**To Do:**

- Test profile creation with staff/some students.

Done Include accurate user manual for profile selection.

Done Integrate last year's assignment.

- Fix references
- Recheck Agent Creation part

Done Reduce alcohol enthusiasm

Done Include music issue

Done Make table of issues

Done Drop who does what issues

Done Move profile options to appendix (separate file)

- Planning: 1 hour HH/1 hour HC back-to-back?
- Manual for H2C negotiation.
- Bibliography

# IN4010TU Practical Assignment Q2: Negotiation

Catholijn Jonker, Koen Hindriks, Dmytro Tykhonov, Bart Grundeken

29th October 2007

## 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 ocassion. 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 up to €1200.

This document describes one of the two assignments that can be chosen to complete a practical assignment for the second quarter of the AI course. The practical assignments aim at familiarizing students with parts of the course material in a more practical way. This assignment is about *negotiation.* First, you will have to think about your preferences and define a preference profile. Then, you will negotiate with another student about the party, where the goal is to reach an agreement that reflects your preferences as much as possible. Next, you will negotiate with a computer agent to see how good you fare against a machine. Finally, you will design and implement a negotiating agent in JAVA. Techniques relevant for building and information about designing negotiating agents can be found among others in chapters 16 and 17 in [?]. The assignment will also require you to go beyond the course material in the book.

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 looser 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 may be quite different from negotiation to negotiation.

Orient yourself towards a win-win approach.

Plan and have a concrete strategy.

Determine what 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 1: Negotiation Guidelines

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 pre-negotiation phase has been completed and you cannot redefine the domain anymore. There are two activities that are usually considered part of the pre-negotiation 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. This profile will be used in grading so it is probably best for you to be true to yourself when defining it. 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.

Finally, Figure 1 provides some initial guidelines based on human experience with negotiation that may help you during your own negotiations and in building your own software negotiating agent.

The remainder of this document is organized as follows. In Section 2 the organizational details (grading, deadlines, etc.) are outlined. Following that, there are sections detailing each of the assignment phases: profile creation in Section 3, human-to-human negotiation in Section 4, human-to-machine negotiation in Section 5, and, finally, agent creation in Section 6.

| Assignment | Grading Criteria | Weight |
|---|---|---|
| Human/Human Negotiation | How well the eventual agreement reflects your profile, eg how high your utility is with the agreement. If no agreement is reached, the grade will be a 1. | 1 |
| Human/Machine Negotiation | Idem, only now you face an electronic negotiation partner. | 1 |
| Negotiating Agent Creation | Agent design and performance in an environment with other such agents. | 2 |

Table 1: Grading criteria and weight

## 2 Organization

The Q2 practical consists of three assignments:

1. Human-to-Human Negotiation
2. Human-to-Machine Negotiation
3. Negotiating Agent Creation.

### 2.1 Grading

For each of the assignment you will receive a grade. Your total grade for the Q2 practical will be a weighted average of these grades. Table 1 contains what will be graded with each assignment and how much weight is attached to those grades.

### 2.2 Dates and Locations

The hands-on negotiation session will take place at 9:00 (AM), November 14th on the Drebbelweg, room DW01_180. Here, you will determine your profile, negotiate with another student and with a software agent. For your agent design and implementation, the deliverables must be send by email to `ai@mmi.nl`. What these deliverables are is detailed in Section 6. The **deadline** for this last assignment is **December 16th, 23:59**.

### 2.3 Assistance

Assistance will be provided on site for the hands-on session. For assistance with the negotiating agent, direct your queries, as usual for the Q1 practical and homework assignments, to `ai@mmi.nl`.

## 3 Profile Creation

Profile creation will be done with a computer program, installed on the computers on the Drebbelweg. During the hands-on session, you will be handed the profile choices. For each category you must determine the following.

1. Your utility value for each option in that category. This indicates which option you prefer for each category.

2. The relative importance of that category to you (weight). This indicates how much you care for a certain category. So if you are only interested in food, but could not care less about music, food will have a high weight for you and music a low weight.

You will both fill in these values on paper and in a computer; giving you a reference sheet for during negotiations as well as an XML version of your profile. The latter is the single deliverable of this phase.

