Decoupling Negotiating Agents to Explore the Space of Negotiation Strategies

Chapter · January 2014

DOI: 10.1007/978-4-431-54758-7_4

5 authors, including:

Tim Baarslag
Centrum Wiskunde & Informatica
81 PUBLICATIONS **994** CITATIONS

SEE PROFILE

Koen V. Hindriks
Vrije Universiteit Amsterdam
245 PUBLICATIONS **3,690** CITATIONS

SEE PROFILE

Mark Hendrikx
9 PUBLICATIONS **357** CITATIONS

SEE PROFILE

Catholijn M. Jonker
Delft University of Technology
562 PUBLICATIONS **7,285** CITATIONS

SEE PROFILE

Decoupling Negotiating Agents to Explore the Space of Negotiation Strategies*

Tim Baarslag, Koen Hindriks, Mark Hendrikx, Alexander Dirkzwager, and Catholijn Jonker

Interactive Intelligence Group, Delft University of Technology,
Mekelweg 4, Delft, The Netherlands
{T.Baarslag,K.V.Hindriks,M.J.C.Hendrikx,A.S.Y.Dirkzwager,C.M.Jonker}@tudelft.nl

Abstract. Every year, automated negotiation agents are improving on various domains. However, given a set of negotiation agents, current methods allow to determine which strategy is best in terms of utility, but not so much the reason of success. In order to study the performance of the individual elements of a negotiation strategy, we introduce an architecture that distinguishes three components which together constitute a negotiation strategy: the bidding strategy, the opponent model, and the acceptance condition. Our contribution to the field of bilateral negotiation is threefold: first, we show that existing state of the art agents are compatible with this architecture; second, as an application of our architecture, we systematically explore the space of possible strategies by recombining different strategy components; finally, we briefly review how the BOA architecture has been recently applied to evaluate the performance of strategy components and create novel negotiation strategies that outperform the state of the art.

Keywords: Automated bilateral negotiation, BOA architecture, bidding strategy, opponent model, acceptance condition, component-based

1 Introduction

In recent years, many new automated negotiation agents have been developed in the search for an effective, generic automated negotiator. There is now a large body of negotiation strategies available, and with the emergence of the International Automated Negotiating Agents Competition (ANAC) [3, 8], new strategies are generated on a yearly basis.

While methods exist to determine the best negotiation agent given a set of agents [3, 8], we still do not know which type of agent is most effective in general, and especially why. It is impossible to exhaustively search the large (in fact, infinite) space of negotiation strategies; therefore, there is a need for a systematic way of searching this space for effective candidates.

* This is an extension of research presented at *The Fifth International Workshop on Agent-based Complex Automated Negotiations (ACAN 2012)*.

Many of the sophisticated agent strategies that currently exist are comprised of a fixed set of modules. Generally, a distinction can be made between three different modules: one module that decides whether the opponent's bid is acceptable; one that decides what set of bids could be proposed next; and finally, one that tries to guess the opponent's preferences and takes this into account when selecting an offer to send out. The negotiation strategy is a result of the complex interaction between these components, of which the individual performance may vary significantly. For instance, an agent may contain a module that predicts the opponent's preferences very well, but utility-wise, the agent may still perform badly because it concedes far too quickly.

This entails that overall performance measures, such as average utility obtained in a tournament, make it hard to pinpoint which components of an agent work well. To date, no efficient method exists to identify to which of the components the success of a negotiating agent can be attributed. Finding such a method would allow to develop better negotiation strategies, resulting in better agreements; the idea being that well-performing components together will constitute a well-performing agent.

To tackle this problem, we propose to analyze three components of the agent design separately. We show that most of the currently existing negotiating agents can be fitted into the so-called *BOA architecture* by putting together three main components in a particular way; namely: a *Bidding strategy*, an *Opponent model*, and an *Acceptance condition*. We support this claim by re-implementing, among others, the ANAC agents to fit into our architecture. Furthermore, we show 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 individual components; second, it allows to systematically explore the space of possible negotiation strategies; third, the identification of unique interacting components simplifies the creation of new negotiation strategies.

Finally, we demonstrate the value of our architecture by assembling, from already existing components, new negotiating agents that perform better than the agents from which they are created. This shows that by recombining the best performing components, the BOA architecture can yield better performing agents.

