IN4010 Practical Assignment Q2: Negotiation

2010-2011

1 Introduction

Imagine you are having a party. You have just graduated (with honors, of course) in Computer Science and plan to throw a nice party for that 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 share the load. Your parents will finance the party for an amount of up to 1200 euro.

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

This document describes the practical assignment for the second quarter of the AI course. The practical assignment aims at familiarizing students with parts of the course material in a more practical way. This assignment is about *negotiation*. Negotiation is 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 two 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 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.

To clarify the remarks about negotiation domains, we add some remarks about the *process* of negotiation. Typically, negotiation viewed as a process is 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 this assignment, the party domain has been selected and the structure of this domain is provided to you. In other words, the prenegotiation phase has been completed and you cannot redefine the domain anymore. 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 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. But 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 an 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.
Generate a variety of possibilities before deciding what to do.
Pay a lot of attention to the flow of negotiation.

Figure 2: Negotiation guidelines

This assignment consists of several parts. First, you will have to think about your preferences and each of you will have to *individually* define a preference profile for the party domain briefly mentioned above. This profile will be used in grading so it is probably good to advice you to be true to yourself. The domain itself will be provided to you at a later time. Then you will negotiate with another student and a software agent about this domain, with the aim of reaching an agreement while at the same time trying to get an outcome as close to your own preferences as possible. Finally, the largest part of this assignment concerns building your own negotiating software agent. This will be a *team effort*: as a team you will design and implement your negotiating agent to do negotiations for you in JAVA.

Some techniques relevant for building and designing negotiating agents can be found among others in chapters 16 and 17 in [5] or [6]. The assignment will also require you to go beyond the course material in the book. Additional information is provided to you in the form of several papers [4, 2, 7]. 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 on the Internet. For example, a brief definition of what negotiation is can be found on Wikipedia [3].

The remainder of this document is organized as follows. In Section 2 the objectives, deliverables, requirements and assignment itself are described. Section 3 documents the items provided to you to complete the assignment. Section 4 describes some organizational details and important dates, including deadlines. Finally, Section 5 documents the evaluation criteria and grading for this assignment.

2 Detailed Assignment Description

The assignment must be completed in teams of 4 or 5 students. In the following paragraphs the objectives, deliverables, requirements, and the detailed assignment description are documented.

2.1 Objectives

- To learn to design a negotiating agent for a realistic domain, 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.

2.2 Deliverables

- A unique number n to identify the team and the negotiating agent.
- A negotiating agent programmed in JAVA using the negotiation environment provided to you.
  - A package containing your agent code. It is obligatory to use the package "negotiation.group" + n for the negotiating agent and any other required files. Your main agent file must be named "Group" + n + ".java". For example, if you are group 3, your agent must be called Group3.java and be located in the package negotiation.group3. In particular, the file Group3.java should contain the following lines:

    package negotiation.group3;
    public class Group3 **extends** Agent

    You must submit both the source code *as well as the class files*. Please make sure all your files are in the directory structure as explained above. If you submit the files in a .zip or .jar file, name it "group" + n + ".jar". The root of the archive must *only* contain the folder "negotiation", which *only* contains the folder "group" + n. This last folder contains your agent "Group" + n + ".java" and all other files. Please look at Figure 3 for an example.
    Submissions that do not meet these requirements will not be accepted.

