Practical Assignment for MAS: Designing a Negotiation Agent

February 10, 2020

Abstract

This practical assignment is about designing a negotiation agent in GENIUS. For questions regarding this practical assignment, consult the FAQ information and/or visit the instruction hours.

1 Introduction

Imagine you have just graduated and plan to throw a nice party for the occasion. The problem is that you do not want to organize everything yourself. Fortunately, a friend of yours has also graduated and wants to throw a party as well. You decide to organize the party together; however, you have different preferences about essential aspects of the party, such as the choice of food, drinks, music, etc. You will now have to make joint decisions about what the party will look like; that is, you will have to *negotiate*. Being a computer science student you use your skills to make a negotiating agent so the party is as close to *your* preferences as possible.

Figure 1: Organizing a party involves choosing music, food, drinks, catering, etc.

This assignment is about *negotiation*: a form of interaction in which two (or more) agents, with conflicting interests and a desire to cooperate, try to reach a mutually acceptable agreement. Negotiation between agents can in many ways be modeled as a game, and game theory is useful to analyze the behavior of negotiating agents. Negotiation is, however, also different from many board games such as chess and reversi.

One of the most important differences is that negotiation as we will study it in this practical assignment never is a zero-sum game. That is, a typical negotiation does not have a winner who takes all and a loser who gets nothing. In order to start a negotiation, it is only reasonable for both parties to believe that there is a *win-win* situation possible where both agents can gain by obtaining a deal through negotiation. Another difference is that the domain of negotiation (what the negotiation is about) may be quite different from one negotiation to the other.

Typically, negotiation is viewed as a *process* divided into several phases. Initially, in a *prenegotiation phase* the negotiation domain and the issue structure related to the domain of negotiation are fixed. Negotiation may be about many things, ranging from quite personal issues such as deciding on a holiday destination to strictly business deals such as trading orange juice in international trade. In general, the issue structure tells you what issues are negotiable and what value-range that issue can take.

In this assignment, the party domain serves as an example and the structure of this domain is provided to you. The party domain is about co-organizing a party and negotiating what you want to spend the available money on. In other words, the prenegotiation phase has been completed and you cannot redefine the domain anymore. However, there are two tasks that are usually considered part of the prenegotiation phase that you do need to consider in this assignment. The first task involves creating a so-called *preference profile* that captures your own preferences with regards to the party domain. The result will be a formal preference function that maps each possible *outcome* of a negotiation (e.g. a party with rock music, catering, and handmade cocktails) to a *utility number* in the range of 0 to 1. The second task involves thinking about a *strategy* used to perform the negotiation itself. In negotiation you need at least two strategies: an offering strategy (what to bid when), and an acceptance strategy (when to accept or reject offers, and when to stop negotating - walk away). The most important part of this assignment concerns the *negotiation phase* itself, i.e. the exchange of offers between you (or your software agent) and the opponent.

Figure 2 provides some initial guidelines based on human experience with negotiation that may help you during your own negotiations and in building your own negotiating software agent.

Orient yourself towards a win-win approach.
Plan and have a concrete strategy.
Know your reservation value, i.e. determine which bids you will never accept.
Create options for mutual gain.
Take the preferences of your opponent into account.
Generate a variety of possibilities before deciding what to do.
Pay a lot of attention to the flow of negotiation.

Figure 2: General negotiation guidelines

This assignment concerns designing and building your own negotiating software agent. This will be a *team effort*: as a team you will design and implement your negotiating agent in Java to do negotiations for you.

Some techniques relevant for building and designing negotiating agents can be found among others in chapters 16 and 17 in [1], chapter 2.3 in [2], [3], and [4]. You might also want to go beyond the standard course material. Additional information is provided to you in the form of several papers [5, 6, 7, 8, 9]. There is a lot of other literature available about negotiation that may help you finish this assignment successfully. As for almost any subject, you can find more information about negotiation strategies online. We recommend to search for strategies used in the ANAC competition [10, 11, 12, 13, 14, 15, 16, 17].

The remainder of this document is organized as follows. Section 2 provides some background to the assignment, which in turn is detailed in Section 3. In Section 4 the objectives, deliverables, requirements

and assignment itself are described. Section 4.6 describes some organizational details and important dates, including deadlines. Finally, Section 5 documents the evaluation criteria and grading for this assignment.

2 Background

Before we present the main tasks and questions you need to complete, some additional background is provided that may help you complete this assignment successfully. After that we present some easy steps to help make yourself familiar with the negotiation environment [7]. Part of that requires you to start and use the negotiation environment and read the user guide as well as this assignment provided to you carefully. Note that the negotiation environment is a scientific software tool which is being improved constantly. Please report any major bugs found so that we can resolve them in a next version.

A negotiation in this assignment is also called a *negotiation session*. In a session two agents negotiate with each other to settle a conflict and negotiate a deal. Each negotiation session is limited by a fixed amount of time. At the end of a session, a score is determined for both agents based on the utility of the deal for each agent if there is a deal, otherwise the score equals the reservation value; in this assignment 0.0. In a sense, a negotiation does not yield a winner since each agent will obtain a score based on the outcome and its own utility function. A failed negotiation, in the sense that no deal is reached, thus is a missed opportunity for both agents. Your agent will also negotiate in a tournament with all other agents where the scores of each session are recorded and averaged to obtain an overall score (see Section 5). A ranking will be compiled using these overall scores.