The remainder of this paper is organized as follows. Section 2 discusses the work related to ours. In Section 3, the BOA agent architecture is introduced, and we outline a research agenda on how to employ it. Section 4 provides evidence that many of the currently existing agents fit into the BOA architecture, and discusses challenges in decoupling existing negotiation strategies. Section 5 shows how the BOA architecture has been applied in education and research. Finally, in Section 6 we discuss lessons learned and provide directions for future work.

2 Related Work

Since this paper introduces an component-based architecture, we have surveyed literature that investigates and evaluates such components. There are three categories of related work: literature detailing the architecture of a negotiating agent's strategy; work that discusses and compares the performance of components of a negotiation strategy; and finally, literature that explores and combines a set of negotiation strategies to find an optimal strategy.

2.1 Architecture of Negotiation Strategies

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. [10] and Dumas et al. [16] treat the negotiation strategy as a singular component. However, there are some notable exceptions.

Jonker et al. [27] 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. There are some similarities between the two architectures; for example, the utility planning and attribute planning component correspond to the bidding strategy component in our architecture. In contrast to our work however, Jonker et al. focus on tactics for finding a counter offer and do not discuss acceptance conditions.

Ashri et al. [2] introduce a general architecture for negotiation agents, discussing components that resemble our architecture; components such as a proposal evaluator and response generator resemble an acceptance condition and bidding strategy respectively. However, the negotiation strategy is described from a BDI-agent perspective (in terms of motivation and mental attitudes).

Hindriks et al. [23] introduce an architecture for negotiation agents in combination with a negotiation system architecture. Parts of the agent architecture correspond to our architecture presented below, but their focus is primarily on how the agent framework can be integrated into a larger system.

2.2 Components of Negotiation Strategy

Evaluation of the performance of components is important to gain understanding of the performance of a negotiation strategy, and to find new, better strategies.

The notion of an opponent model as a component of a negotiation strategy has been discussed by various authors in different forms, including models that estimate the reservation value [42], the (partial) preference profile [24], the opponent's acceptance of offers [33], and the opponent's next move [13]. To our knowledge, there is limited work in which the performance of different opponent models is compared. Two examples are the work by Papaioannou et al. [34], who evaluate a set of techniques that predict the opponent's strategy in terms of resulting performance gain, as well as computational complexity; and Baarslag et al. [4, 5], who compare the performance and accuracy of preference modeling

techniques. The BOA architecture focuses on opponent models which estimate the (partial) preference profile, because most existing available implementations fit in this category; however, in principle, our architecture can accommodate for the other types of opponent models as well.

Regarding acceptance conditions, the performance of a set of acceptance strategies that depend on parameters such as time and utility thresholds have been analyzed in [6].

Although we are not the first to identify the BOA components in a negotiation strategy, our approach seems to be unique in the sense that we vary all of these components at the same time, thereby creating new negotiation strategies, and improving the state of the art in doing so.

2.3 Negotiation Strategy Space Exploration

Various authors have aimed to explore the automated negotiation strategy space by combining a set of negotiation strategies.

Faratin et al. [18] analyze the performance of pure negotiation tactics on single issue domains in a bilateral negotiation setting. The decision function of the pure tactic is then treated as a component around which the full strategy is built. While they discuss how tactics can be linearly combined, the performance of the combined tactics is not analyzed.

Matos et al. [32] employ a set of baseline negotiation strategies that are time dependent, resource dependent, and behavior dependent [18], all with varying parameters. The negotiation strategies are encoded as chromosomes and combined linearly, after which they are utilized by a genetic algorithm to analyze the effectiveness of the strategies. The fitness of an agent is its score in a negotiation competition. This approach analyzes acceptance criteria that only specify a utility interval of acceptable values, and hence do not take time into account; furthermore, the agents do not employ explicit opponent modeling.

Eymann [17] also uses genetic algorithms with more complex negotiating strategies, evolving six parameters that influence the bidding strategy. The genetic algorithm uses the current negotiation strategy of the agent and the opponent strategy with the highest average income to create a new strategy, similar to other genetic algorithm approaches (see Beam and Segev [11] for a discussion of genetic algorithms in automated negotiation). The genetic algorithm approach mainly treats the negotiation strategy optimization as a search problem in which the parameters of a small set of strategies are tuned by a genetic algorithm. We analyze a more complex space of newly developed negotiation strategies in our approach, as our pool of surveyed negotiation strategies consists of strategies introduced in the ANAC competition [3, 8], as well as the strategies discussed by Faratin et al. [18]. Furthermore, each strategy consists of components that can have parameters themselves.