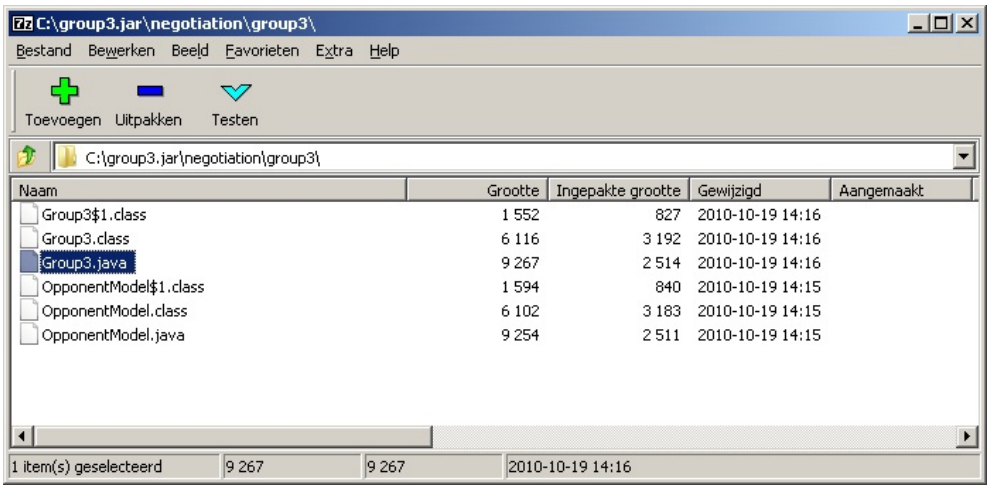

Figure 3: This is what your .jar file must look like

- A report documenting and explaining the solution, including an explanation and discussion of:
  - The main methods used in the negotiating agent and implemented in the source code,

- The negotiation strategy and the factors it takes into account, as well as the decision function for accepting an offer,
- Any preparatory steps the agent takes at the beginning of a negotiation session,
- Arguments why your agent is really aiming at the best negotiation outcome possible.

The report need not be lengthy (10 A4 pages may be enough), 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.

2.3 Requirements

- The agent should implement a negotiation strategy to generate offers, and a decision function 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).
- The agent meets reasonable time constraints and is able to complete a negotiation session within 3 minutes. Any agent that uses more than 3 minutes in order to initiate negotiation or to reply to an opponent's offer will be disqualified.
- It is mandatory to use the negotiation environment provided to you. This environment defines the 'rules of the game'.
- Make sure your agent is able to work on other domains than the Party domain, such as the Laptop domain, etc. Test your agent against different agents on varying domains and preference profiles.
- 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 every method. 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 "negotiation/group*n*/") in the right folder (the same directory in which the negotiation simulator `negosimulator.jar` can be found). If you submit the files in a .zip or .jar file, the root of the archive must only contain the folder negotiator, which only contains the folder "group" + n. Refer to Figure 3 for an example.
- The report should include:
  - the 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,
  - an explanation of the negotiation strategy, decision function for accepting offers, any important preparatory steps, and heuristics that the agent uses to decide what to do next, including the factors that have been selected and their combination into these functions,

- a section documenting the tests you performed to improve the negotiation strength of your agent. You must include scores of various tests over multiple sessions that you performed while testing your agent. Describe how you set up the testing situation and how you used the results to modify your agent.
- answers to the questions posed in the assignment, and
- a conclusion in which you summarize your experience as a team with regards to building the negotiating agent and discuss what extensions are required to use your agent in real-life negotiations to support (or even take over) negotations performed by humans.

2.4 Assignment

In this section, the main tasks and questions you need complete are presented. But before we do this, we present additional background that may help you to complete this assignment successfully. *Please start by making yourself familiar with the negotiation environment (start and use **negosimulator.jar**) and read the user guide (**userguide.pdf**) as well as this assignment provided to you carefully!* 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; in this practical assignment it is set to 30 minutes for negotiations that involve a human negotiator and it is set to 3 minutes for negotiations between two software agents. 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 0. In a sense, there is no 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. In a negotiation session between software agents, the agent that starts is determined randomly. In the tournament that will be played, each agent will negotiate with all other agents and the scores of each session are recorded and averaged to obtain an overall score for the agent (see also Section 5). A ranking will be compiled using these overall scores.

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 s/he has more of those in stock. They could, however, agree on the brand of laptop that they want to buy/sell. An outcome of a negotiation reconciles such differences and results in an agreed solution to resolve the conflict.

