain than an improved OM.

	OM	**AC**	**Diff**$_6$	**Rank**$_6$ **pre**	**Rank**$_6$ **post**	**Diff**$_4$	**Rank**$_4$ **pre**	**Rank**$_4$ **post**
CUHKAgent	–	$\mathbf{AC}_{\text{next}}(1,0)$	0.001	1	1	0.081	1	1
Gahboninho	–	$\mathbf{AC}_{\text{AgentLG}}$	0.006	2	2	0.007	2	2
The Neg. Rel.	*CUHK*	$\mathbf{AC}_{\text{opt.stop}}$	0.037	3	2	0.026	3	2
OMAC Agent	*CUHK*	$\mathbf{AC}_{\text{AgentLG}}$	0.014	4	4	0.005	5	5
Agent K2	*IAH*	$\mathbf{AC}_{\text{opt.stop}}$	0.042	5	4	0.042	6	4
Agent K	*CUHK*	$\mathbf{AC}_{\text{opt.stop}}$	0.047	6	4	0.044	7	5
IAMhaggler2011	*IAH*	$\mathbf{AC}_{\text{opt.stop}}$	0.022	7	7	0.008	9	9
IAMhaggler2012	*Smith*	$\mathbf{AC}_{\text{opt.stop}}$	0.059	8	4	0.077	11	5
HardHeaded	*IAH*	$\mathbf{AC}_{\text{opt.stop}}$	0.134	9	3	0.133	13	2
BRAMAgent	–	$\mathbf{AC}_{\text{opt.stop}}$	0.036	10	7	0.037	10	8
Nozomi	*Smith*	$\mathbf{AC}_{\text{opt.stop}}$	0.160	11	4	0.155	14	4
IAMcrazyHaggler	*IAH*	$\mathbf{AC}_{\text{opt.stop}}$	0.190	12	4	0.186	16	5
FSEGA	*CUHK*	$\mathbf{AC}_{\text{opt.stop}}$	–	–	–	0.027	12	8
Agent Smith	*IAH*	$\mathbf{AC}_{\text{OMAC}}$	–	–	–	0.103	15	9
IAMhaggler	*IAH*	$\mathbf{AC}_{\text{opt.stop}}$	–	–	–	0.072	8	4
Nice Tit for Tat	*IAH*	$\mathbf{AC}_{\text{opt.stop}}$	–	–	–	0.025	4	2

Table 4.2: Results of the optimized BOA agents, when tested on both $n = 6$ domains and $n = 4$ domains. The **Diff**$_n$ column indicates the utility gain of the agent when coupled with the optimal components listed in the **OM** and **AC** column. **Rank**$_n$ **pre** indicates the rank of the original agent, while **Rank**$_n$ **post** gives its ranking after optimization.

4.3 Building New Agents

With the aid of the BOA architecture, it is simple to create and tune new agents, allowing the reuse of good performing or well-known components. Being able to separate a negotiation strategy into components makes the development much easier, dividing the workload into manageable size tasks.

4.3.1 The Negotiator Reloaded

The Negotiator Reloaded (TNR) is the first agent to be developed and tuned using the BOA architecture and participated in the ANAC 2012 competition. This section presents the implementation of the TNR components as well as how they were tuned and optimized. We also evaluated the performance of TNR using certain quality measures and comparing them to the performances of other ANAC agents.

Bidding Strategy

The BS of TNR takes the opponent's strategy and domain characteristics into account to optimize its negotiation strategy. The discussion below follows the diagram of the complete negotiation strategy depicted in Figure 4.1.

The first step taken by TNR, is to determine if the discount of the domain is low, medium, or high. Next, the duration of the negotiation is divided into a set of windows. At the start of each window, the *domain analyzer* is used to estimate the Kalai-Smorodinsky

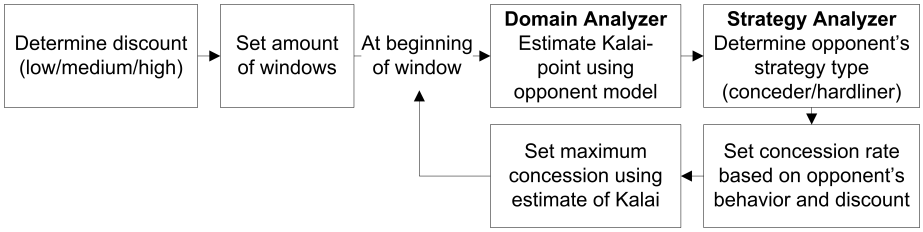

Figure 4.1: Overview of bidding strategy of TNR.

point and the *strategy analyzer* is used to determine whether the opponent is a conceder or hardliner. It is preferable to do these calculations at each turn, however this proved to be computationally expensive. The target utility in a specific round is determined using the time-dependent decision function [36] depicted in Equation 4.1. We opted for this decision function as its parameters can be adjusted during the negotiation.

$$P_{min} + (P_{max} - P_{min}) \cdot (1 - F(t)) \text{ where } F(t) = k + (1 - k) \cdot t^{1/e}. \tag{4.1}$$

The value for the concession rate e is selected from a table depending on the discount type (low, medium, high) and opponent's strategy type (conceder or hardliner). While the discount type does not alter, the opponent's behavior is likely to change over time. The greatest amount of concession, P_{min}, is set to the estimated Kalai-Smorodinsky point, which is calculated by the *domain analyzer*. For domains with a discount, P_{min} is multiplied by the discount factor to ensure that the agent concedes faster. When the undiscounted reservation value is higher than P_{min}, then P_{min} is set to the reservation value. As a safeguard, P_{min} is not allowed to be lower than a predefined constant. The variable k is always 0, and P_{max} 1. The calculated target utility is used by the bid selector to select a bid with a utility as close as possible to the target utility.

The tactic discussed above strongly relies on the concession rate table. Since three discount types and two strategy types are distinguished, there are six concession rates that need to be determined. In order to optimize the BS of TNR, we created a variant of the ANAC 2011 competition that excludes the agent *ValueModelAgent* and the domains *Energy* and *NiceOrDie* to reduce the computational time. For each discount type we generated a representative set of domains, for example for the type medium discounts we created a set of preference profiles with discounts in the range $(0.4, 0.8]$. Next, we ran the competition multiple times for each discount type to determine the optimum values for the strategy type parameters.

Opponent Model

In a paper by Baarslag et al. [7], the *IAMhaggler Bayesian Model* introduced by Williams et al. [133] was found to be the most accurate in estimating the Kalai-Smorodinsky point, which *The Negotiator Reloaded* uses as part of its *domain analyzer*.

We optimized the OM in a few ways. The computational time required by the *IAMhaggler Bayesian Model* depends strongly on the domain size, where on larger domains the

OM slows down the negotiation agent. Therefore the OM is not used in very large domains, in this case the agent estimates the Kalai-Smorodinsky point to be equal to a predefined constant. While the accuracy of the estimation increases at the beginning of the negotiation, it eventually decreases over time. We believe that this can be attributed to the assumed decision function, that more accurately reflects the real decision function at the beginning of the negotiation for most agents. To avoid this decay in accuracy, TNR stops updating the OM after a predefined amount of time.