Finally, Ilany and Gal [26] take the approach of selecting the best strategy from a predefined set of agents, based on the characteristics of a domain. The difference with our work is that they combine whole strategies, whereas the BOA architecture combines the components of strategies. Our contribution is to define

and implement an architecture that allows to easily vary all main components of a negotiating agent.

3 The BOA Agent Architecture

In the last decade, many different negotiation strategies have been introduced in the pursuit of a versatile and effective automated negotiator (see related work in Section 2). Current work often focuses on optimizing the negotiation strategy as a whole. We propose to direct our attention to a component-based approach, especially now that we have access to a large repository of mutually comparable negotiation strategies due to ANAC. This approach has several advantages:

1. Given measures for the effectiveness of the individual components of a negotiation strategy, we are able to pinpoint the most promising components, which gives insight into the reasons for success of the strategy;
2. Focusing on the most effective components helps to systematically search the space of negotiation strategies by recombining them into new strategies.

We make a distinction between two types of components in the sections below: elements that are part of the agent's environment, and components that are part of the agent itself.

3.1 Negotiation Environment

We employ the same *negotiation environment* as in [3, 8, 31]; that is, we consider bilateral automated negotiations, where the interaction between the two negotiating parties is regulated by the alternating-offers protocol [35]. The agents negotiate over a set of issues, as defined by the negotiation *domain*, which holds the information of possible bids, negotiation constraints, and the discount factor. The negotiation happens in real time, and the agents are required to reach an agreement (i.e., one of them has to accept) before the deadline is reached. The timing of acceptance is particularly important because the utility may be discounted, that is: the utility of an agreement may decrease over time.

In addition to the domain, both parties also have 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 player only has access to its own utility function, and is unaware of the opponent's preferences. The player can attempt to learn this during the negotiation encounter by analyzing the *bidding history*, using an opponent modeling technique.

3.2 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*, possibly an *opponent model*, and an *acceptance*

condition (BOA). The elements of a BOA agent are visualized in Figure 1. In order to fit an agent into the BOA architecture, it should be possible to distinguish these components in 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.

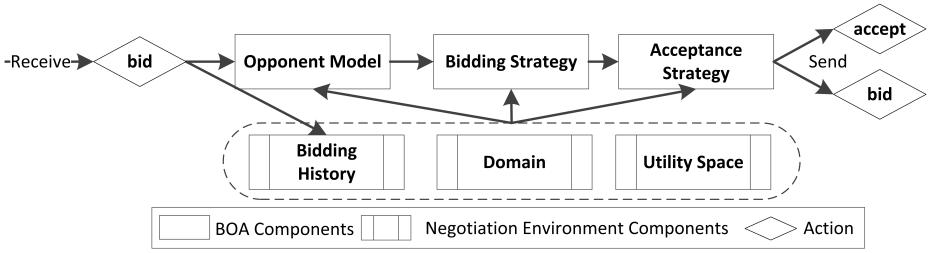

Fig. 1. The BOA architecture negotiation flow.

1. **Bidding strategy**. A bidding strategy is a mapping from a negotiation trace to a bid. The bidding strategy determines the appropriate concessions to be made, depending on factors such as the opponent's negotiation trace, a target threshold, time, discount factor, etc. The bidding strategy can consult the opponent model by passing one or multiple bids to see how they compare within the estimated opponent's utility space.
   Input: *opponent utility of bids, negotiation trace.*
   Output: *provisional upcoming bid.*
2. **Opponent model**. An opponent model is a learning technique that constructs a model of the opponent's preferences. In our approach, the opponent model should be able to estimate the opponent's utility of any given bid.
   Input: *set of possible bids, negotiation trace.*
   Output: *estimated opponent utility of a set of bids.*
3. **Acceptance Condition**. The acceptance condition determines whether the bid that the opponent presents is acceptable.
   Input: *provisional upcoming bid, negotiation trace.*
   Output: *send accept, or send out the upcoming bid.*

The components interact in the following way (the full process is visualized in Figure 1): 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 is used, maximizing the information known about the environment and opponent.

Given the opponent bid, the *bidding strategy* determines the counter offer by first generating a set of bids with a similar preference for the agent. The *bidding strategy* uses the *opponent model* (if present) to select a bid from this set by taking the opponent's utility into account.

Finally, the *acceptance condition* decides whether the opponent's action should be accepted. If the opponent's bid is not accepted by the acceptance condition,

then the bid generated by the bidding strategy 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 acceptance conditions that depend on the utility of the counter bid that is ready to be sent out. This method is widely used in existing agents [6]. Such acceptance mechanisms can make a more informed decision by postponing their decision on accepting until the last step; therefore, and given our aim to incorporate as many agent designs as possible, we adopt this approach in our architecture.