Ideally, such an outcome 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 one can do better or the same (i.e. they both score higher or equal, and therefore both 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. [7]). 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 since to reach an agreement is better than to reach no agreement. 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, as it is called, excludes certain outcomes as being unfair. It is, for example, very unlikely that your opponent will accept an offer that is most favorite to you and leaves your opponent with empty hands. In general, it is to be expected that an outcome is the result of a number of concessions that both parties make. Concessions can be made in various ways, quite easily or more slowly. The speed of making concessions, or the concession rate is the derivative of the (size of the) concession steps taken during a negotiation. An agent that makes concessions in very small steps initially uses a so-called Boulware strategy, but other strategies are conceivable and may be necessary given the deadlines (cf. [1]).

To compute whether a bid in a negotiation is Pareto optimal or a Nash solution we use so-called *utility functions* (cf. also [5]). In our case, utility functions assign quantative 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 4 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 $w_1, \ldots, w_4$) can be calculated by:

$$u(i_1, i_2, i_3, i_4) = w_1 * u(i_1) + w_2 * u(i_2) + w_3 * u(i_3) + w_4 * u(i_4)$$

There is only one complication that you will need to address in this assignment: there may be an additional *hard* constraint imposed on possible outcomes. A constraint is hard if any outcomes that do not satisfy such a constraint have no utility at all, i.e. 0 utility. In the laptop domain, for example, you can maximally spend 1200 Euro. *Please take into account that this significantly complicates finding a good negotiation outcome as well as defining a preference profile matching your preferences!*

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 4.

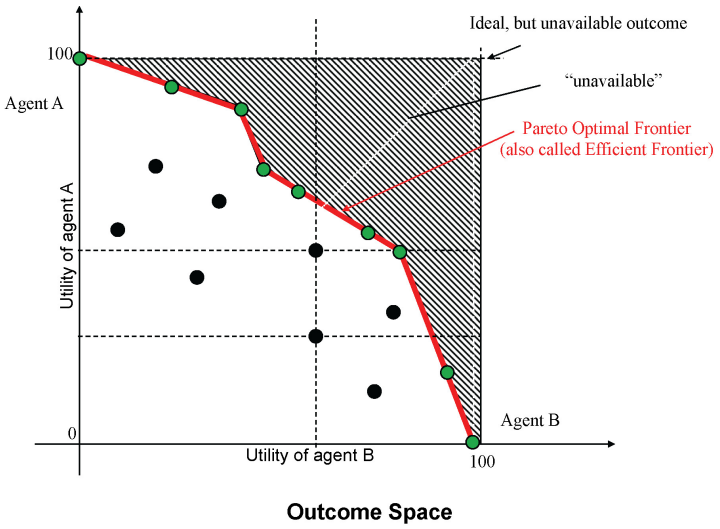

Figure 4: Pareto Frontier

What sets the negotiations considered in this assignment apart from other negotiations is the fact that both agents have exactly the same deadline for achieving a deal and both agents "know" this. For example, in the negotiation environment provided to you, any negotiation session between two software agents is limited to take at most three minutes in total.

The assignment consists of first familiarizing yourself with the negotiation environment and then designing and implementing a negotiating software agent.

2.4.1 Familiarizing Yourself With the Negotiation Environment

1. Read the documents provided to you, i.e. the User Guide for the negotiation environment, and this Assignment document. In order to familiarize yourself with the environment, complete the items below. This will help you understand how the negotiation environment works.

2. Inspect the negotiation templates named `laptop_domain.xml`, `laptop_buyer_utility.xml`, and `laptop_seller_utility.xml`. Start up the profile editor and load the domain and utility file for the buyer. Make a new preference profile for buting a laptop and edit the preferences according to your own preferences. *Take into account that the costs may not exceed 1200 Euro!* Save the results. (See the User Guide on how to do this.)

3. Start up the negotiaton simulator. Let the environment compute the utilities for all the possible bids in the outcome space and plot them on a graph such as Figure 4, using the seller utility space and

the buyer utility space created by yourself. You may want to exchange utility spaces created by your other team members and compare.