Each negotiator will have a specific set of *preferences* regarding the possible outcomes. Such preferences can be specified by an ordering over the set of possible outcomes: an outcome ω is said to be weakly preferred over an outcome ω' by an agent if $\omega \succeq \omega'$, or strictly preferred if $\omega \succ \omega'$. If only a limited amount of these preferences are known, the agent is said to act under *preference uncertainty*. Under mild assumptions [4], preferences can also be expressed in a more complete, numerical way by a so-called *utility function* (cf. [1]). In our case, utility functions assign quantitative values to bids in the negotiation space. In this assignment, you may assume that all utility functions are additive, i.e. they will always be linear in the number of issues. For example, if there are four issues to negotiate about, the utility function can be computed by a *weighted sum* of the values associated with each of these issues. So, let $bid = \langle i_1, i_2, i_3, i_4 \rangle$ be a particular bid. Then the utility $u(bid) = u(i_1, i_2, i_3, i_4)$ (given weights w_1, w_2, w_3, w_4) can be calculated by:

$$u(i_1, i_2, i_3, i_4) = w_1 \cdot u(i_1) + w_2 \cdot u(i_2) + w_3 \cdot u(i_3) + w_4 \cdot u(i_4).$$

Key to this assignment is the fact that you have incomplete information about your opponent. You may assume that there is a conflict of interests, but at the start you do not know on which issues agents agree and on which they disagree. For example, in a negotiation about buying a laptop, the buyer may prefer to have a middle-sized screen but the seller may prefer to sell laptops with small screens because they have more of those in stock. They could, however, find themselves aligned on what brand of laptop that they want to buy/sell. An outcome of a negotiation recognizes and reconciles such differences and results in an agreed solution to resolve the conflict.

Ideally, an agreed solution has certain properties. One of these properties is that the outcome should be Pareto optimal. An outcome is Pareto optimal when there does not exist a deal where one agent can do better and the other can do no worse (i.e. they both score higher or equal, and therefore would prefer this new deal over the old one). Another property is related to the Nash solution concept. A Nash solution is an outcome that satisfies certain bargaining axioms and may be viewed as a "fair outcome" which is reasonable to accept for both parties. An outcome is said to be a Nash solution whenever the product of the utility (in the range $[0,1]$) of the outcome for the first agent and that for the second agent is maximal (cf. [9]; note how this is different from the concept of a Nash equilibrium).

The notion of a fair outcome is important in negotiation because it provides a reference point for what your opponent might be willing to accept. Typically, a negotiation is started in the first place because reaching an agreement is preferable for both. But in order to get an agreement, you will need to get your opponent to agree. In a negotiation between self-interested agents, however, each agent is determined to get the best possible outcome for itself.

The Nash solution concept excludes certain outcomes as being unfair. It is, for example, very unlikely that your opponent will accept an offer that is most favorable to you and leaves your opponent with empty hands. Therefore, an outcome is usually the result of a number of concessions that both parties make. The speed of making concessions, or the concession rate, is the derivative of the (size of the) concession steps taken during a negotiation. Concessions can be made in various ways; an agent that makes concessions in very small steps initially uses a so-called Boulware strategy. A strategy that makes faster concessions initially is called a Conceder. Other strategies are conceivable and may be necessary given the deadlines (cf. [18]).

The outcome space of a negotiation, i.e. all possible bids, can be plotted on a graph which indicates the utility of both agents on the x and y axes respectively. An example is provided in Figure 3. This graph is useful for understanding and analyzing bilateral negotiations. This will become clear while you work on this assignment.

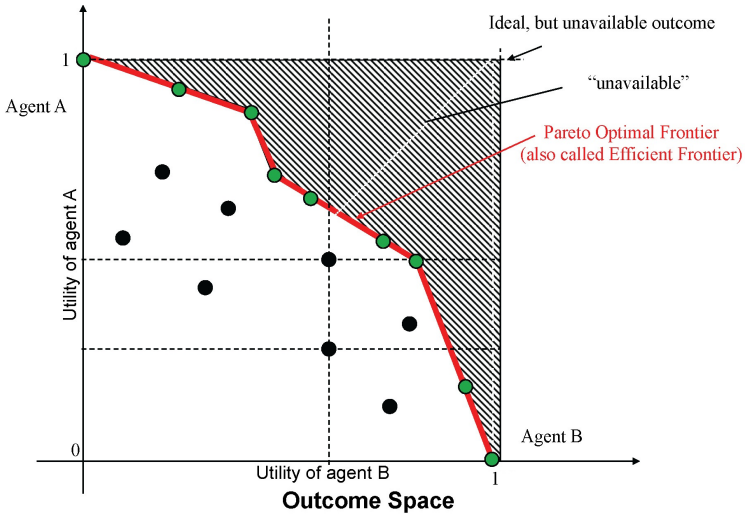

Figure 3: Pareto Frontier

Besides utility, often in negotiations time plays a crucial role. Negotiations can have deadlines, e.g., if the date of the party has already been fixed, then an agreement should be made in advance of that day. In this assignment there is a deadline for reaching an agreement: both parties have exactly the same deadline for achieving a deal and both parties "know" this. If they fail to reach an agreement each participant gets 0 utility.

3 Assignment

The assignment consists of first familiarizing yourself with the negotiation environment and then designing and implementing a negotiating software agent. You do not have to discuss the questions in 3.1 in your report.

3.1 Familiarizing Yourself With the Negotiation Environment

i Go to the GENIUS website, to find the latest release and read its User Guide. In order to familiarize yourself with the environment, complete the items below.

ii Inspect the negotiation templates named `laptop_domain.xml`, `laptop_buyer_utility.xml` and `laptop_seller_utility.xml` in the GUI. Make a new preference profile for buying a laptop and edit the preferences according to your own preferences.

iii Enable preference uncertainty for your new laptop preference profile and select a number of rankings to generate from your profile. Think about why the maximum number of rankings is 27. Then save your changes.

iv Create a new negotiation session. For the agents (called *parties* in GENIUS) choose the `UIagent` and the `ConcederNegotiationParty` [8]. Select a different preference profile for each agent within the Laptop domain (i.e. a buyer and a seller profile) and start the negotiation.