3.3 Employing the BOA Architecture

We have implemented the BOA architecture as an extension of the GENIUS framework [31]. GENIUS 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 negotiation. The framework was developed as a research tool to facilitate the design of negotiation strategies and to aid in the evaluation of negotiation algorithms. It provides a flexible and easy to use environment for implementing agents and negotiation strategies as well as running negotiations. GENIUS can further aid the development of a negotiation agents by acting as an analytical toolbox, providing a variety of tools to analyze the negotiation agents performance, based on the outcome and dynamics of the negotiation. 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 2. 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.

The framework enables us to follow at least two approaches: first of all, it allows us to independently analyze the components of every negotiation strategy that fits in to our architecture. For example, by re-implementing the ANAC agents in the BOA architecture, it becomes possible to compare the accuracy of

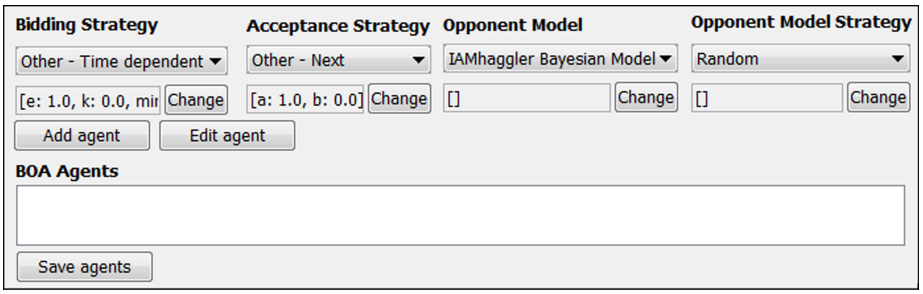

Fig. 2. The BOA architecture GUI.

all ANAC opponent models, and to pinpoint the best opponent model among them. Following this approach, we are able to identify a categories of opponent models that outperform others [4, 5]; naturally, this helps to build better agents in the future.

Secondly, we can proceed to *mix* different BOA components, e.g.: replace the opponent model of the runner-up of ANAC by a different opponent model and then examine whether this makes a difference in placement. Such a procedure enables us to assess the reasons for an agent's success, and makes it possible to systematically search for an effective automated negotiator.

The first part of the approach gives insight in what components are best in isolation; the second part gives us understanding of their influence on the agent as a whole. At the same time, both approaches raise some key theoretical questions, such as:

1. Can the BOA components be identified in all, or at least most, current negotiating agents?
2. How do we measure the performance of the components? Can a single best component be identified, or does this strongly depend on the other components?
3. If the individual components perform better than others (with respect to some performance measure), does combining them in an agent also improve the agent's performance?

In this work we do not aim to fully answer all of the above questions; instead, we outline a research agenda, and introduce the BOA architecture as a tool that can be used towards answering these questions.

Nonetheless, in the next section, we will provide empirical support for an affirmative answer to the first theoretical question: indeed, in many cases the components of the BOA architecture can be identified in current agents, and we will also provide reasons for when this is not the case.

The answer to the second question depends on the component under consideration: for an opponent model, it is straightforward to measure its effectiveness [4, 5, 25]: the closer the opponent model is to the actual profile of the opponent, the better it is. The performance of the other two components of the BOA architecture is better measured in terms of utility obtained in negotiation (as has been done for acceptance strategies in [6]), as there seems no clear alternative method to define the effectiveness of the acceptance condition or bidding strategy in isolation. In any case, the BOA architecture can be used as a research tool to help answer such theoretical questions.

Regarding the third question: suppose we take the best performing bidding strategy, equip it with the most faithful opponent model, and combine this with the most effective acceptance condition; it would seem reasonable to assume this combination results in an effective negotiator. We plan to elaborate on this conjecture in future work (see also Section 6); however, Section 5 will already provide a first step towards this goal by recombining components of ANAC agents to create more effective agents than the original versions.

4 Decoupling Existing Agents

In this section we provide empirical evidence that many of the currently existing agents can be decoupled by separating the components of a set of state of the art agents. 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; 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.

4.1 Identifying the Components

In this section we identify the components of 21 negotiating agents, taken from the ANAC competition of 2010 [8], 2011 [7] 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 the acceptance mechanism, 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 3.2.