4. Load the `SimpleAgent` provided to you with the negotiation environment with the seller utility space and the `UIagent` with the profile created by yourself for the buyer. Play against this agent and familiarize yourself with the working of the simulator.

5. Analyze the performance of the `SimpleAgent` in the laptop domain by letting it play against itself. Is a Pareto optimal outcome reached? Explain your answer. Also explain why it obtains the resulting outcome that it does.

2.4.2 Design and Implement a Negotiating Software Agent

The remainder of this assignment concerns the design and implementation of a winning negotiating software agent. This negotiating agent will be tested on the party domain. We have included many different Party preferences that you can test your agent with. This will complete all the input that you will need for the tournament that will be played at the end of the assignment.

6. Preparation: Analyze Party Domain

   (a) Create a PEAS description (in the same format of a table as in [5]) of the negotiating agent that you need to design and implement in this assignment. Be as complete as possible and pay special attention to the performance measure of the agent. Also include a brief discussion about each element included in your PEAS description. In particular, explain informally what goals the agent has in a negotiation.

   (b) Design the overall structure and components of your agent. Describe the structure of your agent and the components it is consists of.

   (c) Inspect the negotiation template named `party_domain.xml` and the utility profiles provided to you on Blackboard (on 18 November). Compute and draw the efficient frontier in a graph such as Figure 4 for some of the templates provided to you.

   (d) Analyze the performance of the agent `SimpleAgent` in the party domain playing against itself. Is a Pareto optimal outcome reached? Explain your answer. Also explain why it obtains the resulting outcome that it does.

7. Design and implement a negotiation strategy.

   (a) Decide on the way your agent will set a *reservation value*. Explain in the report informally how your agent determines its reservation value and describe how you implemented this in your agent. In case your agent also uses other considerations to determine whether it will never accept particular bids, also explain these considerations.

   (b) Design a *decision function* which your agent uses to determine whether it will accept a bid or not. Describe which factors the agent takes into account for making this decision and how you implemented the decision function.

   (c) Decide on the way your agent will compute a *counter offer* (a follow-up bid). Explain in the report informally 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.

   (d) Does your agent ever propose offers that *increase* the utility of the offer again compared with previous offers made? 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.

8. Play against your own agent.

(a) Have your agent negotiate against itself, and against *all* agents provided to you on the party domain. Run several negotiation sessions using various utility profiles. 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?

(b) 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 *all* agents provided to you on the *laptop* and *inheritance* domain (see the xml templates provided to you, and make sure that you have utility files for the inheritance domain). Run several negotiation sessions using various utility profiles. 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?

2.4.3 Concluding: Future Perspectives

9. As you will have noticed by now, it is hard to find deals that are Pareto optimal. An alternative for one-to-one negotiation as used in this practical assignment is to use a trusted mediator to which each agent declares its private preferences. The mediator then will compute an efficient outcome using this input. Do you think that introducing a mediator is a better option for finding an optimal outcome for both agents? If so, argue why you think so and describe the circumstances in which it is particularly suitable to use a mediator. If not, argue why you think it is not useful to do this.

10. The agents in the negotiation environment have limited capabilities in order to achieve a negotiated deal. Think about capabilities (e.g. other actions or forms of communication) that an agent might be extended with. Can you think of additional capabilities that would help an agent to achieve a better deal? If so, explain why these capabilities would help an agent to achieve a better deal. If not, argue why there will not be any capabilities that could help an agent achieve a better deal.

As a final remark, we want to make clear that it is very important in order to improve the negotiation strength of your agent and to identify potential methods to improve it to extensively *test your agent*. A warning is in place though: do not assume that your goal is to outperform a 'stupid' agent such as the one provided with the negotiation environment. 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. You can run your agent against itself, or, for example, against the simple agent provided to you with the negotiation environment or against earlier versions of your own agent. You may also agree to run it against an agent of another team. In general, 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, e.g. time an opponent takes to reply with a counter offer, consistency of an opponent's offers, concession rate of your opponent, history of previous negotiation sessions, and possibly other considerations about the domain of negotiation itself. Of course, also use the suggestions and literature references provided to you elsewhere in this document.