Acceptance Condition

The AC of TNR consists of a set of basic acceptance mechanisms discussed by Baarslag et al. [11]. The flowchart of the acceptance condition is depicted in Figure 4.2. As visualized, there are two paths depending whether the discount is negligible or not and six parameters $(\alpha, \beta, \gamma, \delta, \epsilon, \zeta)$.

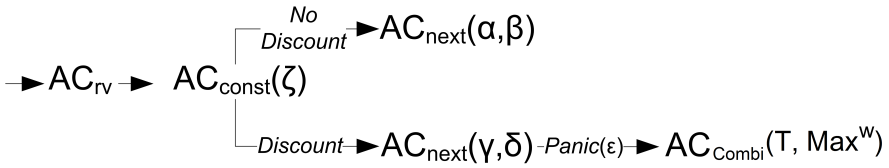

Figure 4.2: Basic acceptance conditions used by TNR.

AC_{rv} is an AC that decides to accept when the (discounted) utility of the opponent's bid is lower or equal to the reservation value. AC_{const} is an AC that accepts when the utility of the opponent's bid is at least equal to a constant ζ. AC_{next} accepts when a linear transformation of the opponent's bid utility is better than the utility of the bid it is about to offer. Finally, TNR uses $AC_{Combi}(T, Max^w)$, when there is $1 - \epsilon$ time left and the utilities of the bids of the agents have not crossed, the AC looks for the maximum bid from a set of previously made offers by the opponent within a particular time window. If the utility of the offered bid is higher than 0.5 and higher than the utility of the maximum bid, it will then accept.

The multi-acceptance condition (MAC), discussed in section 4.2.1 was used to optimize the AC. In total 288 AC's were tested varying in the usage of the panic phase and the four parameters of the two acceptance conditions AC_{next}. The parameters $\alpha = 1.0, \beta = 0.0, \gamma = 1.05, \delta = 0.0, \epsilon = 0.99$ were found to be optimal.

4.3.2 Empirical Evaluation

In this section we describe how we evaluated the performance of TNR. We ran a modified ANAC 2011 competition setup to compare TNR's performance to other ANAC agents.

Experimental Setup

To test the performance of TNR we ran a modified ANAC 2011 competition where *ValueModelAgent* was replaced with TNR. These agents competed on variants of the ANAC

2011 domains based on the three discount types, resulting in a total of 24 domains. The complete tournament was run ten times to increase the statistical significance of the results. In a single tournament eight agents competed against all agents except themselves on these 24 domains, playing both possible preference profiles. This resulted in a total of 13440 matches that were run using a distributed version of GENIUS. The overview of quality measures that were implemented to quantify the agent's performance is depicted in Table 4.3.

Quality measure	Description
Avg. time of agreement	The average time of agreement of all matches which resulted in agreement
Avg. discounted utility	Average discounted utility of all matches
Trajectory analysis	The opponent's moves can be classified based on their concession [20]. An unfortunate move for example, is a concession that accidentally results in a lower utility for the agent in comparison to the opponent's previous bid

Table 4.3: Overview of quality measures used to quantity performance.

Experimental Results

This section discusses the results of the tournament visualized in Table 4.4. The standard deviations were shown to be negligible and thus were not presented in the table nor the ratio of agreements as they were higher than 99% for all agents. This high percentage of agreement illustrates that most ANAC 2011 agents prefer agreement over disagreement, and ultimately give in, i.e. concede more in order to reach an agreement.

TNR achieved the highest discounted utility and strongly outperformed the runner-up. With regard to the trajectory measures, the agent made the least concessions, as indicated by its high percentage of silent moves and its low ranking on the percentage of unfortunate moves, fortunate moves, nice moves, and concession moves, which are the types of moves made when the agent tries to make a concession. TNR agent does not make selfish moves that increase its own utility without conceding within the opponents utility space, which can be attributed to its usage of the time-dependent strategy.

Since The Negotiator Reloaded placed within the top five in the qualifying rounds of the ANAC 2012 competition, we were invited to attend the ACAN workshop that was held in Valencia, Spain during the 2012 Autonomous Agent and Multi-Agent System (AAMAS) conference. At this conference we were asked to present our agent's strategy to the academic community. In addition, we also presented a short paper on the BOA architecture [8].

			Trajectory Analysis					
Agent	Avg. time of agreement	Avg. discounted utility	Avg. unfortunate moves	Avg. fortunate moves	Avg. nice moves	Avg. selfish moves	Avg. concession moves	Avg. silent moves
The Negotiator Reloaded	0.545	**0.809**	0.033	0.000	0.033	0.000	0.003	**0.930**
Gahboninho	0.528	0.782	0.027	0.001	0.038	0.002	0.004	0.929
HardHeaded	**0.638**	0.778	0.111	0.013	**0.133**	0.052	0.028	0.663
Nice Tit For Tat Agent	0.605	0.767	0.112	0.079	0.066	0.116	0.11	0.512
Agent K2	0.493	0.755	0.154	0.116	0.069	**0.203**	0.174	0.284
The Negotiator	0.591	0.751	0.080	0.036	0.071	0.077	0.051	0.685
IAMhaggler2011	0.377	0.748	**0.162**	**0.120**	0.074	**0.203**	**0.178**	0.263
BRAMAgent	0.578	0.740	0.115	0.075	0.085	0.148	0.104	0.472

Table 4.4: Overview of the experimental results. Bold text is used to emphasize the highest value, and underlined the lowest value.

Chapter 5

Analysis of Strategy Components

Abstract

An active topic within the field of automated negotiation is the development of negotiation strategies. A negotiation strategy is made up of different components and their complex interactions. In literature, negotiation strategies and their components have been investigated before; however little time has been spent analyzing the relationships between them. The aim of this paper is to investigate these relationships. We chose to divide a negotiation strategy into three main components: the ***B****idding strategy*, the ***O****pponent model* and the ***A****cceptance condition*. We examined the effects of combining different components and we determined the key components that contribute the most to the agent's performance. We found a significant difference in the amount of contribution each key component makes to the performance of a negotiation agent. We also established that combining better performing components does lead to better performing negotiation strategies. Furthermore, throughout our research we have found that the interactions between components play a large role as well in the performance of a negotiation strategy and that these interactions should not be overlooked.

As we plan to publish our findings in a conference this chapter is limited to an abstract of the to be published paper. The committee members graded the full chapter. A full version of the paper can be requested by contacting the authors:

Tim Baarslag (T.Baarslag@tudelft.nl)

Alexander Dirkzwager (asyd68@gmail.com)

Koen Hindriks (K.V.Hindriks@tudeflt.nl)

Catholijn Jonker (C.M.Jonker@tudelft.nl).