The first step in decoupling an agent is to determine which components can be identified. For example, in the ANAC 2010 agent *FSEGA* [36], an acceptance condition, a bidding strategy, and an opponent model can all be identified. The acceptance condition combines simple, utility–based criteria (called $\mathbf{AC}_{\text{const}}$ and $\mathbf{AC}_{\text{prev}}$ in [6]), and can be easily decoupled in our architecture. The opponent model is a variant of the *Bayesian opponent model* [4, 5, 24], which is used to optimize the opponent utility of a bid. Since this usage is consistent with our architecture (i.e., the opponent model 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 bidding strategy to use the generic opponent model and acceptance conditions instead of its own specific implementation. In addition to this, the opponent model and acceptance condition need to be altered to allow the other bidding strategies 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 to separate the design into singular components. For example, the acceptance condition and bidding strategy of the ANAC 2011 agent *The Negotiator*[1] 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

[1] Descriptions of all ANAC 2011 agents can be found in [3].

Table 1. Overview of the BOA components found in every agent. ✓: original has component, which can be decoupled. ∅: original has no such component, but it can be added. – : no support for such a component.

ANAC 2010	**B**	**O**	**A**	**ANAC 2011**	**B**	**O**	**A**	**ANAC 2012**	**B**	**O**	**A**
FSEGA [36]	✓	✓	✓	*Agent K2* [30]	✓	∅	✓	*AgentLG*	✓	∅	✓
Agent K [29]	✓	∅	✓	*BRAMAgent* [20]	✓	–	✓	*AgentMR*	✓	∅	✓
Agent Smith [37]	✓	✓	✓	*Gahboninho* [12]	✓	–	✓	*BRAMAgent2*	✓	–	✓
IAMcrazyHaggler [39]	✓	∅	✓	*HardHeaded* [38]	✓	✓	✓	*CUHKAgent* [22]	✓	–	–
IAMhaggler [39]	✓	✓	✓	*IAMhaggler2011* [41]	✓	∅	✓	*IAMhagger2012*	✓	∅	✓
Nozomi	✓	∅	✓	*Nice Tit for Tat* [7]	✓	✓	✓	*OMAC Agent* [14]	✓	∅	✓
Yushu [1]	✓	∅	✓	*The Negotiator* [15]	✓	∅	✓	*The Negotiator Rel.*	✓	✓	✓

loss, by sharing the code between the components. 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.

Table 1 provides an overview of all agents that we re-implemented in our architecture, and more specifically, which components we were able to decouple. In fact, we were able to decouple all ANAC 2010, and most ANAC 2011 and ANAC 2012 agents.

There were two agents (*ValueModelAgent* [21] and *Meta-Agent* [26]) that were not decoupled due to practical reasons, even though theoretically it is possible. The *ValueModelAgent* was not decoupled because there were unusually strong dependencies between its components. Decoupling the strategy would result in computationally heavy components when trying to combine them with other components, making them impractical to use. The ANAC 2012 *Meta-Agent* chooses an offer among 17 agents from the ANAC 2011 qualifying round. This agent was not decoupled because it requires the decoupling of all 17 agents, of which only 8 optimized versions entered the finals.

The *CUHKAgent*, like *ValueModelAgent*, is heavily coupled with multiple variables that are shared between the bidding strategy and acceptance condition. This makes it very hard to decouple and can make components unusable in combination with other components (e.g. variables might not properly be set). However, since *CUHKAgent* was placed first in the ANAC 2012 competition, we decided to decouple its bidding strategy, allowing it to work with other acceptance conditions and opponent models.

Four additional agents were only partially decoupled: *AgentLG*, *BRAMAgent*, *BRAMAgent2*, and *Gahbininho*. As is evident from Table 1, the only obstacle in decoupling these agents fully is their usage of the opponent model, as it can be employed in many different ways. Some 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 in opponent utility, and extrapolating opponent utility. The generic opponent

model interface needs to sufficiently accommodate such requirements from the bidding strategy 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.

With regard to the opponent model, there are three groups of agents: first, there are agents such as *FSEGA* [36], which use an opponent model that can be freely interchanged; second, there are agents such as the ANAC 2010 winner *Agent K* [28], which do not have an opponent model themselves, but can be extended to use one. Such agents typically employ a bidding strategy that first decides upon a specific target utility range, and then picks a *random* bid within that range. These agents can easily be fitted with an opponent model instead, by passing the utility range through the opponent model before sending out the bid. Lastly, there are agents, for example *Gahboninho* and *BRAMAgent*, that use a similarity heuristic which is not compatible with our architecture, as their opponent models do not yield enough information to compute the opponent utility of bids. For these type of agents, we consider the opponent model part of the bidding strategy. *AgentLG* also uses an opponent model which is not compatible with our BOA architecture; however, it has been adopted to be able to use other opponent models.