As you generate your profile, try to make such choices as you would in real life, given the same options. During negotiations, will have no written or printed reference to your profile available, so you must memorize it; it is thus easier to take choices you can easily remember because they reflect your real preferences. Also bear in mind the grading: you must maximize your utility in negotiations to get a good grade, which counts for 25 percent of your total grade. Therefore, assigning your utility such that you can never get all your preferred options because together they exceed the budget is not a good idea.

What follows now is a step-by-step description on how to create your profile. This profile is generated as an XML file and you should submit it to the instructors. How you submit it is explained below. For all numerical values, use positive integers. Do not change anything but the sliders and your utility values. Changing anything else will result in an unusable profile for you, and than you cannot continue with the remainder of the assignment. Should an accidental change occur, contact one of the instructors.

1. Start up your computer to Windows, then double-click the ”Profile Creator” icon on your desktop.

2. Click on File>Open Domain, and select the ”party.xml” file on your desktop.

3. Click on File>Open Utility Space and select the ”profile.xml” file on your desktop.

4. For each issue, select it, then click ”Edit”.

4a. Select the option in question.

4b. Enter the utility for each option in the "Evaluation" column, the same value as on your hand-written profile.

4c. After you have entered your utility for each option, click "Ok". You are returned to the domain screen.

5. Now, for each issue, indicate the relative weight by using the sliders next to that issue. You can lock the sliders in place by clicking the checkbox next to the slider. Note that you have to enter normalized values, and the total value of the sliders must add up to 1.

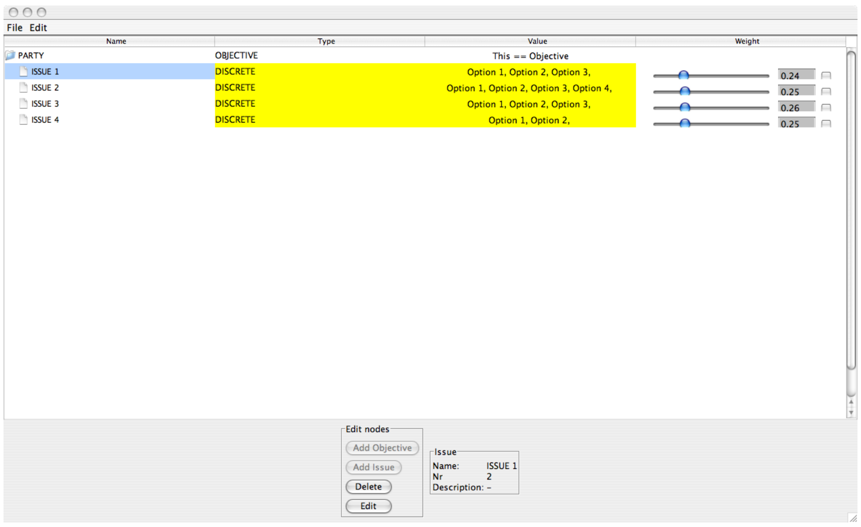
6. Once you have set all sliders and have filled out the evaluations for the options under each issue, click File>Save Utility Space, and save your file on the desktop. You can use "profile.xml" if you want to.

7. Quit the application with File>Quit.

8. Mail you profile to `ai@mmi.nl`, clearly stating your name and student ID.

## 4 Human-to-Human Negotiation Round

After profile selection, there is a period of **TODO: still correct?** one hour where you will negotiate with a **randomly assigned** partner. Within that period you will have to arrive at an agreement that resembles your profile as closely as possible. You will then hand in your agreement. Note that it must be as complete (eg. for all categories you have selected an option)



| Name | Type | Value | Weight |
|---|---|---|---|
| PARTY | OBJECTIVE | This == Objective | |
| ISSUE 1 | DISCRETE | Option 1, Option 2, Option 3, | 0.24 |
| ISSUE 2 | DISCRETE | Option 1, Option 2, Option 3, Option 4, | 0.25 |
| ISSUE 3 | DISCRETE | Option 1, Option 2, Option 3, | 0.26 |
| ISSUE 4 | DISCRETE | Option 1, Option 2, | 0.25 |

Issue
Name: ISSUE 1
Nr 2
Description: –

Figure 2: The domain screen.

as possible and must fit within the budget; your grade is influence by these factors.