Familiarize yourself with running a negotiation and note the difference between both agents: the `ConcederNegotiationParty` is automated, whereas `UIagent` asks the user for input. You may want to repeat this exercise using two automated agents. While entering bids for `UIagent`, try to aim for concessions that lie on the Pareto frontier (while choosing your offers, you can ignore the column called `lastAcceptedBid`).

v Analyze the performance of the `SimpleAgent` in the laptop domain by letting it play against itself. Is a Pareto optimal outcome reached? Add the example agent `RandomBidderExample` to the agent repository and run the same negotiation again. Think about why they obtain the outcomes that they do. Further information about `SimpleAgent` and `RandomBidderExample` is available in the user guide.

vi Now we know how to run a single negotiation session, it is time to run a tournament. In the user guide it is discussed how to run a tournament. Specify a simple tournament with the following parameters:

- PROTOCOL : *Stacked Alternating Offers Protocol*.
- DEADLINE : *60 time (seconds)*.
- NUMBER OF TOURNAMENTS: *1*.
- AGENTS PER SESSION: *2* (this always needs to be set to 2 for bilateral negotiations).
- AGENT REPETITION: *false* (can be changed after selection of agents).
- RANDOMIZE SESSION ORDER: *false*.
- ENABLE SYSTEM.OUT PRINT: *false*.
- DATA PERSISTENCY: *disabled*.
- PARTIES: *ConcederNegotiationParty, IAMhaggler2011 (ANAC2011)*.
- PREFERENCE PROFILES: *both preference profiles of "anac/y2011/Amsterdam"*.
- AGENTS PLAY BOTH SIDES: *true*.

Give GENIUS some time to run the tournament. After running your tournament there should be two logs in the log directory. As the logs are in XML format, you can easily analyze them by importing them with a script or in a spreadsheet and using pivot tables. For Excel users, note that these features are only available in the full version.

vii Using a similar method as above, we can also run a tournament featuring BOA parties. The difference between a normal party and a BOA party is that a BOA party consists of four components (cf. [19]): an *offering strategy* (for deciding what to bid to the opponent), an *acceptance strategy* (for deciding when to accept an offer from the opponent), an *opponent model* (for learning the opponent's preferences) and *an opponent model strategy* (for deciding how the opponent model is taken into account when making an offer). A large set of components is included in GENIUS and we can simply create a BOA party from components of each type.

Follow the user guide to add a single BOA agent with the following components:

- Bidding strategy: *2011 - Agent K2*;
- Acceptance strategy: *Other - next*;
- Opponent model: *HardHeaded Frequency Model*;

- Opponent model strategy: *Best bid*.

Use a name for your BOA agent that you can easily find again. Run a negotiation with the agent and analyze the results.

viii Finally, check the example code included in GENIUS discussed in the userguide, and read how to configure Genius to create and analyze your own agent.

3.2 Designing and Implementing a Negotiating Software Agent

The remainder of this assignment concerns the design and implementation of a negotiation agent. This negotiating agent will be tested on a large set of scenarios, of which some are made available to you.

1. PREPARATION: ANALYZE NEGOTIATION DOMAIN

   (a) Consider the following holiday negotiation between two agents A and B:

Issue:	Possible Values:
Location	Antalya, Barcelona, Milan
Duration	1 week, 2 weeks
Hotel Quality	Hostel, 3 star hotel, 5 star hotel

Agent A's preferences:

Weights: $W_{location}$= 0.5 $W_{duration}$= 0.2 $W_{hotel\text{-}quality}$=0.3
Evaluation values: 4, 10, 2 (for Antalya, Barcelona and Milan respectively)
Evaluation values: 3, 10 (for 1 week and 2 weeks respectively)
Evaluation values: 10, 2, 3 (for hostel, 3 star hotel and 5 star hotel respectively)

Agent B's preferences:

Weights: $W_{location}$= 0.5 $W_{duration}$= 0.4 $W_{hotel\text{-}quality}$=0.1
Evaluation values: 3, 2, 10 (for Antalya, Barcelona and Milan respectively)
Evaluation values: 4, 10 (for 1 week and 2 weeks respectively)
Evaluation values: 3, 3, 10 (for hostel, 3 star hotel and 5 star hotel respectively)

Figure 4: Pareto Frontier

   Use the GUI of GENIUS to define the scenario and to compute and draw the Pareto efficient frontier in a graph such as Figure 3. What would you argue is the most fair holiday arrangement for A and B?

   (b) Analyze the performance of the example agent `RandomBidderExample` in the party domain playing against itself in a single negotiation session. Is a Pareto optimal outcome reached? Explain your answer. Also explain why it obtains the resulting outcome that it does and how the agreement depends on the negotiation setting. Now change the value of the `MINIMUM_TARGET` of `RandomBidderExample`; can you find a value for which a Pareto optimal outcome is reached? Perform the same analysis again for `BoulwareNegotiationParty` playing against `ConcederNegotiationParty` [8].

2. DESIGN A NEGOTIATION STRATEGY

   In this section you have to design and implement your negotiation agent using the BOA framework. A main advantage of this framework is that you can benefit from using a large set of existing components and mix and match them [20]. Before doing so, it is recommended to read about existing negotiation strategies (cf. [21, 11, 12, 13, 14, 15, 16, 17]).

   Note that this section adheres to the good practice of designing an agent properly before implementation; it is allowed however, to later amend your design when new evidence turns up (e.g. during the evaluation in Section 5), as long as these changes in design are documented and substantiated in your report.

(a) Create a PEAS description [1] of the negotiating agent that you need to design and implement in this assignment. That is, provide a description of your agent's:

- *Performance measure*: what you would like your agent to aspire (i.e. what should be achieved);
- *Environment*: what setting your agent will face (i.e. what is there);
- *Actuators*: what actions are available to your agent (i.e. what can be done);
- *Sensors*: what input is available to your agent (i.e. what can be observed).

For the performance measure, it is important to think carefully about how *you* will measure the success of your agent in order to achieve the ultimate goal of the agent: to outperform your peers in a tournament.