When decoupling the agents, we can distinguish different classes within each component, except for the bidding strategy component, which varies greatly between different agents. For instance, there are only three main types of opponent models 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 [24, 42]. 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 opponent models, most agents use variations and combinations of a small set of acceptance conditions. Specifically, many agents use simple thresholds for deciding when to accept (called $\mathbf{AC}_{\text{const}}$ in [6]) and linear functions that depend on the utility of the bid under consideration ($\mathbf{AC}_{\text{next}}(\alpha, \beta)$ [6]).

4.2 Testing Equivalence of BOA Agents

A BOA agent should behave identically to the agent from which its components are derived. Equivalence can be verified in two ways; first, given the same negotiation environment and the same state, both agents should behave in exactly identical ways; second, the performance in a real time negotiation of both agents should be similar.

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 picking from a set of bids of similar utility; second, the default protocol in GENIUS uses real time [31], 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 [3] 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 secondly, the opponent strategy should not be the first to accept, to avoid masking errors in the agent's acceptance condition. Given these two criteria, we used the standard time-dependent tactics [18, 19] for the opponent bidding strategy. Specifically, we use *Hardliner* ($e = 0$), *Linear Conceder* ($e = 1$), and *Conceder* ($e = 2$). In addition, we use the *Offer Decreasing* agent, which offers the set of all possible bids 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 exactly 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.

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, most agents record only a partial negotiation history, while some acceptance strategies require the full history of

Table 2. ANAC 2011 reference results of the original agents using our hardware ($n = 10$). Best results are marked bold.

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

the agent and/or its opponent. In such cases, the agent can be constrained to be incompatible with these acceptance strategies, or generalized to work with the full set of available acceptance strategies. We typically elected the most universal approach, even when this negatively influenced performance. We will demonstrate that while there is some performance loss when decoupling existing agents, it 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, cf. [3]). 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. Therefore, for each of these agents we 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. Next, 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 2, which is used as our baseline to benchmark the decoupled agents. The difference in performance caused small changes compared to the official ANAC 2011 ranking, as *Agent K2* moved up from 5th to 3rd place.

Table 3 provides an overview of the results. We evaluated the performance in terms of the the difference in 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*

Table 3. Differences in overall utility and time of agreement between the original agents and their decoupled version. Positive difference means the BOA agent performed slightly better.

	Diff. time agr.	**SD time agr.**	**Diff. utility**	**SD utility**
Agent K [29]	0.001	0.003	0.006	0.006
Agent Smith [37]	0.010	0.010	0.004	0.006
FSEGA [36]	0.001	0.004	0	0.003
IAMcrazyHaggler [39]	-0.004	0.012	0.003	0.013
IAMhaggler [39]	0.003	0.015	0.002	0.011
Nozomi	0.003	0.009	0.004	0.008
Yushu [1]	0.002	0.004	0.002	0.005
Agent K2 [30]	0.002	0.009	0.001	0.005
BRAMAgent [20]	0.004	0.011	0	0.006
Gahboninho [12]	0.001	0.008	0.006	0.005
HardHeaded [38]	-0.003	0.003	-0.009	0.004
IAMhaggler2011 [41]	-0.010	0.013	-0.002	0.003
Nice Tit for Tat [7]	0.006	0.010	-0.008	0.005
The Negotiator [15]	0	0.002	0	0.004
BRAMAgent2	0.002	0.011	-0.015	0.012
IAMhaggler2012	-0.005	0.006	-0.013	0.003
OMAC Agent [14]	0.003	0.003	0.012	0.015

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.

5 Applications of the BOA Architecture

The BOA architecture has already been widely applied since it was first released. Since its implementation in 2011, the BOA architecture has been used in the ANAC competitions that followed. In ANAC 2012, the BOA agent *The Negotiator Reloaded* reached the finals and finished overall third and received the reward for best performing agent in non-discounted domains. In ANAC 2013, two agents that used the BOA architecture reached the finals. The agent *Inox* finished fourth, and *The Fawkes* agent won the 2013 competition.