Chapter 6

Conclusion and Future Work

Negotiation is an everyday process which happens all around us. The more serious negotiations however take place in business settings where companies need to make deals with suppliers, customers and organizations in order to be profitable. Negotiations can be a very time consuming and costly process, but through the use of modern computers and negotiation support software it is possible to reduce these costs as well as the time needed to reach an agreement [87]. Negotiation support software has been shown to be very helpful in teaching people how to negotiate and automated negotiation agents are now at a point where they can conduct actual negotiations on behalf of human negotiators, sometimes even outperforming them [79, 80].

A fair amount of research has already been done investigating negotiation strategies in order to better understand them and make them more effective. Some researchers look at a negotiation strategy as a whole entity while others focus on certain components of a negotiation strategy. Despite all this research little time has been spent on investigating the relationships and interactions between the different components. Our work looks deeper into the workings of a negotiation strategy, creating a way to identify generic components of a negotiation strategy, which makes it possible to study the individual contribution of each component as well as the interrelations that take place.

This work introduces an agent architecture (BOA architecture), that distinguishes between three different components: the bidding strategy, the opponent model, and the acceptance condition, which when combined constitute a negotiation strategy. The main idea behind the BOA architecture is that we can identify several components in a negotiating agent, all of which can be optimized individually. By combining the best components, researchers can create more proficient negotiating agents.

We have shown in this work that the BOA architecture is not just a theoretical concept but one that can actually be applied, as many existing negotiation strategies could be fitted into our architecture. We were able to identify and classify their key components and have demonstrated that the original agents and their decoupled versions have identical behavior and similar performance, thus proving that it is indeed possible to identify generic components in most negotiation strategies. Our work has also shown that the advantages of fitting agents into the BOA architecture are threefold: first, it allows the study of the behavior and

performance of the individual components; second, by only altering one of the components at a time it allows for systematic exploration of the space of possible negotiation strategies; third, it aids the development of negotiation strategies by allowing researchers to focus on specific behaviors of a negotiation agent.

We have demonstrated with the BOA architecture, that it is possible to search the vast space of available negotiation strategies and identify better ones. By taking already existing (ANAC) agents and upgrading them by replacing their components with new or improved ones, we have succeeded in greatly improving the performance and the ranking of these (ANAC) agents. The BOA architecture also enables developers to divide the workload into bite size tasks, allowing them to focus on certain components and permitting the reuse of effective or well-known components. The BOA architecture makes it easier for developers to optimize and improve the agent strategy as well as the individual components, as was done with The Negotiator Reloaded. The Negotiator Reloaded was the first negotiation agent to be developed using the BOA architecture and placed third in the ANAC 2012 competition while also being named best performing agent in non-discounted domains. Another BOA agent and winner of ANAC2013, Fawkes, used an improved bidding strategy of the OMAC agent (which was a participant at ANAC2012).

Improving negotiation agents either by recombining components or optimizing their component parameters requires numerous negotiation simulations, where a vast number of strategies and BOA components need to be combined or multiple parameter variations need to be tested. To reduce the number of negotiations needed to test a set of negotiation strategies (as well as the amount of time required), a Multi Acceptance Condition (MAC) mechanism was integrated into the BOA architecture. MAC has the ability to run multiple acceptance conditions in parallel, allowing us to test multiple negotiation strategies where only the acceptance condition differ in a single negotiation.

With the BOA architecture and the library of components it has become possible to investigate the dynamics of the different BOA components, specifically looking at the contribution of these BOA components to the performance of a negotiation agent as well as the effects of combining the different components. With the ability to recombine different components, which can result in new strategies, we were able to examine the effects and determine whether replacing components with better performing ones would improve the overall performance of a negotiation agent. Analyzing all the possible combinations within the BOA repository is an immense task. Therefore to simplify this, we reduced the number of possible combinations by grouping the components into performance groups, this allowed us to compare the performance of different combination groups.

Our experiment proved that combining better components does indeed lead to a better strategy. By averaging the performance of each of the different performance combination groups we saw that the combination group High, High, High achieved the best average utility, performing significantly better (*one-tailed t-test*, $p < 0.01$) than all other performance groups. There is however possible overlap between the different performance combinations due to the large standard deviation within each of these combination groups, thereby showing that these groups are not statistically exclusive. Combining the best components does therefore not necessarily always guarantee the best single strategy. This was evident in our

results where a few strategies from the performance group High, Medium, High performed better than the best High, High, High strategy combination. This is caused by the interaction effects between the different components, where one component works better with another thus achieving a higher utility than if it would have been combined with an optimal performing component.

Even though combining optimal components does not guarantee the optimal strategy, it gives researchers insights which will enable them to systematically search the vast negotiation space for effective strategies by selecting components on the basis of their performance and their interactions.

From the rankings of these performance combinations we observed that the performance depended on which of the BOA components had the higher performing component, whether this was the bidding strategy, the acceptance condition or the opponent model. This suggested that altering the components does not necessarily provide equal gain. Investigating this further, we specifically looked at the combinations of two components and presented their performance in a contour plot. The contour plots showed islands that represent component combinations that perform better or worse than others in its surrounding area. Upon further inspection we were able to determine that these discrepancies in performance were due to the interaction between the components. Also, in all the possible two component combinations there was always one dominant component, which produced more utility change when altered than the others.

To establish the importance of each BOA component and its relative ranking we examined the contribution they made to the negotiation strategy. The contribution was determined by using η^2, which is an effect size measure looking at the variance a component produces.

From this investigation we were able to answer the question: *which of the components is the most important to the negotiation strategy?* We found significant difference (*one-tailed t-test*, $p < 0.01$) between the contributions of the three BOA components, where the bidding strategy is the most important one, followed by the acceptance condition and finally the opponent model.

Our findings give insight in what is the most important aspect in the design of a negotiation agent, by determining the contribution and importance of a component, we help researchers to improve their negotiation strategy as they can now take into account where and how the most gain in performance can be achieved.

Furthermore, the interactions between the components have shown to play a crucial role in the performance of a negotiation strategy. We found that combining the best components did not guarantee the best strategy due to the fact that there were interactions, where certain combinations worked better even though the components used might not have been the optimal ones. The islands presented in the contour plot, also indicated that the interactions affected the performance of the negotiation strategy. This observation was further corroborated as, the contributions of the component interactions were found to be relatively high (large η^2 values). This applied specifically to two interactions. The first is the interaction between the bidding strategy and the opponent model, as the contribution of the opponent model was found to be very dependent on how it is utilized by the bidding strategy. The other significant interaction was between the bidding strategy and the acceptance condi-

tion; these two components are complementary, i.e. one component can compensate for the shortcomings of the other and vice versa. Even though the contributions of the interactions are high and definitely noteworthy, they do not surpass the importance of the bidding strategy and the acceptance condition.