To specify the task environment, it suffices to provide a detailed account of this negotiation assignment setup. Be as complete as possible and pay special attention to the performance measure of your agent. Also include a brief discussion about each element included in your PEAS description. In particular, explain informally what goals your agent has in a negotiation.

(b) For implementing the agent we use the BOA agent framework as introduced in [5] and described in the userguide. Provide a small description of the role of each component in the BOA framework, as well as a high-level description of your envisioned design for each component for your agent. Discuss which parts will build on existing techniques and which parts will be new.

3. IMPLEMENT A NEGOTIATION STRATEGY

(a) Implement the **acceptance strategy** (Group*n*_AS). Describe which factors the agent takes into account for making this decision. Does your agent take time and/or opponent's actions into account when deciding to accept an offer? In case your agent also uses other considerations to determine whether it will accept particular bids, also explain these considerations.

Useful literature: [22],[23], [24].

(b) Implement the **offering strategy** (Group*n*_BS). Decide on the way your agent will make its first offer and counter offers. Explain why you have chosen this to be the agent's opening bid and how your agent computes a counter offer. List all considerations that your agent takes into account explicitly and explain why you have chosen to base the agent's decision on these considerations.

Does your agent ever propose offers that *increase* the utility for itself relative to its previous bid? If it does, explain when and why it will decide to do this. If not, demonstrate that the utility associated with bids in any bid sequence your agent will ever propose in a negotiation monotonically decreases. If it does, explain when and why it will decide to do this.

Useful literature: [25], [26], [27].

(c) Implement the **opponent model** (Group*n*_OM). For this assignment we included the source code of the *HardHeaded Frequency Model* (see [16]). Frequency models learn issue weights and value weights of the opponents preference profile separately. The weight of issue determines how much that issue impacts the utility of a bid. For example, in the domain of cars, some buyers find the price more important than the color. A very rich buyer might think differently. Knowing how important an issue is to a negotiator, is not same as knowing the preferences for all different *values* of that issue. For example, for the issue color, some might prefer the value "blue" over the value "red". Issue weights are calculated based on the frequency that an issue changes between two offers. In a frequency model, the value weights are calculated based on the frequency of appearance in offers made by the opponent.

While the Frequency Model can perform among the best known opponent models, it can be improved on. Initially, the quality of the opponent model is increases over time. However, after a small amount of time passed, the model actually becomes less accurate over time. Describe how you can improve (or completely replace) the given model and implement your changes. Explain why you believe your changes result in a better opponent model.

Useful literature: [28], [29], [30], [31], [32], [33].

(d) Implement the **opponent model strategy** (Group*n*_OMS). For this assignment you may improve the opponent model strategy "Best bid" which is included in this distribution or design your own opponent model strategy. A hint for an alternative implementation is to keep in mind that opponent models are imperfect.

Useful literature: [34], [35].

(e) Make your agent perform well under **preference uncertainty**. When there is preference uncertainty, a fully specified utility function is unavailable to your agent; instead, the preferences are specified in an alternative way by a *user model*, in our case by a *ranking* of a (limited) set of outcomes. For the exact details of how this works in GENIUS, please refer to the User Guide. The main idea is that instead of being able to calculate the utility of a Bid, as you normally would, the agent receives merely a list of example Bids, ordered by utility. You will have to use these example Bids to assign a value or order to all the other Bids you may encounter.

The standard implementation provided by GENIUS estimates the utility function from a ranking by using a simple counting heuristic. However, this can be greatly improved upon. For this assignment an example is included in the distribution of an acceptance strategy that is able to deal with preference uncertainty ("AC_Uncertain"). Incorporate your own way of estimating the utility function from a given set of bid rankings, not only in your acceptance strategy, but also in your offering strategy. Explain your design choices. (A hint for an alternative implementation is that you could apply ideas similar to the ones used in your opponent model.)

Useful literature: [36], [37], [38], [39], [40], [41], [42].

4. QUANTIFY THE PERFORMANCE OF YOUR AGENT

(a) Have your agent negotiate against itself on the party domain, and against the three ANAC2011 agents *HardHeaded*, *Gahboninho*, and *IAMhaggler2011* provided in GENIUS. Run several negotiation sessions using various utility profiles (with preference uncertainty disabled). Report on the outcomes reached. Try to explain why the outcome is as it is. Are any of the outcomes a Nash solution? Is it possible to reach an efficient outcome, i.e. an outcome that lies on the Pareto Frontier?

(b) Test your agent in a small tournament while varying the acceptance strategy. Is there an acceptance strategy which is better than your acceptance strategy?

(c) Test your agent in a small tournament and compare your opponent model with the original HardHeaded frequency model. Does your model lead to a significantly higher utility?

(d) Inspect the utility profiles that have preference uncertainty enabled for the party domain (i.e. `party1_utility_u`*N*`.xml` and `party2_utility_u`*N*`.xml`). Test and report on how your agent performs under different degrees of preference uncertainty (i.e. for different values of *N*).

(e) One important question remains. How generic is your agent? That is, does it work as well on the party domain as on other domains? To verify this, have your agent negotiate against itself, and against at least three agents provided to you on at least three scenarios (for example, *ItexvsCypress*, *Pie* and *Smart_Grid*). Report on the outcomes reached. Try to explain why the outcome is as it is. Is any of the outcomes a Nash solution? Is it possible to reach an efficient outcome, i.e. an outcome that lies on the Pareto Frontier?

Useful literature: [43], [44].

3.3 Concluding: Future Perspectives

5. The agents in the negotiation environment have limited capabilities in order to achieve a negotiated deal. What extensions would be required to use your agent in real-life negotiations to support (or even take over) negotiations performed by humans? Can you think of additional capabilities (e.g.

other actions or forms of communication) for your agent that would make it more practical and help achieve better deals?