You may not explicitly reveal your profile to your opponent. Also, the assignment falls under the normal rules for fraud: so bribery with anything from outside the domain is not allowed. Naturally, badgering your opponent physically or vocally is also not allowed.

## 5 Human-to-Computer Negotiation Round

After human-to-human negotiation, the remainder of the time you will negotiate with a software agent (which is assigned a random profile). Like human-to-human negotiations, try to reach a complete agreement within budgetary bound

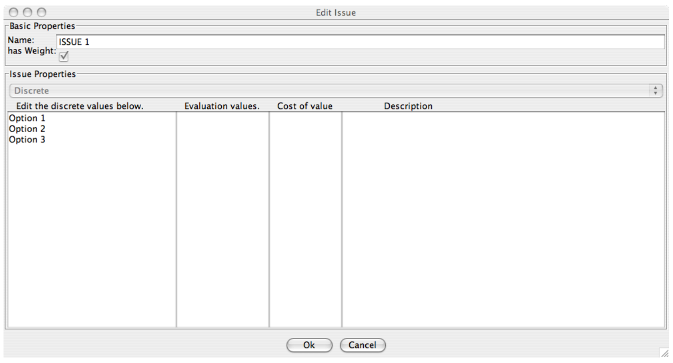
**TODO: Specifics: interface use**

## 6 Creating Your Own Negotiating Agent

The assignment may be completed in teams of 4-5 students. In the sections below the objectives, deliverables, and requirements

### 6.1 Objectives

- To learn to design a negotiating agent, 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.



Basic Properties

Name: ISSUE 1

has Weight: ✓

Issue Properties

Discrete

| Edit the discrete values below. | Evaluation values. | Cost of value | Description |
|---|---|---|---|
| Option 1 | | | |
| Option 2 | | | |
| Option 3 | | | |

Figure 3: The utility screen.

## 6.2 Deliverables

- A unique name 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 "negotiation.group" + ♯your_agent for the negotiating agent and any other required files.
- Your (programmed) solution for computing and picturing the outcome space (including Pareto Optimal Frontier).
- A report (in PDF!) 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 explaination 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.

## 6.3 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 2 minutes. Any agent that uses more than 2 minutes in order to initiate negotiation or to reply to an opponent's offer will be disqualified.

- The source code has to be written in Java 1.5. Moreover, it is mandatory to use the negotiation environment provided to you.
- 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.
- 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 in the right folder (which needs to comply with the directory structure: "negotiation.group♯.YourAgentHere".
- The report should include:
  - 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.

Please remove all debug printouts from your code, and make sure that your agent does not flood the 'System.out' channel.

Note that the report should help the reader to understand the organization of the source code (details should be commented on in the source code itself)

## 6.4 Assignment

In this section, the main questions you need to answer are presented. But before we do this, we present additional background that is required for completing the assignment successfully. Please also review some additional constraints and assumptions summarized at the end of this document. These constraints and assumptions may also help you to clarify the content and structure of the assignment.

A negotiation in this assignment may consist of multiple *negotiation sessions*. In each session, two agents negotiating with each other need 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 2 minutes. At the end of each 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. The agent that starts a negotiation is determined randomly for each negotiation session. In the tournament that will be played, each agent will negotiate 5 sessions with all other agents and the scores of each *final session* are recorded and averaged to obtain an overall score for the agent (see also below). A ranking will be compiled using these overall scores (and a bonus point will be assigned to the group with the highest ranking).

The main reason for multiple negotiation sessions is that agents may learn about their opponent from the sequence of bids exchanged and from the negotiation outcome (e.g. deal or no deal). Exchanging offers allows the parties to learn about the others' limitations, and to identify the key issues and critical areas of disagreement. During this phase, the parties realize the potential of a compromise and can assess its main features. The analysis of a negotiation may focus on the modification of strategies, the determination of concessions, and on the restriction of efficient solutions to those which may be acceptable to the parties. A second reason for having multiple sessions (where in the tournament only the last session will be scored) is that it will sometimes be useful to break off the negotiation to signal dissatisfaction about the way negotiation proceeds to the other agent.