The essence of a negotiation is to work with the opponent in order to reach a mutually acceptable agreement. The characteristics of the opponent can therefore influence the agents' behavior and have a huge impact on the final outcome. This necessitated a closer inspection of how the different types of opponents influence the BOA components' contribution. When negotiating against opponents who concede constantly (based on time), it was possible to distinguish a trend, where the BOA components' contributions change as the opponent's concession rate changes. During a negotiation against a hardliner, an agent who concedes very little, the contribution of the bidding strategy was high and the contribution of the acceptance condition was found to be low. As the opponent's concession rate increased (the opponent makes larger and more frequent concessions) the contribution of the bidding strategy decreased and the contribution of the acceptance condition increased. This showed us that the bidding strategy matters most when negotiating against a hardliner, while the acceptance condition matters more when negotiating against a conceding opponent. This knowledge aids the development of better negotiation agents, which will need to take into account what type of opponent pool the agent will be negotiating in, whether it is competitive with a lot of hardliner agents or cooperative with many conceding agents.

This work has been able to divide negotiation strategies into components and determine their contribution to the agent's performance as well as the relationships between one another.

A possible direction for future work is to investigate the BOA components independently, e.g. finding methods to evaluate or optimize them by investigating the key processes. Research has already been done to find better opponent models [5, 10, 14], looking at how they obtain opponent information, but little research has looked at how to effectively use an opponent model. Other research has looked at the acceptance conditions [7, 47], but little time spent has been so far on the bidding strategy.

Currently the BOA architecture focuses on opponent models, predicting the opponent's preferences in order to estimate the utility of an arbitrary bid in the opponent's utility space. The BOA architecture could be extended to support more types of opponent models, e.g. opponent models that predict the opponent's strategy or the chances of accepting an agreement. Another interesting possibility that requires further study is the use of multiple components in one strategy. One possible approach would consist of running different components in parallel and using a recommendation system to choose the best component to use. This is similar to the strategy of Ilany and Gal [57]. Another approach that could be taken is to use genetic algorithms on the different BOA components in order to find the best negotiation strategy.

More future work can be done by investigating the influence of the negotiation domain on the BOA components' contribution.

We have seen that the negotiation domains produce high standard deviation for the con-

tribution of the BOA components, just like with the opponents. It is however still not clear what the effects are and how the domain influences the contribution of the BOA components. A better understanding of this aspect will help researchers gain insight and enable them to create agents that can be adapted to different negotiation domains.

More work can be done looking closer at the interactions for example, the three way interactions between the BOA components. Also throughout this paper we have seen that interactions affect the performance of an agent and these interactions have been identified and discussed. However, future work can focus on the kind of effects these interactions have and how we can use this to our advantage when designing an agent. Questions that could be addressed are for example: given a type of bidding strategy, which type of opponent model should be used and what kind of acceptance conditions works well with a particular bidding strategy type (hardliner, conceder, etc)? By understanding the effects of these interactions we will become capable of optimizing strategies and combining components more effectively which will then lead to better performing negotiating agents.

Bibliography

[1] Hervé Abdi and Lynne J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[2] Bo An and Victor Lesser. Yushu: a heuristic-based agent for automated negotiating competition. In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*, pages 145–149, Berlin, Heidelberg, 2012. Springer-Verlag.

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

[4] Reyhan Aydoğan, Tim Baarslag, Koen V. Hindriks, Catholijn M. Jonker, and Pınar Yolum. Heuristic-based approaches for CP-nets in negotiation. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 113–123. Springer Berlin / Heidelberg, 2013.

[5] Tim Baarslag. Designing an automated negotiator: Learning what to bid and when to stop. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '13, pages 1419–1420. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[6] Tim Baarslag, Katsuhide Fujita, Enrico H. Gerding, Koen Hindriks, Takayuki Ito, Nicholas R. Jennings, Catholijn Jonker, Sarit Kraus, Raz Lin, Valentin Robu, and Colin R. Williams. Evaluating practical negotiating agents: Results and analysis of the 2011 international competition. *Artificial Intelligence*, 198(0):73 – 103, 2013.

[7] Tim Baarslag, Mark Hendrikx, Koen Hindriks, and Catholijn Jonker. Measuring the performance of online opponent models in automated bilateral negotiation. In Michael Thielscher and Dongmo Zhang, editors, *AI 2012: Advances in Artificial Intelligence*, volume 7691 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2012.

[8] Tim Baarslag, Koen Hindriks, Mark Hendrikx, Alex Dirkzwager, and Catholijn Jonker. Decoupling negotiating agents to explore the space of negotiation strategies. In *Proceedings of The Fifth International Workshop on Agent-based Complex Automated Negotiations (ACAN 2012)*, 2012.

[9] Tim Baarslag, Koen Hindriks, Mark Hendrikx, and Catholijn Jonker. Predicting the performance of opponent models in automated negotiation. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 2013.

[10] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. Acceptance conditions in automated negotiation. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 95–111. Springer Berlin / Heidelberg, 2013.

[11] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. Effective acceptance conditions in real-time automated negotiation. *Decision Support Systems*, 2013.

[12] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. A tit for tat negotiation strategy for real-time bilateral negotiations. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 229–233. Springer Berlin Heidelberg, 2013.

[13] Tim Baarslag, Koen Hindriks, Catholijn M. Jonker, Sarit Kraus, and Raz Lin. The first automated negotiating agents competition (ANAC 2010). In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*, pages 113–135, Berlin, Heidelberg, 2012. Springer-Verlag.

[14] Tim Baarslag and Koen V. Hindriks. Accepting optimally in automated negotiation with incomplete information. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 715–722, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[15] C. Bartolini, C. Preist, and N. Jennings. A software framework for automated negotiation. *Software Engineering for Multi-Agent Systems III*, pages 213–235, 2005.

[16] C. Bartolini, C. Preist, and N.R. Jennings. A generic software framework for automated negotiation. In *First International Conference on Autonomous Agent and Multi-Agent Systems*. Citeseer, 2002.

[17] C. Beam and A. Segev. Automated negotiations: A survey of the state of the art. *Wirtschaftsinformatik*, 39(3):263–268, 1997.

[18] Mai Ben Adar, Nadav Sofy, and Avshalom Elimelech. Gahboninho: Strategy for balancing pressure and compromise in automated negotiation. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 205–208. Springer Berlin Heidelberg, 2013.

[19] K. Binmore and N. Vulkan. Applying game theory to automated negotiation. *Netnomics*, 1(1):1–9, 1999.

[20] Tibor Bosse and Catholijn M. Jonker. Human vs. computer behaviour in multi-issue negotiation. In *Proceedings of the Rational, Robust, and Secure Negotiation Mechanisms in Multi-Agent Systems*, RRS '05, pages 11–, Washington, DC, USA, 2005. IEEE Computer Society.

[21] S.J. Brams. *Negotiation Games: Applying game theory to bargaining and arbitration*, volume 2. Psychology Press, 2003.

[22] R. Carbonneau, G.E. Kersten, and R. Vahidov. Predicting opponent's moves in electronic negotiations using neural networks. *Expert Systems with Applications*, 34(2):1266–1273, 2008.