Useful literature: [45], [46]

As a final remark, we want to make clear that it is very important to test your agent in order to improve the negotiation strength of your agent in different types of settings. A warning is in place: do not assume that your goal is to outperform a 'stupid' agent'. Try to beat as many agents as you can and older versions of your own agent. A good outcome is determined by other criteria, identified by efficiency of the outcome (i.e. is it close or not to the Pareto Frontier). To test your agent in this regard, it will in particular be useful to test your agent on various negotiation templates. You can create these templates both by modeling real-life examples of negotiation or create them randomly.

There are many techniques to improve the negotiation strength of agents. Various factors can be taken into account when deciding on acceptance, breaking off, or computing a next offer in a negotiation, for example you can take into account the time an opponent takes to reply with a counter offer, the consistency of an opponent's offers, the concession rate of your opponent, and possibly other considerations about the domain of negotiation itself.

If your agent performs well enough, you could even participate in the yearly competition for automated negotiating agents, called ANAC[1] [10, 47, 48]. Student teams that submitted their agent after this assignment have received prizes in the competition (and won $1000,- in 2011 and 2013) and were awarded student scholarships to attend international conferences to present their agent.

4 Detailed Assignment Deliverables

The assignment must be completed in teams of 4 students. In the following paragraphs the objectives, deliverables, requirements, and the detailed assignment description are documented.

4.1 Objectives

- To learn and apply techniques from multi-agent negotiation, preference modeling and utility theory, communication protocols and coordination in multi-agent systems, group decision-making, opponent modeling, decision-making under uncertainty.
- To learn to design and analyze a negotiating agent for a realistic domain with discrete issues, including among others a negotiation strategy.
- To learn techniques for implementing (adversarial) search and design heuristics while taking into account time constraints.
- To actively interact with other students and participate in student groups by discussing and coordinating the design and construction of a negotiating agent.

4.2 Deliverables

- Your group number n. Group members must register their group (i.e. your names, your background, and the name of the group coordinator) and then receive their group number n. Your deliverable and negotiation party must use the assigned group number.
- A negotiating agent programmed in Java using the negotiation environment provided to you. Your submission should consist of a package containing your source agent code: both the class and src files. It is obligatory to use the package "mas2020.group" + n for the negotiating agent and any other required files. An agent consists of four components and optionally several helper classes. For example for group 3 the package should at least include: *Group3_BS* (bidding strategy), *Group3_AS* (acceptance strategy), *Group3_OM* (opponent model), and *Group3_OMS* (opponent model strategy).

[1] For more information on the automated negotiation competition see: http://web.tuat.ac.jp/~katfuji/ANAC2020

In addition, a custom *BOArepository.xml* should be included specifying the components with optionally their parameters.

- Reports documenting and explaining the solution.

The final report need not be lengthy (5 pages may be enough, 10 pages maximum), but should include an explanation and motivation of *all* of the choices made in the design of negotiating agent. The report should also help the reader to understand the organization of the source code (important details should be commented on in the source code itself). This means that the main Java methods used by your agent should be explained in the report itself.

Please make sure all your files are in the directory structure as explained above. Assuming that the group has number three, a valid directory structure is:

```
boarepository.xml
Group3_report.pdf
mas2020/
    group3/
        Group3_AS.class
        Group3_AS.java
        Group3_BS.class
        Group3_BS.java
        Group3_OM.class
        Group3_OM.java
        Group3_OMS.class
        Group3_OMS.java
        SomeHelperClass.class
        SomeHelperClass.java
```

4.3 Requirements

- The agent should implement a negotiation strategy to generate offers, and an acceptance strategy to decide whether to accept an offer or not.
- The agent must aim at the best negotiation outcome possible (i.e. the highest score for itself given the agent's preferences, while taking into account that it may need to concede to its opponent).
- Your opponent model has to *learn* the preferences of the opponent during the negotiation process.
- The agent meets reasonable time and memory constraints. Any agent that uses more than a minute in order to initiate the negotiation or to reply to an opponent's offer will be disqualified.
- Make sure your party works on all domains with linear additive profiles with *discrete* issues. You may also find references in the negotiation environment to *integer* issues, *real* issues, *reservation values*, and *discount factors*, but you can ignore these, as they do not play play a role in this assignment.
- The source code should contain explanatory comments that will allow third parties not involved in programming the code to understand it. A clear explanation of a method must be provided at the beginning of important methods. Parts of the code that are not self-explaining should receive additional documentation. Please incorporate enough information in comments in the code to ensure this. Finally, make sure that you have removed all debug printouts from your code and that your agent does not flood the 'System.out' channel.
- The agent implementation should be tested to ensure that it works on multiple systems, and requires nothing other than the files contained in your group's package. Integrating your agent into the negotiation environment should not be more difficult than adding your package (e.g. the folder "mas2020/group*n*/") in the right folder.
- The report should include:

- Your group number;
- An introduction to the assignment;
- A high-level description of the agent and its structure, including the main Java methods (mention these explicitly!) used in the negotiating agent that have been implemented in the source code. This includes detail of any changes that were made to the original agent design and a motivation for these;
- An explanation of the negotiation strategy, decision function for accepting offers, estimators for preference uncertainty, any important preparatory steps, learning techniques, and heuristics that the agent uses to decide what to do next;
- A section documenting the strong and weak points of your agent, the tests you performed to analyze (and improve) the negotiation strength of your agent. You must include scores of various tests over multiple sessions that you performed. Describe how you set up the testing situation and how you used the results to modify your agent, and motivate why you choose these testing methods/metrics/graphs;
- Answers to the questions posed in the assignment;
- A final conclusion written by the group coordinator, in which the coordinator discusses a summary of the experience of the team with regards to writing the report and building the negotiating agent, and a summary of who performed which tasks.

4.4 Fair Play

Agents may be disqualified if they violate the spirit of fair play. In particular, the following behaviors are strictly prohibited: designing an agent in such a way that it benefits some specific other agent, starting new Threads, or hacking the API in any way.