3 What is provided to you as a start?

The following items are provided to you to help you complete the assignment.

- This document, called `negotiation.pdf`,
- A user guide, called `userguide.pdf` for the negotiation environment,
- A negotiation environment called Genius which implemented in JAVA, including an example agent called `SimpleAgent`,
- A set of XML *negotiation templates*.

4 Organization

Important Dates:

1. **19 November, 15.45-17.45**: Deadline for registering your group. Please send an e-mail to

   `ai@mmi.tudelft.nl`

   listing your group members, and we will assign a group number to you.

2. **17 January, 09.00**: Deadline for submitting your team solution including a negotiating software agent and report.
   Please submit your report in PDF format and the package for your negotiating agent by mail to `ai@mmi.tudelft.nl`. 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.

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 assignment 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, quality of documentation,
- *Performance:* Agents will be ranked according to negotiating strength that is measured in a tournament (see also the next section below),
- *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 e.g. other students will not be evaluated and be judged as insufficient. Detection of fraud will be reported to the administration.

5.1 Competition

Agents of teams will be ranked according to negotiation strength. The ranking will be decided by playing a tournament in which every agent plays negotiation sessions against every other agent on the party domain. However, your agent should be generic enough to play on any domain. Note that certain preference profiles can be assigned to your agents randomly. The agreements reached will be used to rank agents, based on the utility value associated with that agreement (normalized to make sure values range from 0 to 1 and to ensure fairness; i.e. if a utility profile is such that a maximum utility of less than 1 can be achieved this will be taken into account).

5.2 Grading

Final grades will be determined as a weighted average of several components. The final grade will be determined by your team solution (i.e. your assignment, the performance of your negotiating agent and report).

The components that are graded and their relative weights are described in Table 1 below.

Assignment	Grading Method	Weight
Negotiating Agent	Performance in a tournament with other agents.	50%
Report	Agent design and the final report about your negotiating agent.	50%

Table 1: Grading criteria and weight

5.3 Master Thesis about negotiation

Did you like thinking about efficient negotiating strategies, or implementing a successful negotiating agent? Negotiation provides a lot of subjects to do your Master Thesis on! Please have a look at our website at `http://mmi.tudelft.nl/negotiation`. You can also have a look at the last page of this assignment, or contact us for more information: `negotiation@mmi.tudelft.nl`.

References

[1] S. Shaheen Fatima, Michael Wooldridge, and Nicholas R. Jennings. Optimal negotiation strategies for agents with incomplete information. In John-Jules Ch. Meyer and Milind Tambe, editors, *Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, pages 53–68, 2001. URL = citeseer.ist.psu.edu/fatima01optimal.html, November 2006.

[2] Catholijn M. Jonker and Jan Treur. An agent architecture for multi-attribute negotiation. In *IJCAI*, pages 1195–1201, 2001. URL = citeseer.ist.psu.edu/jonker01agent.html, November, 2006.

[3] http://en.wikipedia.org/wiki/negotiation. November 2006.

[4] N.R. Jennings P. Faratin, C. Sierra. Negotiation decision functions for autonomous agents. *International Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1998.

[5] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[6] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.

[7] Roberto Serrano. Bargaining, 2005. URL = http://levine.sscnet.ucla.edu/econ504/bargaining.pdf, November, 2005.

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

Negotiation has become a major and interesting research area within Artificial Intelligence. Negotation 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, either related to a specific domain or not,
- 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,
- Design of a testbed for negotiating agents,
- Research related to negotiation, trust and the reputation of agents, either cognitively motivated or viewed from a more engineering point of view,
- Research on negotiation and culture. How does culture influence negotiation between different agents?
- Etc.

If you are interested, don't hesitate to ask us: `negotiation@mmi.tudelft.nl`