[23] Kalyan Chatterjee and William Samuelson. Bargaining under incomplete information. *Operations Research*, 31(5):835–851, 1983.

[24] J.H. Chen, R. Anane, K.M. Chao, and N. Godwin. Architecture of an agent-based negotiation mechanism. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 379–384. IEEE, 2002.

[25] Siqi Chen, Haitham Bou Ammar, Kurt Driessens, Karl Tuyls, and Gerhard Weiss. Automatic transfer between negotiation tasks. In *Proceedings of the Adaptive Learning Agents (ALA) at International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, Minnesota, USA, 2013.

[26] Siqi Chen, Haitham Bou Ammar, Karl Tuyls, and Gerhard Weiss. Transfer learning for bilateral multi issue negotiation. In *Proceedings of the Benelux Conference on Artificial Intelligence (BNAIC)*, Maastricht, The Netherlands, 2012.

[27] Siqi Chen and Gerhard Weiss. An efficient and adaptive approach to negotiation in complex environments. In *European Conference on Artificial Intelligence*, pages 228–233, 2012.

[28] Siqi Chen and Gerhard Weiss. An efficient automated negotiation strategy for complex environments. *Engineering Applications of Artificial Intelligence*, 2013.

[29] D. de Jonge and C. Sierra. Automated negotiation for package delivery. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2012 IEEE Sixth International Conference on*, pages 83–88, 2012.

[30] A.S.Y. Dirkzwager, M.J.C. Hendrikx, and J.R. Ruiter. The Negotiator: A dynamic strategy for bilateral negotiations with time-based discounts. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 217–221. Springer Berlin Heidelberg, 2013.

[31] M. Dumas, G. Governatori, A.H.M. Ter Hofstede, and P. Oaks. A formal approach to negotiating agents development. *Electronic Commerce Research and Applications*, 1(2):193–207, 2002.

[32] T. Eymann. Co-evolution of bargaining strategies in a decentralized multi-agent system. In *AAAI fall 2001 symposium on negotiation methods for autonomous cooperative systems*, pages 126–134, 2001.

[33] Angela Fabregues and Carles Sierra. Dipgame: a challenging negotiation testbed. *Engineering Applications of Artificial Intelligence*, 24(7):1137–1146, 2011.

[34] Angela Fabregues Vinent et al. Facing the challenge of automated negotiation with humans. 2012.

[35] P. Faratin, C. Sierra, and N.R. Jennings. Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence*, 142(2):205 – 237, 2002. International Conference on MultiAgent Systems 2000.

[36] Peyman Faratin, Carles Sierra, and Nick R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3-4):159 – 182, 1998. Multi-Agent Rationality.

[37] S. S. Fatima, M. Wooldridge, and N. R. Jennings. An agenda-based framework for multi-issue negotiation. *Artificial Intelligence*, 152(1):1–45, 2004.

[38] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. Multi-issue negotiation under time constraints. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 143–150, New York, NY, USA, 2002. ACM.

[39] J. Ferber. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 33. Addison-Wesley Reading, MA, 1999.

[40] Radmila Fishel, Maya Bercovitch, and Ya'akov(Kobi) Gal. Bram agent. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 213–216. Springer Berlin Heidelberg, 2013.

[41] Asaf Frieder and Gal Miller. Value model agent: A novel preference profiler for negotiation with agents. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 199–203. Springer Berlin Heidelberg, 2013.

[42] Markus M. Geipel and Gerhard Weiss. A generic framework for argumentation-based negotiation, 2007.

[43] E.H. Gerding, D.D.B. van Bragt, JA La Poutre, and Centrum voor Wiskunde en Informatica. *Scientific approaches and techniques for negotiation: a game theoretic and artificial intelligence perspective*. Citeseer, 2000.

[44] R.H. Guttman, A.G. Moukas, and P. Maes. Agent-mediated electronic commerce: a survey. *The Knowledge Engineering Review*, 13(02):147–159, 1998.

[45] Jianye Hao and Ho-Fung Leung. Abines: An adaptive bilateral negotiating strategy over multiple items. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '12, pages 95–102, Washington, DC, USA, 2012. IEEE Computer Society.

[46] Peter Henderson, Stephen Crouch, RobertJohn Walters, and Qinglai Ni. Comparison of some negotiation algorithms using a tournament-based approach. In JaimeG. Carbonell, Jrg Siekmann, Ryszard Kowalczyk, JörgP. Müller, Huaglory Tianfield, and Rainer Unland, editors, *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, volume 2592 of *Lecture Notes in Computer Science*, pages 137–150. Springer Berlin Heidelberg, 2003.

[47] M.J.C. Hendrikx. *Evaluating the Quality of Opponent Models in Automated Bilateral Negotiations*. PhD thesis, Delft University of Technology, 2012.

[48] Koen Hindriks, Catholijn Jonker, and Dmytro Tykhonov. Analysis of negotiation dynamics. In Matthias Klusch, Koen V. Hindriks, Mike P. Papazoglou, and Leon Sterling, editors, *Cooperative Information Agents XI*, volume 4676 of *Lecture Notes in Computer Science*, pages 27–35. Springer Berlin Heidelberg, 2007.

[49] Koen Hindriks, Catholijn M. Jonker, Sarit Kraus, Raz Lin, and Dmytro Tykhonov. Genius: negotiation environment for heterogeneous agents. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 1397–1398, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[50] Koen Hindriks, Catholijn M. Jonker, and Dmytro Tykhonov. Negotiation dynamics: Analysis, concession tactics, and outcomes. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IAT '07, pages 427–433, Washington, DC, USA, 2007. IEEE Computer Society.

[51] Koen V. Hindriks, Catholijn Jonker, and Dmytro Tykhonov. Towards an open negotiation architecture for heterogeneous agents. In Matthias Klusch, Michal Pechoucek, and Axel Polleres, editors, *Cooperative Information Agents XII*, volume 5180 of *Lecture Notes in Computer Science*, pages 264–279. Springer Berlin Heidelberg, 2008.

[52] Koen V. Hindriks and Catholijn M. Jonker. Creating human-machine synergy in negotiation support systems: towards the pocket negotiator. In *Proceedings of the 1st*

International Working Conference on Human Factors and Computational Models in Negotiation, HuCom '08, pages 47–54, New York, NY, USA, 2009. ACM.

[53] Koen V. Hindriks and Dmytro Tykhonov. Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1*, AAMAS '08, pages 331–338, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[54] Koen V. Hindriks and Dmytro Tykhonov. Towards a quality assessment method for learning preference profiles in negotiation. In Wolfgang Ketter, Han Poutré, Norman Sadeh, Onn Shehory, and William Walsh, editors, *Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis*, volume 44 of *Lecture Notes in Business Information Processing*, pages 46–59. Springer Berlin Heidelberg, 2010.

[55] J. Hodgson and Institute of Management (Great Britain). *Thinking on your feet in negotiations*. Pitman Great Britain, UK, 1996.