4.5 Plagiarism

Both the agent source code and the report need to be your own work. For the agent, you can use the code from example agents provided, but anything else needs to be clearly acknowledged in the report and when submitting the agent. Note that you are not allowed to directly use code of agents programmed by others. However, you are allowed to use ideas and strategies reported in academic papers, as long as you implement these strategies yourselves and you acknowledge the papers in your report. In case of doubt, feel free to ask! This is important as violations, deliberate or otherwise, will be reported.

4.6 Submission

Only group coordinators may use the submission email **e.westenbroek@students.uu.nl** to submit deliverables to the assignment. Make sure to read the requirements section thoroughly before working on your deliverables.

Important Dates:

1. **Group deadline**: Deadline for registering your group by the group coordinator is **13 Feb 2020, 23:59 CEST**. Please register as a group through e-mail before the deadline. Make sure your group consists of individuals with a heterogeneous background, with a range of expertise that covers all aspects of the assignment.

   Once the deadline has passed, it is not possible to change groups. All subsequent assignment submissions by the group coordinator are considered as submissions from the group. Should any problems arise within the group, the group coordinator should get in contact with the course managers well before the submission deadline.

2. **Submission deadlines**: There are 2 intermediate submissions and one final submission for this assignment. The deadlines and expected deliverables for all the submissions are given below. Please note that compilation failures and execution errors will be perceived as an incomplete submission.

Deadline	Topic	Deliverables
13 Feb 2020, 23:59 CEST	Group registration	Group members and group coordinator
26 Feb 2020, 23:59 CEST	Question 1	Report 1 (2 pages max)
11 March 2020, 23:59 CEST	Question 2	Report 2 (5 pages max)
12 April 2020, 23:59 CEST	Question 3-5 + negotiation agent	Final report: adhering to the structure (5 pages may be enough, 10 pages maximum) and expected content mentioned in the requirements section. Source code with correct packaging and structure.

The group coordinator submits the reports in PDF format and the package for your negotiating agent through e-mail. Use the naming conventions described elsewhere in this document, including your group number. Do not submit incomplete assignment solutions; only a complete assignment solution containing all deliverables will be accepted. The deadline for submitting the assignment is strict. If you have problems with your assignment, please contact us well in advance. Note that the last opportunity for this is the Question & Answer session on Thu. 26 Mar (15:15-17:00).

5 Evaluation

Assignments are evaluated based on several criteria. All assignments need to be complete and satisfy the requirements stated in the detailed assignment description section above. *Incomplete assignments are not evaluated*. That is, if any of the deliverables is incomplete or missing (or fails to run), then the assignment is not evaluated.

The final grade for the tutorial will be determined as a weighted average of several components, based on your team solution for your reports and the quality and performance of your negotiating agent. The components that are graded and their relative weights are described in Table 1 below.

Assignment	**Weight**
Report 1	10%
Report 2	20%
Final report and Negotiation agent	70%

Table 1: Grading weights for each deliverable

Broadly, the assignments will be evaluated using the following evaluation criteria:

- *Quality of the deliverables*: Overall explanation and motivation of the design of your negotiating agent; quality and completeness of answers and explanations provided to the questions posed in the assignment; explanatory comments in source code and quality of documentation.
- *Performance:* Agents of teams will be ranked according to negotiation strength (i.e. average selfish utility obtained in all settings). The ranking will be decided by playing a tournament in which every agent plays negotiation sessions against a set of agents – including the agents programmed by your peers – on a set of domains and for different levels of uncertainty. Therefore, your agent should be generic enough to play on any domain.

- *Originality:* Any original features of the agent or solutions to questions posed are evaluated positively. Note that you are required to submit *original* source code designed and written by your own team. Source code that is very similar to that of known agents or other students will not be evaluated and be judged as insufficient. Detection of fraud will be reported to the administration.

The final report will be graded according to the set of measures outlined in Table 2.

Criterion	**Measure**	**Weight**
Description	Completeness, correctness and clarity of the agent description	10%
Strategy	Motivation and understanding about the strategic aspects of the negotiation game	10%
Creativity	Sophistication and originality of the agent design through novel ideas and adaptations	20%
Literature	Motivation by, support from, and improvement over existing literature	10%
Analysis	Discussion of the performance of the agent, and ways to overcome the deficiencies	20%
Agent performance	Performance in a tournament with other agents, including baseline agents and agents implemented by other groups	20%
Code quality	Code correctness, readability, and class composition	10%

Table 2: Grading criteria, measures, and weights.

6 Master Thesis about negotiation

Did you like thinking about efficient negotiating strategies, or implementing a successful negotiating agent? Automated negotiation provides a lot of subjects to do your Master Thesis on! You can always ask the course assistants, have a look at the last page of this assignment, or contact us for more information: **T.Baarslag@uu.nl**.

Acknowledgements

This assignment has been written with the help of many people, including R. Aydoğan, T. Baarslag, H. Griffioen, C.M. Jonker, W. Pasman, P. Prajod.

Interested in the possibility of doing a Master Thesis about negotiation?

Negotiation has become a major and interesting research area within Artificial Intelligence. Negotiation is important in many domains ranging from business applications such as Internet market places to incident and crisis management. Negotiation is one of the main tools an autonomous agent has to agree about a course of action or to resolve conflicts with other agents. As such, you might be interested in doing a Master Thesis about negotiation and there are many angles that you can take, e.g.:

- Design of negotiation strategies and better opponent models (using machine learning techniques);
- Design of a negotiation support system, either advising humans in a particular domain, or even taking over negotiation all together;
- Design of a negotiation protocol that can generate better agreements (e.g., multi-agent settings, auctions, use a form of communication between the agents);
- Design of a testbed for negotiating agents;
- Extend the current negotiation setting to non-linear utility functions, add dependencies;
- Design agents in a multi-agent setting, for example an auction;
- Research related to negotiation, trust and the reputation of agents, either cognitively motivated or viewed from a more engineering point of view;
- Etc.