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 choosing a type of restaurant both agents may not agree about their first choice but may agree on their second choice. They could, however, be in serious conflict over the location (e.g. city) where they would like to have diner. 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

Figure 4: Pareto Frontier **TODO: Picture was not included**

is that the outcome should be Pareto optimal. An outcome is Pareto optimal when there does not exist a deal where both agents do better (i.e. they both score higher 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 of the outcome for the first agent times that for the second agent is maximal (cf. [**?**]).

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. [**?**]).

To compute whether a bid in a negotiation is Pareto optimal or a Nash solution we use so-called *utility functions* (cf. also [**?**]). 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 $\mathit{bid} = \langle i_1, i_2, i_3, i_4 \rangle$ be a particular bid. Then the utility $u(\mathit{bid}) = u(i_1, i_2, i_3, i_4)$ (given weights $w_1, \ldots, w_4$) can be calculated by:

$$u(i_1, i_2, i_3, i_4) = w_1 * u_1(i_1) + w_2 * u_2(i_2) + w_3 * u_3(i_3) + w_4 * u_4(i_4)$$

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.

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. In the negotiation environment provided to you, any negotiation session between two agent is limited to take at most two minutes in total.

The assignment consists of two parts. The first part concerns so-called *fair division problems* ([?]). The second concerns so-called *multi-issue negotation.* A fair division problem can be viewed as a simple instance of the more general class of multi-issue negotiations with multiple values. Fair division problems thus provide for a more simple setting to think about and study the basic negotiation concepts and techniques than general multi-issue negotiation with multiple values. In the second part the complexity of issues with multiple values will need to be dealt with. The examples used in the assignment are based on [?].

Before we introduce the two parts of the assignment, you are asked to answer the following questions.

1. Create a PEAS description (in the same format of a table as in cf. [?]) 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.

2. Design the overall structure and components of your agent. Describe the structure of your agent and the components it is consists of.

### 6.4.1 Fair Division Problems

A fair division problem concerns the division of a number of items among two (or more) agents. Typical real-life examples of such negotiations are the division of an inheritance. Each agent that takes part in such a division problem has its own preferences about obtaining some item. Some of these may be similar to other agents. For example, two agents may both prefer most to obtain the antique chinese vase. Since we only consider bilateral agents in this assignment we will only look at the two-party case. In all negotiations that we will consider you may assume that agents need to settle multiple issues.

3. Inspect the negotiation template named `inheritance.xml` and utility templates `inheritance_Janet.xml` and `inheritance_Marty.xml`. Compute the utilities for all the possible bids in the outcome space and plot them on a graph such as Figure 4. Draw the efficient frontier. Also compute the Nash solution. Submit your solution for computing and picturing the outcome space with your report.

4. Analyze the performance of the simple agent `MyAgent` in the folder `agentExample` provided to you with the negotiation environment on the negotiation template `inheritance.xml`. Load the negotiation template into the negotiation environment. This will automatically

select and load agent `MyAgent` for both agent A and agent B into the negotiation environment, as well as load the two utility profiles. Start a negotiation session. Is a Pareto optimal outcome reached? Explain your answer. Also explain why it obtains the resulting outcome that it does.

5. 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.

6. Experiment: Negotiate against own agent.

   (a) Assume there are only two items (or issues), called *blue* and *yellow*, that need to be divided between two agents. Create a negotiation template called `two_issue.xml` with just these two issues by modifying a previous template. The values of these issues should be `Janet` (for agent A) and `Marty` (for agent B). Create a utility template called `Janet_two_issue.xml` which associates a utility of 70 with issue *blue* if Janet gets it and 0 if Marty gets it, and a utility of 40 with issue *yellow* if Janet gets it and 0 if Marty gets it. Create a second utility template called `Marty_two_issue.xml` which associates a utility of 50 with issue *blue* if Marty gets it and 0 if Janet gets it, and a utility of 80 with issue *yellow* if Mart gets it, and 0 if Janet gets it. Change the agent name (to your agent's name) and the utility space labels in the negotiation template accordingly. Load the negotiation template into the negotiation environment. Run five negotiation sessions in which your software agent negotiates with itself using each a different profile though. Report the outcome. Explain why the outcome is as it is. Is any of the outcomes a Nash solution? Is it possible to reach a non-efficient outcome, i.e. an outcome that does not lie on the Pareto Frontier?