[56] Chongming Hou. Predicting agents tactics in automated negotiation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 127–133. IEEE Computer Society, Published by the IEEE Computer Society, 2004.

[57] Litan Ilany and Ya'akov Gal. Algorithm selection in bilateral negotiation (accepted). In *Proceedings of The Sixth International Workshop on Agent-based Complex Automated Negotiations (ACAN 2013)*, 2013.

[58] H. Jazayeriy, M. Azmi-Murad, M.N. Sulaiman, and N.I. Udzir. A review on soft computing techniques in automated negotiation. *Scientific Research and Essays*, 6(24):5100–5106, 2011.

[59] Nick Jennings and Michael Wooldridge. Software agents. *IEE review*, 42(1):17–20, 1996.

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

[61] Ian Jolliffe. *Principal Component Analysis*. John Wiley & Sons, Ltd, 2005.

[62] Catholijn Jonker, Valentin Robu, and Jan Treur. An agent architecture for multi-attribute negotiation using incomplete preference information. *Autonomous Agents and Multi-Agent Systems*, 15:221–252, 2007.

[63] C.M. Jonker and J. Treur. An agent architecture for multi-attribute negotiation. In *Proceedings of IJCAI'01*, pages 1195–1201, 2001.

[64] Bart Kamphorst, Arlette Wissen, and Virginia Dignum. Incorporating bdi agents into human-agent decision making research. In Huib Aldewereld, Virginia Dignum, and Gauthier Picard, editors, *Engineering Societies in the Agents World X*, volume 5881 of *Lecture Notes in Computer Science*, pages 84–97. Springer Berlin Heidelberg, 2009.

[65] Shogo Kawaguchi, Katsuhide Fujita, and Takayuki Ito. Compromising strategy based on estimated maximum utility for automated negotiation agents competition (ANAC-10). In KishanG. Mehrotra, ChilukuriK. Mohan, JaeC. Oh, PramodK. Varshney, and Moonis Ali, editors, *Modern Approaches in Applied Intelligence*, volume 6704 of *Lecture Notes in Computer Science*, pages 501–510. Springer Berlin Heidelberg, 2011.

[66] Shogo Kawaguchi, Katsuhide Fujita, and Takayuki Ito. Agentk2: Compromising strategy based on estimated maximum utility for automated negotiating agents. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 235–241. Springer Berlin Heidelberg, 2013.

[67] G. E. Kersten and S. J. Noronha. Www-based negotiation support: design, implementation, and use. *Decision Support Systems*, 25(2):135–154, 1999.

[68] G.E. Kersten and G. Lo. Negotiation support systems and software agents in e-business negotiations. In *The First International Conference on Electronic Business*, pages 19–21. Citeseer, 2001.

[69] G.E. Kersten and G. Lo. Aspire: an integrated negotiation support system and software agents for e-business negotiation. *International Journal of Internet and Enterprise Management*, 1(3):293–315, 2003.

[70] Gregory E. Kersten and Grant Zhang. Mining inspire data for the determinants of successful internet negotiations. *InterNeg Research Papers INR 04/01 Central European Journal of OperationalResearch*, 11(3):297–316, 2003.

[71] Mark Klein. From problems to protocols: Towards a negotiation handbook. *Available at SSRN*, 2013.

[72] R. Kowalczyk, M. Ulieru, and R. Unland. Integrating mobile and intelligent agents in advanced e-commerce: A survey. *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, pages 295–313, 2010.

[73] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1-2):79–97, 1997.

[74] S. Kraus. Automated negotiation and decision making in multiagent environments. *Multi-agent systems and applications*, 1:150, 2001.

[75] R. Krovi, A.C. Graesser, and William E. Pracht. Agent behaviors in virtual negotiation environments. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 29(1):15–25, 1999.

[76] Guoming Lai, Katia Sycara, and Cuihong Li. A decentralized model for automated multi-attribute negotiations with incomplete information and general utility functions. *Multiagent and Grid Systems*, 4(1):45–65, 2008.

[77] R.Y.K. Lau, M. Tang, O. Wong, S.W. Milliner, and Y.P.P. Chen. An evolutionary learning approach for adaptive negotiation agents. *International journal of intelligent systems*, 21(1):41–72, 2005.

[78] C. Li, J. Giampapa, and K. Sycara. A review of research literature on bilateral negotiations. Technical report, Robotics Institute, Pittsburgh, PA, November 2003.

[79] R. Lin, Y. Oshrat, and S. Kraus. Investigating the benefits of automated negotiations in enhancing people's negotiation skills. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 345–352. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[80] Raz Lin, Ya'akov (Kobi) Gal, Sarit Kraus, and Yaniv Mazliah. Training with automated agents improves people's behavior in negotiation and coordination tasks. *Decision Support Systems*, 2013.

[81] Raz Lin, Yehoshua Gev, and Sarit Kraus. Animed*: An automated animated mediator for facilitating negotiation with people.

[82] Raz Lin, Yehoshua Gev, and Sarit Kraus. Bridging the gap: Face-to-face negotiations with an automated mediator. *IEEE Intelligent Systems*, 26(6):40–47, 2011.

[83] Raz Lin, Yehoshua Gev, and Sarit Kraus. Facilitating better negotiation solutions using animed. *Proc. of Agent-based Complex Automated Negotiations*, pages 64–70, 2011.

[84] Raz Lin and Sarit Kraus. From research to practice: Automated negotiations with people. In Antonio Krüger and Tsvi Kuflik, editors, *Ubiquitous Display Environments*, Cognitive Technologies, pages 195–212. Springer Berlin Heidelberg, 2012.

[85] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, 2012.

[86] Zvi Aviad Livne. The role of time in negotiation. Subitted for PhD in Philosophy.

[87] Alessio Lomuscio, Michael Wooldridge, and Nicholas Jennings. A classification scheme for negotiation in electronic commerce. In Frank Dignum and Carles Sierra, editors, *Agent Mediated Electronic Commerce*, volume 1991 of *Lecture Notes in Computer Science*, pages 19–33. Springer Berlin / Heidelberg, 2001.

[88] M. Lopez-Carmona, I. Marsa-Maestre, E. de la Hoz, and J. Velasco. Negoexplorer: A region-based recursive approach to bilateral multi-attribute negotiation. *Principles of Practice in Multi-Agent Systems*, pages 261–275, 2009.

[89] N. Matos, C. Sierra, and N.R. Jennings. Determining successful negotiation strategies: an evolutionary approach. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 182–189, 1998.

[90] Oskar Morgenstern. Ingolf ståhl: Bargaining theory. *The Swedish Journal of Economics*, 75(4):410–413, 1973.

[91] John F. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

[92] J. Niu, K. Cai, P. McBurney, and S. Parsons. An analysis of entries in the first tac market design competition. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 2, pages 431–437. IEEE, 2008.

[93] P. Noriega and C. Sierra. *Agent Mediated Electronic Commerce: First International Workshop on Agent Mediated Electronic Trading, AMET'98, Minneapolis, MN, USA, May 10th, 1998 Selected Papers*, volume 1571. Springer, 1999.