This can all be combined with a Literature Survey, in order to kick-start your thesis. If you are interested, don't hesitate to ask us: **T.Baarslag@uu.nl**

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[2] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2009. [Online]. Available: http://www.masfoundations.org/mas.pdf

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

[4] R. L. Keeney and H. Raiffa, *Decisions with Multiple Objectives*. Cambridge University Press, 1976.

[5] T. Baarslag, K. V. Hindriks, M. J. Hendrikx, A. S. Dirkzwager, and C. M. 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. [Online]. Available: http://ii.tudelft.nl/sites/default/files/boa.pdf

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

[7] R. Lin, S. Kraus, T. Baarslag, D. Tykhonov, K. V. Hindriks, and C. M. Jonker, "Genius: An integrated environment for supporting the design of generic automated negotiators," *Computational Intelligence*, vol. 30, no. 1, pp. 48–70, 2014. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8640.2012.00463.x

[8] P. Faratin, C. Sierra, and N. R. Jennings, "Negotiation decision functions for autonomous agents," *Robotics and Autonomous Systems*, vol. 24, no. 3-4, pp. 159 – 182, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889098000293

[9] R. Serrano, "Bargaining," 2005. [Online]. Available: http://levine.sscnet.ucla.edu/econ504/bargaining.pdf

[10] T. Baarslag, R. Aydoğan, K. V. Hindriks, K. Fujita, T. Ito, and C. M. Jonker, "The automated negotiating agents competition, 2010-2015," *AI Magazine*, vol. 36, no. 4, pp. 115–118, 12/2015 2015. [Online]. Available: http://www.aaai.org/ojs/index.php/aimagazine/article/view/2609

[11] M. Ben Adar, N. Sofy, and A. Elimelech, "Gahboninho: Strategy for balancing pressure and compromise in automated negotiation," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2013, vol. 435, pp. 205–208. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30737-9_13

[12] A. Dirkzwager, M. Hendrikx, and J. Ruiter, "The Negotiator: A dynamic strategy for bilateral negotiations with time-based discounts," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2013, vol. 435, pp. 217–221. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30737-9_16

[13] R. Fishel, M. Bercovitch, and Y. Gal, "Bram agent," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2013, vol. 435, pp. 213–216. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30737-9_15

[14] A. Frieder and G. Miller, "Value model agent: A novel preference profiler for negotiation with agents," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2013, vol. 435, pp. 199–203. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30737-9_12

[15] S. Kawaguchi, K. Fujita, and T. Ito, "AgentK2: Compromising strategy based on estimated maximum utility for automated negotiating agents," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2013, vol. 435, pp. 235–241. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30737-9_19

[16] T. van Krimpen, D. Looije, and S. Hajizadeh, "Hardheaded," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2013, vol. 435, pp. 223–227. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30737-9_17

[17] C. R. Williams, V. Robu, E. H. Gerding, and N. R. Jennings, "Iamhaggler2011: A gaussian process regression based negotiation agent," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2013, vol. 435, pp. 209–212. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30737-9_14

[18] S. S. Fatima, M. J. Wooldridge, and N. R. Jennings, "Optimal negotiation strategies for agents with incomplete information," in *Revised Papers from the 8th International Workshop on Intelligent Agents VIII*, ser. ATAL '01. London, UK: Springer-Verlag, 2002, pp. 377–392. [Online]. Available: http://dl.acm.org/citation.cfm?id=648208.757345

[19] T. Baarslag, K. V. Hindriks, M. J. Hendrikx, A. S. Dirkzwager, and C. M. Jonker, "Decoupling negotiating agents to explore the space of negotiation strategies," in *Novel Insights in Agent-based Complex Automated Negotiation*, ser. Studies in Computational Intelligence, I. Marsa-Maestre, M. A. Lopez-Carmona, T. Ito, M. Zhang, Q. Bai, and K. Fujita, Eds. Springer, Japan, 2014, vol. 535, pp. 61–83. [Online]. Available: http://dx.doi.org/10.1007/978-4-431-54758-7_4

[20] T. Baarslag, A. S. Dirkzwager, K. V. Hindriks, and C. M. Jonker, "The significance of bidding, accepting and opponent modeling in automated negotiation," in *21st European Conference on Artificial Intelligence*, ser. Frontiers in Artificial Intelligence and Applications, vol. 263. IOS Press, 2014, pp. 27–32. [Online]. Available: http://ebooks.iospress.nl/volumearticle/36911

[21] T. Baarslag, K. V. Hindriks, and C. M. Jonker, "A tit for tat negotiation strategy for real-time bilateral negotiations," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, and T. Matsuo, Eds. Springer Berlin Heidelberg, 2013, vol. 435, pp. 229–233. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30737-9_18

[22] ——, "Effective acceptance conditions in real-time automated negotiation," *Decision Support Systems*, vol. 60, pp. 68–77, Apr 2014. [Online]. Available: http://dx.doi.org/10.1016/j.dss.2013.05.021

[23] T. Baarslag and K. V. Hindriks, "Accepting optimally in automated negotiation with incomplete information," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 715–722. [Online]. Available: http://dl.acm.org/citation.cfm?id=2484920.2485033

[24] S. Kawaguchi, K. Fujita, and T. Ito, "Compromising strategy based on estimated maximum utility for automated negotiating agents," in *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*, T. Ito, M. Zhang, V. Robu, S. Fatima, and T. Matsuo, Eds. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 137–144. [Online]. Available: http://link.springer.com/content/pdf/10.1007%2F978-3-642-24696-8_8

[25] R. Ros and C. Sierra, "A negotiation meta strategy combining trade-off and concession moves," *Autonomous Agents and Multi-Agent Systems*, vol. 12, pp. 163–181, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10458-006-5837-z

[26] S. S. Fatima, M. J. Wooldridge, and N. R. Jennings, "Multi-issue negotiation under time constraints," in *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2002, pp. 143–150.