(b) Change the negotiation template `inheritance.xml` such that instead of `MyAgent` it loads your agent. Call this template `inheritance`+your agent name+`.xml`. Analyze the performance of the agent by running at least five negotiation sessions. Is a Pareto optimal outcome reached? Explain your answer. Also explain why it obtains the resulting outcome that it does.

### 6.4.2 Negotiating Multiple Issues with Multiple Values

7. Inspect the negotiation template named `AMPOvsCity.xml` and utility templates `AMPO.xml` and `City.xml`. Compute and draw the efficient frontier in a graph such as Figure 4. Also compute the Nash solution. Your (programmed) solution should be submitted with your report.

8. Analyze the performance of the simple agent `MyAgent` in the folder `agentExample` provided to you with the negotiation environment on the negotiation template `AMPOvsCity.xml`. Load the negotiation template into the negotiation environment. Start a negotiation session. Is a Pareto optimal outcome reached? Explain your answer. Also explain why it obtains the resulting outcome that it does.

9. Design and implement a negotiation strategy.

   (a) Adapt the functions that you implemented in your negotiating agent such that it can handle multiple values for multiple issues. That is, modify the way your agent determines a reservation value, adapt its strategy and its decision function for accepting to handle multiple values for issues.

   (b) Run your adapted agent again on the inheritence example. That is, using your new agent, answer questions 6b and 6a again. Does your new agent reach the same outcomes? If not, analyze the differences and explain why these differences result.

10. Play against own agent.

ided; Still needs to be MODIFIED Assume there are only two issues, called *typeOfFood* and *location*, about which two agents need to negotiate to fix particular values.

Create a negotiation template called `two_issue.xml` with just these two issues by modifying a previous template. The values of these issues should be respectively `Chinese`, `French`, `Indian`, `Italian`, `Pub`, `Thai`, `Turkish` and `Amersfoort`, `Amsterdam`, `Delft`, `DenHaag`, `Gouda`, `Rotterdam`, `Utrecht`. Create a utility template called `Janet_two_issue_mv.xml` which associates the following utilities with the list of values for the first issue: 70, 60, 30, 25, 40, 10, 85 and the following utilities with values for the second issue: 100, 20, 5, 10, 10, 10, 75. Create a second utility template called `Marty_two_issue_mv.xml` which associates the following utilities with the first issue: 20, 55, 80, 50, 40, 60, 20 and the following utilities with the second issue: 20, 90, 50, 50, 10, 30, 30. Change the agent name (to your agent's name) and the utility space labels in the negotiation template accordingly. Load the negotiation template into the negotiation environment. Run five negotiation sessions in which your software agent negotiates with itself using each a different profile though. Report the outcome. Explain why the outcome is as it is. Is any of the outcomes a Nash solution? Is it possible to reach a non-efficient outcome, i.e. an outcome that does not lie on the Pareto Frontier?

(a) Change the negotiation template `AMPOvsCity.xml` such that instead of `MyAgent` it loads your agent. Call this template `AMPOvsCity`+your agent name+`.xml`. Analyze the performance of the agent by running at least five negotiation sessions. Is a Pareto optimal outcome reached? Explain your answer. Also explain why it obtains the resulting outcome that it does.

### 6.4.3 Concluding: Future Perspectives

10. 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.

11. 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 strenght 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.

## 6.5 Implementation

### 6.5.1 What is provided to you as a start?

The following file is provided to you to complete the assignment, which can be downloaded from Blackboard:

- A compressed file `negotiator.zip`, which contains:
  - This assignment file called `negotiation.pdf`,
  - The negotiation environment implemented in JAVA, including an example agent `SimpleAgent`,
  - An example negotiating agent JAVA file, called `SimpleAgent.java`, located in the subfolder *negotiation/group*0.
  - A set of *negotiation templates* in that can be found in the directory `templates`.