The BOA architecture has also found its way into the classroom. At academic institutes such as *Bar-Ilan University*, *Ben-Gurion University of the Negev*, *Maastricht University*, and *Delft University of Technology*, GENIUS and the BOA

architecture have been integrated into artificial intelligence courses, where part of the syllabus covers automated negotiation and the creation of negotiation strategies.[2] The BOA framework offers the students an easier and more structured way to 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.

The BOA framework also allows us to search the large space of negotiation strategies [4, 5]. Section 5.1 describes techniques integrated in the BOA framework that aid in this search by scaling down the negotiation strategy space. Section 5.2 describes an application of this technique, where we employ the BOA framework to improve upon existing ANAC strategies.

5.1 Scaling the Negotiation Space

Suppose that two negotiating BOA agents A and B have identical bidding mechanisms and the same opponent modeling technique, so that only their acceptance criteria differs. Furthermore, suppose agent A accepted in the middle of the negotiation, while agent B accepted somewhere towards the end. The agents accepted at a different time during the negotiation, but their bidding behavior will be identical up to the point of the first acceptance. The only difference between the complete traces is that the trace of agent A is cut-off in the middle of the negotiation.

In the BOA architecture we exploit this property by running all acceptance conditions in parallel while we record when each acceptance condition accepts. This drastically reduces the amount of different component combinations, as any amount of acceptance conditions can be investigated during one negotiation session. We refer to this approach as multi-acceptance criteria (MAC). Note that a similar technique cannot be applied for the bidding strategy and the opponent model, as both components directly influence the negotiation trace.

In addition, a large number of acceptance conditions varying only in their parameter value can be tested during the same negotiation thread. This technique can then be used to easily optimize a parameter of a single acceptance condition. Note that this approach assumes that checking additional acceptance conditions does not introduce a large computational overhead. In practice we found that the computational overhead was less than 5%, even when more than 50 variants of acceptance conditions were used at the same time.

5.2 Improving the State of the Art

Using the scaling methods discussed in the previous section, we give a practical application of the BOA architecture to show how it can be employed to explore the negotiation strategy space. To do so, we considered the original bidding strategy of every ANAC agent, and attempted to find a better accompanying opponent model and acceptance condition.

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

Table 4. All acceptance conditions that were used in the experiment to search the negotiation strategy space.

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
$\mathbf{AC}_{\text{opt.stop}}$	–	–
$\mathbf{AC}_{\text{AgentLG}}$	–	–
$\mathbf{AC}_{\text{OMAC}}$	–	–
$\mathbf{AC}_{\text{TheNegotiatorReloaded}}$	–	–

Searching the Negotiation Space We used the following combinations of BOA components:

(B) For the bidding strategies, we used all ANAC agents that we were able to successfully decouple (see Table 1).

(O) For our opponent model set, we selected the best Bayesian model (*IAMhaggler Bayesian Model* [40]), frequency model (*Smith Frequency Model* [37], and the best value model (*CUHKAgent Value Model* [22]) as identified in [5].

(A) All acceptance conditions of the top four agents of ANAC 2012 were used, except for *CUHKAgent* as it could not be decoupled. In addition, we used a set of baseline acceptance criteria, such as $\mathbf{AC}_{\text{combi}}(T, \text{MAX}^W)$ [6], and an optimal stopping acceptance condition $\mathbf{AC}_{\text{opt.stop}}$ based on Gaussian process strategy prediction as discussed in [9]. Table 4 provides an overview of all 15 tested acceptance conditions.

For each bidding strategy, we ran a tournament on a subset of the ANAC 2012 domains against the eight ANAC 2012 agents. Note that 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 [3] which may all last 3 minutes. In worst case, it requires 22 hours to run a single tournament, and almost four weeks for running it 28 times, as we did for the similarity test discussed in Section 4.2.

We opted to use a representative subset of the domains to improve scalability. The following domains were used: *Barter* (80), *IS BT Acquisition* (size 384), *Barbecue* (1440), *Phone* (1600), *Energy (small)* (15625), and *Supermarket (112896)*. 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 as discussed in Section 5.1. The complete tournament is repeated five times to improve 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 5. This table shows the optimal acceptance condition and opponent model for each agent, as well as

Table 5. Results of the optimized BOA agents, when tested on both $n = 6$ domains and $n = 4$ domains. The $\mathbf{Diff}_n$ column indicates the utility gain of the agent when coupled with the optimal components listed in the **OM** and **AC** column. $\mathbf{Rank}_n$ **pre** indicates the rank of the original agent, while $\mathbf{Rank}_n$ **post** gives its ranking after optimization.