[27] P. Faratin, C. Sierra, and N. R. Jennings, "Using similarity criteria to make issue trade-offs in automated negotiations," *Artificial Intelligence*, vol. 142, no. 2, pp. 205 – 237, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370202002904

[28] R. A. Carbonneau, G. E. Kersten, and R. M. Vahidov, "Predicting opponent's moves in electronic negotiations using neural networks," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1266–1273, Feb 2008. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2006.12.027

[29] S. Chen, H. B. Ammar, K. Tuyls, and G. Weiss, "Optimizing complex automated negotiation using sparse pseudo-input gaussian processes," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 707–714. [Online]. Available: http://dl.acm.org/citation.cfm?id=2484920.2485032

[30] T. Baarslag, M. J. Hendrikx, K. V. Hindriks, and C. M. Jonker, "Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 5, pp. 849–898, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10458-015-9309-1

[31] K. V. Hindriks and D. Tykhonov, "Opponent modelling in automated multi-issue negotiation using bayesian learning," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS '08, vol. 1. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 331–338. [Online]. Available: http://dl.acm.org/citation.cfm?id=1402383.1402433

[32] F. Zafari and F. Nassiri-Mofakham, *Recent Advances in Agent-based Complex Automated Negotiation*. Cham: Springer International Publishing, 2016, ch. BraveCat: Iterative Deepening Distance-Based Opponent Modeling and Hybrid Bidding in Nonlinear Ultra Large Bilateral Multi Issue Negotiation Domains, pp. 285–293. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-30307-9_21

[33] J. Zhang, F. Ren, and M. Zhang, "Bayesian-based preference prediction in bilateral multi-issue negotiation between intelligent agents," *Knowledge-Based Systems*, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705115001446

[34] K. V. Hindriks, C. M. Jonker, and D. Tykhonov, "The benefits of opponent models in negotiation," in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2. IEEE Computer Society, Sep 2009, pp. 439–444.

[35] ——, "Using opponent models for efficient negotiation," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '09, vol. 2. International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 1243–1244.

[36] R. Aydoğan, T. Baarslag, K. V. Hindriks, C. M. Jonker, and P. Yolum, "Heuristics for using cp-nets in utility-based negotiation without knowing utilities," *Knowledge and Information Systems*, vol. 45, no. 2, pp. 357–388, Nov. 2015. [Online]. Available: http://dx.doi.org/10.1007/s10115-014-0798-z

[37] D. Tsimpoukis, T. Baarslag, M. Kaisers, and N. Paterakis, "Automated negotiations under user preference uncertainty: A linear programming approach," in *Proceedings of Agreement Technologies*, Jan. 2018. [Online]. Available: https://ir.cwi.nl/pub/28326/2018_AT_Tsimpoukis.pdf

[38] T. Baarslag and M. Kaisers, "The value of information in automated negotiation: A decision model for eliciting user preferences," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. Richland, SC: International Foundation

for Autonomous Agents and Multiagent Systems, 2017, pp. 391–400. [Online]. Available: http://dl.acm.org/citation.cfm?id=3091125.3091185

[39] E. Roszkowska, "The application of uta method for support evaluation negotiation offers," 2016.

[40] V. Srinivasan and A. D. Shocker, "Estimating the weights for multiple attributes in a composite criterion using pairwise judgments," *Psychometrika*, vol. 38, no. 4, pp. 473–493, 1973. [Online]. Available: https://link.springer.com/article/10.1007/BF02291490

[41] S. Greco, V. Mousseau, and R. Slowinski, "Ordinal regression revisited: Multiple criteria ranking using a set of additive value functions," *European Journal of Operational Research*, vol. 191, no. 2, pp. 416 – 436, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221707008752

[42] L. M. Zintgraf, D. M. Roijers, S. Linders, C. M. Jonker, and A. Nowé, "Ordered preference elicitation strategies for supporting multi-objective decision making," *arXiv preprint arXiv:1802.07606*, 2018. [Online]. Available: https://arxiv.org/pdf/1802.07606

[43] T. Baarslag, M. J. Hendrikx, K. V. Hindriks, and C. M. Jonker, "Measuring the performance of online opponent models in automated bilateral negotiation," in *AI 2012: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, M. Thielscher and D. Zhang, Eds., vol. 7691. Springer Berlin Heidelberg, 2012, pp. 1–14. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35101-3_1

[44] ——, "Predicting the performance of opponent models in automated negotiation," in *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, ser. WI-IAT '13, vol. 2. Washington, DC, USA: IEEE Computer Society, Nov 2013, pp. 59–66. [Online]. Available: http://dx.doi.org/10.1109/WI-IAT.2013.91

[45] 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*, vol. 10, no. 2, pp. 199–215, 2001.

[46] T. Baarslag, M. Kaisers, E. H. Gerding, C. M. Jonker, and J. Gratch, "When will negotiation agents be able to represent us? The challenges and opportunities for autonomous negotiators," in *Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence*, ser. IJCAI'17, 2017, pp. 4684–4690. [Online]. Available: https://doi.org/10.24963/ijcai.2017/653

[47] T. Baarslag, K. V. Hindriks, C. M. Jonker, S. Kraus, and R. Lin, "The first automated negotiating agents competition (ANAC 2010)," in *New Trends in Agent-based Complex Automated Negotiations*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, S. Fatima, and T. Matsuo, Eds., vol. 383. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 113–135. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24696-8_7

[48] C. M. Jonker, R. Aydoğan, T. Baarslag, K. Fujita, T. Ito, and K. Hindriks, "Automated negotiating agents competition (ANAC)," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence and Twenty-Ninth Innovative Applications of Artificial Intelligence Conference*, Feb 2017, pp. 5070–5072. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/download/14745/14021