[94] Hyacinth S Nwana et al. Software agents: An overview. *Knowledge Engineering Review*, 11(3):205–244, 1996.

[95] Jim R. Oliver. A machine-learning approach to automated negotiation and prospects for electronic commerce. *Journal of Management Information Systems*, 13:83–112, December 1996.

[96] Jim R. Oliver. On artificial agents for negotiation in electronic commerce. In *Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences*, volume 4, pages 337–346 vol.4, 1996.

[97] Jim R. Oliver. A machine-learning approach to automated negotiation and prospects for electronic commerce. *Journal of Management Information Systems*, 13:83–112, 1997.

[98] Martin J. Osborne and Ariel Rubinstein. *Bargaining and Markets (Economic Theory, Econometrics, and Mathematical Economics)*. Academic Press, April 1990.

[99] Y. Oshrat, R. Lin, and S. Kraus. Facing the challenge of human-agent negotiations via effective general opponent modeling. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 377–384. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[100] S. Parsons, C. Sierra, and N. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and computation*, 8(3):261–292, 1998.

[101] A. Pommeranz, W.P. Brinkman, P. Wiggers, J. Broekens, and C.M. Jonker. Design guidelines for negotiation support systems: an expert perspective using scenarios. In *European Conference on Cognitive Ergonomics: Designing beyond the Product—Understanding Activity and User Experience in Ubiquitous Environments*, page 27. VTT Technical Research Centre of Finland, 2009.

[102] C. Preist, C. Bartolini, and I. Phillips. Algorithm design for agents which participate in multiple simultaneous auctions. *Agent-Mediated Electronic Commerce III*, pages 139–154, 2001.

[103] Yong quan Liang and Yong Yuan. Co-evolutionary stability in the alternating-offer negotiation. In *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*, pages 1176–1180, 2008.

[104] Iyad Rahwan, Sarvapalic Ramchurn, Nicholas R. Jennings, Peter McBurney, Simon Parsons, and Liz Sonenberg. Argumentation-based negotiation. *The Knowledge Engineering Review*, 18(04):343–375, 2003.

[105] H. Raiffa. *The art and science of negotiation: How to resolve conflicts and get the best out of bargaining*. Harvard University Press, Cambridge, MA, 1982.

[106] F. Ramezani, N. Ghasem-Aghaee, and M. Kazemifard. Modeling of emotional-social negotiator agents. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 42–46, 2011.

[107] A. Rangaswamy and G.R. Shell. Using computers to realize joint gains in negotiations: toward an electronic bargaining table. *Management Science*, 43(8):1147–1163, 1997.

[108] Raquel Ros and Carles Sierra. A negotiation meta strategy combining trade-off and concession moves. *Autonomous Agents and Multi-Agent Systems*, 12:163–181, 2006.

[109] J.Z. Rubin, B.R. Brown, and M. Deutsch. *The social psychology of bargaining and negotiation*. Academic press, 1975.

[110] Ariel Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, 1982.

[111] Victor Sanchez-Anguix, Reyhan Aydogan, Vicente Julian, and Catholijn M. Jonker. Analysis of intra-team strategies for teams negotiating against competitor, matchers, and conceders. In *The Fifth International Workshop on Agent-based Complex Automated Negotiations (ACAN 2012)*, Valencia, Spain, 2012.

[112] Tuomas Sandholm and Nir Vulkan. Bargaining with deadlines. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference*, AAAI '99/IAAI '99, pages 44–51, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[113] Jr. Scott, John T. Factor analysis and regression. *Econometrica*, 34(3):pp. 552–562, 1966.

[114] Liviu Dan Serban, Gheorghe Cosmin Silaghi, and Cristian Marius Litan. Agent FSEGA - time constrained reasoning model for bilateral multi-issue negotiations. In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*, pages 159–165, Berlin, Heidelberg, 2012. Springer-Verlag.

[115] Ashwini Sharma. *A study on adoption of e-Negotiation through software agents*. PhD thesis, 2013.

[116] G.C. Silaghi, L.D. Serban, and C.M. Litan. A framework for building intelligent sla negotiation strategies under time constraints. In *Proceedings of Economics of Grids, Clouds, Systems, and Services: 7th International Workshop*, volume 6296, page 48. Springer-Verlag New York Inc, 2010.

[117] G.C. Silaghi, L.D. Serban, and C.M. Litan. A framework for building intelligent SLA negotiation strategies under time constraints. In *Proceedings of Economics of Grids, Clouds, Systems, and Services: 7th International Workshop*, volume 6296, page 48. Springer-Verlag New York Inc, 2010.

[118] Gheorghe Cosmin Silaghi, Liviu Dan Serban, and Cristian Marius Litan. A time-constrained SLA negotiation strategy in competitive computational grids. *Future Generation Computer Systems*, 28(8):1303–1315, 2012.

[119] Kwang-Mong Sim. Agent-based cloud commerce. In *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on*, pages 717–721, 2009.

[120] V. Snchez-Anguix. *Complex Negotiations in Multi-Agent Systems*. PhD thesis, Departament de Sistemes Informàtics i Computació, Universitat Politècnica de València, 2013.

[121] LiviuDan Srban, CristinaMaria tefanache, GheorgheCosmin Silaghi, and Cristian-Marius Litan. A qualitative ascending protocol for multi-issue one-to-many negotiations. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 143–159. Springer Berlin Heidelberg, 2013.

[122] John Sutton. Non-cooperative bargaining theory: An introduction. *The Review of Economic Studies*, 53(5):709–724, 1986.

[123] Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[124] Thomas M Tripp and Harris Sondak. An evaluation of dependent variables in experimental negotiation studies: Impasse rates and pareto efficiency. *Organizational Behavior and Human Decision Processes*, 51(2):273–295, 1992.

[125] Thomas M Tripp and Harris Sondak. An evaluation of dependent variables in experimental negotiation studies: Impasse rates and pareto efficiency. *Organizational Behavior and Human Decision Processes*, 51(2):273 – 295, 1992. ¡ce:title¿Decision Processes in Negotiation¡/ce:title¿.

[126] M. T. Tu, E. Wolff, and W. Lamersdorf. Genetic algorithms for automated negotiations: A fsm-based application approach. In *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, DEXA '00, pages 1029–, Washington, DC, USA, 2000. IEEE Computer Society.

[127] Niels van Galen Last. Agent smith: Opponent model estimation in bilateral multi-issue negotiation. In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*, pages 167–174, Berlin, Heidelberg, 2012. Springer-Verlag.

[128] Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh. Hardheaded. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 223–227. Springer Berlin Heidelberg, 2013.

[129] Colin R. Williams. *Practical Strategies for Agent-Based Negotiation in Complex Environments*. PhD thesis, University of Southampton, December 2012.

[130] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. Iamhaggler: A negotiation agent for complex environments. In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*, pages 151–158, Berlin, Heidelberg, 2012. Springer-Verlag.