	OM	**AC**	$\mathbf{Diff}_6$	$\mathbf{Rank}_6$ **pre**	$\mathbf{Rank}_6$ **post**	$\mathbf{Diff}_4$	$\mathbf{Rank}_4$ **pre**	$\mathbf{Rank}_4$ **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

their scores in the tournament. Due to scalability issues, some agents were only run on the four smallest domains instead of all six domains. Therefore, we show the results for these four domains, as well as for all 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 improves from the twelfth place to the fourth when it employs *IAMhaggler*'s opponent model, and optimal stopping as its acceptance mechanism.

The only agents we were not able to improve are *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 opponent model at all, or because their bidding technique does not benefit much from one. The second reason is that these agents already employ acceptance criteria that perform well, or have an acceptance strategy that is tightly coupled with their biddings strategy.

An interesting pattern in the results, is that nearly all agents were improved by using the acceptance condition $\mathbf{AC}_{\text{opt.stop}}$. For the opponent model, the *IAMhaggler Bayesian Model* is often best, although the results indicate that the differences between the opponent models are minimal; that is, a better acceptance strategy often results in a larger gain than an improved opponent model.

All in all, the results demonstrate that the BOA architecture not only assists in exploring the negotiation strategy space and to strongly improve existing

agents, but it also helps to identify which components of the agent are decisive in its performance.

6 Conclusion and Future Work

This paper introduces an architecture that distinguishes the bidding strategy, the opponent model, and the acceptance condition in negotiation agents, and recombines these components to systematically explore the space of automated negotiation strategies. 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. Our motivation in the end is to create a proficient negotiating agent by combining the best components.

We have shown that many of the existing negotiation strategies can be re-fitted into our architecture. We identified and classified the key components in them, and we have demonstrated that the original agents and their decoupled versions have identical behavior and similar performance. Finally, we discussed several applications of the BOA architecture, one of which was to recombine different components of the ANAC agents, and we have demonstrated this significantly improved their performance.

One obvious direction of future research is to look at any of the BOA components in isolation. After identifying the best performing components, we can turn our attention to answer whether combining effective components leads to better overall results, and whether an optimally performing agent can be created by taking the best of every component. Another interesting question then is which of the BOA components turns out to be most important with regard to the overall performance of an agent. Our architecture allows us to make these questions precise and provides a tool for answering these questions.

Another possible improvement is extend the focus of current work on preference profile modeling techniques to a larger class of opponent modeling techniques, such as strategy prediction. Also, an agent is currently equipped with a single component during the entire negotiation session. It would be interesting to run multiple BOA components in parallel, and use recommendation systems to elect the best component at any given time.

Acknowledgments

This research is supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs. It is part of the Pocket Negotiator project with grant number VICI-project 08075.

References

1. 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.

2. 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.
3. 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.
4. 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.
5. Tim Baarslag, Mark Hendrikx, Koen Hindriks, and Catholijn Jonker. Predicting the performance of opponent models in automated negotiation (submitted). In *2013 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2013.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. C. Beam and A. Segev. Automated negotiations: A survey of the state of the art. *Wirtschaftsinformatik*, 39(3):263–268, 1997.
12. 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.

13. 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.
14. Siqi Chen and Gerhard Weiss. An efficient and adaptive approach to negotiation in complex environments. In *ECAI*, pages 228–233, 2012.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. Radmila Fishel, Maya Bercovitch, and Yaakov(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.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.

26. 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.
27. 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.
28. 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.
29. Shogo Kawaguchi, Katsuhide Fujita, and Takayuki Ito. Agentk: Compromising strategy based on estimated maximum utility for automated negotiating agents. In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-Based Complex Automated Negotiations*, volume 383 of *Studies in Computational Intelligence*, pages 137–144. Springer Berlin Heidelberg, 2012.
30. 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.
31. 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.
32. 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.
33. 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.
34. I. Papaioannou, I. Roussaki, and M. Anagnostou. Multi-modal opponent behaviour prognosis in e-negotiations. In *Proceedings of the 11th international conference on Artificial neural networks conference on Advances in computational intelligence-Volume Part I*, pages 113–123. Springer-Verlag, 2011.
35. Ariel Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, 1982.
36. 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.

37. 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.
38. 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.
39. 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.
40. 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.
41. 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.
42. D. Zeng and K. Sycara. Bayesian learning in negotiation. *International Journal of Human Computer Systems*, 48:125–141, 1998.