In case you need more information about JAVA programming, please use the following link: http://java.sun.com/docs/books/tutorial/index.html.

### 6.5.2 Negotiation Environment

The negotiation environment can be started by:

```
java -cp negotiator.jar negotiator.Main
```

or by using the batch file `run.bat` on a Windows machine.

The screen that pops up when the negotiation environment is started allows you to:

- Load a negotiation template, which will automatically also load agents and utility profiles,
- Load two agents, one playing the role of agent A in the negotiation, the other playing that of agent B,
- Load two utility templates, one associated with agent A and one associated with agent B.

**Adding an Agent** Loading an agent can be done by specifying the full name of the agent, prefixed by its package name. If you are in group NR, and your agent is called MyAgent, and it resides in package "negotiation.groupNR", you can add it by entering "negotiation.groupNR.MyAgent" in the corresponding text field. *Your actual group number is determined by the number you received while registering on http://dublin.twi.tudelft.nl/webapp/ for the Practical. Make sure you use this number to submit your deliverable.*

**Starting a Negotation** You can start a negotiation between any two available agents by loading these agents and by clicking on the start button on the bottom left side of the interface. The outcome of a negotiation will be shown in the [... interface]. The [...] is a special agent that is not bothered by any time restrictions, and allows human users to negotiate against a software agent. It is not possible to select who will make the first offer in a negotiation, this is randomly allocated.

**Negotiating Multiple Sessions** In order to put your agent to a test, it is interesting to see how it performs against a number of other agents in a number of negotiation sessions. The negotiation environment does not provide a special tournament mode, but does allow you to set the number of sessions that two agents are supposed to play against each other. A history over multiple sessions may be maintained by either agent in such a run.

### 6.5.3 Implementation instructions

In the subfolder "negotiation", all required classes for the negotiation environment can be found.

A negotiating agent will receive a [...] object through [...]. It has to reason about the best next move, and return a [...] object . The [..] object contains all the necessary information about the current negotiation state. It contains the [...] and a Boolean indicating whether the negotiation has finished.

An important consideration for the implementation is that an agent may participate in multiple negotiation sessions. For this reason, a [Id] field is included. This is a unique number identifying the current negotiation session. [Include this id in the [...] object.]

An example agent has been included in the installer, and can be found in the subfolder negotiation/group0. This [Agent] can be used as a start for your implementation. It is does not perform well in a negotiation, as it will always attempt to propose as next bid that concedes minimally, by selecting randomly an issue on which to concede. Moreover, it only takes the order of the issue values into account and does not consider the associated utility for the agent.

## 6.6 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 explanation provided to the questions posed in the assignment; Explanatory comments in source code, quality of documentation,

- *Performance:* Agents will be ranked according to negotiating strenght that is measured in a tournament (cf. also the next section below),

- *Originality:* Any original features of the agent or solutions to questions posed are evaluated positively,

### 6.6.1 Competition

Agents of teams are ranked according to negotiation strength. The ranking will be decided by playing a tournament in which all agents play 5 negotiation sessions against each other. In case there will be many teams choosing this assignment, teams may be split up in two or more pools in which agents play against each other. In that case, a winner will be determined in a second round in which pool winners play against each other. The members of the team that designed the highest ranked negotiating agents in the competition will be awarded with 1 bonus point for their exam out of a maximal score of 10. The exact number of teams that will receive a bonus will depend on the total number of teams that sign up for this assignment and will be announced when this information has been collected.

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 fraude will be reported to the administration.

Negotiation has become a major and interesting research area within Artificial Intelligence. Negotation is important in many domains ranging from business applications such as Internal 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 the possibility of doing a Master Thesis about negotiation and there are many angles that you can take, e.g.

- Design of a negotiation support system, either advicing humans in a particular domain, or even taking over negotiation all together,
- Design of a testbed for negotiating agents,
- Design of negotiation strategies, either related to a specific domain or not,
- 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!