[131] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. Iamhaggler: A negotiation agent for complex environments. *This volume*, pages 151–158, 2012.

[132] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. Negotiating concurrently with unknown opponents in complex, real-time domains. In *20th European Conference on Artificial Intelligence*, volume 242, pages 834–839, August 2012.

[133] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. Iamhaggler2011: A gaussian process regression based negotiation agent. In Takayuki Ito, Minjie Zhang, Valentin Robu, and Tokuro Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 209–212. Springer Berlin Heidelberg, 2013.

[134] D. Zeng and K. Sycara. Bayesian learning in negotiation. *International Journal of Human Computer Systems*, 48:125–141, 1998.

Appendix A

BOA Architecture Class Diagram

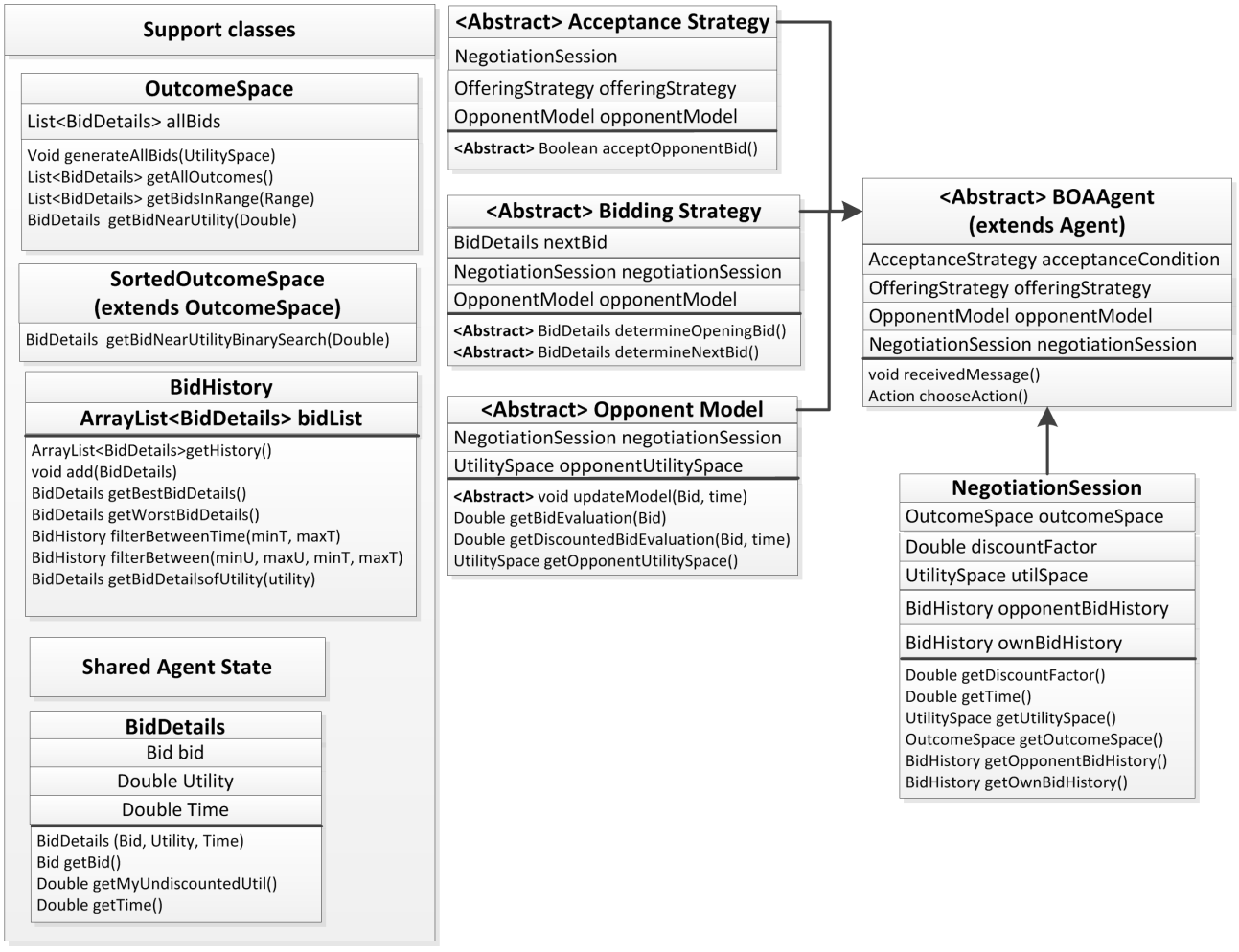

Figure A.1: Higher level class diagram of the BOA architecture.

Appendix B

BOA Component Repository

BOA Components		
Bidding Strategy	**Opponent Model**	**Acceptance Condition**
IAMhaggler2010	Bayesian Model	$AC_{IAMhaggler2010}$
IAMCrazyhaggler	IAMhaggler Bayesian Model	$AC_{IAMCrazyhaggler}$
Agent K	Agent LG Model	AC_{AgentK}
Agent Smith	AgentX Frequency Model	$AC_{AgentSmith}$
Nozomi	CUHK Frequency Model	AC_{Nozomi}
Yushu	NoModel	AC_{Yushu}
Agent FSEGA	FSEGA Bayesian Model	$AC_{AgentFSEGA}$
Agent K2	HardHeaded Frequency Model	AC_{AgentK}
BRAM Agent	InoxAgent Model	$AC_{BRAMAgent}$
Gahboninho	Nash Frequency Model	$AC_{Gahbininho}$
HardHeaded	Opposite Model	$AC_{HardHeaded}$
IAMhaggler2011	Perfect IAMhaggler Bayesian Model	$AC_{IAMhaggler2011}$
Nice Tit For Tat	Perfect Model	$AC_{NiceTitForTat}$
TheNegotiator	Perfect Scalable Bayesian Model	$AC_{TheNegotiator}$
ValueModel Agent	Scalable Bayesian Model	$AC_{ValueModelAgent}$
CUHK	Smith Frequency Model	$AC_{AgentLG}$
Agent LG	TheFwakes Model	$AC_{AgentMR}$
Agent MR	Uniform Model	$AC_{BRAMAgent2}$
BRAM Agent2	Worst Model	$AC_{IAMhaggelr2012}$
IAMhaggler2012		$AC_{OMACAgent}$
OMAC Agent		$AC_{TheNegotiator\_Reloaded}$
TheNegotiator_Reloaded		AC_{Fawkes}
Fawkes		$AC_{InoxAgent\_OneIssue}$
Inox Agent_OneIssue		$AC_{InoxAgent}$
Inox Agent		$AC_{Previous}(a, b)$
Time Dependent		$AC_{Next}(a, b)$
Zero Intelligence		$AC_{Time}(t)$
Offer Decreasing		$AC_{Constant}(c)$
		$AC_{ReservationValue}(r)$
		$AC_{OptimalStopping}$
		$AC_{Combi}(T, Max^w)$

Table B.1: Overview of all the BOA components.