COMP6203-2021/22 Intelligent Agents Coursework Specification

Enrico H. Gerding, eg@ecs.soton.ac.uk

Updated on: November 9, 2021

Deliverable	Deadline	Feedback	Marking Scheme	Weight
Negotiation Agent	Tuesday, Week 11, Dec 14, 4pm	Jan 11	The score that your agent achieves in the class tournament will determine 15% of the total module mark. The score will be an average of the performance based on multiple criteria and negotiation scenarios.	15%
Group Report	Tuesday, Week 12, Jan 11, 4pm	Feb 8	Top scoring reports will describe in detail the challenge that the agent faces, and the design of the strategy implemented. They will present qualitative and quantitative analysis of the agents performance, and will show evidence that related literature has been read, understood and applied.	25%

Table 1: Deliverables and Deadlines

Contents

1 Introduction

This assignment is about building your own *negotiation agent*. Negotiation is a form of interaction in which two (or more) parties, with conflicting interests and a desire to cooperate, try to reach a mutually acceptable agreement. Negotiation between parties can in many ways be modeled as a game, and game theory is useful to analyze the behavior of negotiating parties. Negotiation is, however, also different from many board games such as chess. 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 parties can gain by obtaining a deal through negotiation. Another difference is that the domain of negotiation (i.e. what is being negotiated) may be quite different from one negotiation to the other. Finally, in negotiation there is a lot of *uncertainty*: a negotiation agent typically does not know the negotiation strategy used by its opponent, or even know what outcome the other agent prefers. In fact, as we will see, an agent may not even know its own preferences exactly.

For this assignment you will build your own negotiating software agent to participate in the agent negotiation competition, which uses the GENIUS framework. The implementation is in Java and you will work in a team of up to 4 people. In addition, you will need to submit a group report describing the agent and performing an analysis of the agents. See Table 1 for the hand in dates.

2 Negotiation Setup

The negotiation will run using the GENIUS framework and the setup is based on the rules from the International Automated Negotiating Agents Competition (ANAC). It will be a **bilateral** negotiation between **2 agents,** using the **Stacked Alternating Offers Protocol** (SAOP). Negotiations are conducted using various multi-issue preference domains, which are described by additive weighted utility functions. In the tournament setup there is *preference uncertainty*, i.e. the agents have uncertainty about their own preferences. Specifically, the agents will have a partial ranking of outcomes, and no access to their entire utility function. However, the agent can elicit additional information from a virtual user at a fixed cost (see Section 10.5 for more details). Also, the agents will not have any knowledge of the preferences of its opponent. Negotiation is round based (as opposed to time based) and each negotiation lasts **200 rounds**. In addition, there will be a time out after a certain period of time. Finally, agents have *reservation values* which is the utility received by the agents when no agreement is achieved. Reservation values of agents can be different and agent does not know the reservation value of its opponent. There will not be any discounting factors used in the final competition (i.e. the discounting factors are set to 1).

3 Submitting the Agent

Please follow the next instructions carefully or your agent will not work and so the group will receive no marks for agent performance. All your code should be placed in a *package* called `group`*n*, where *n* is the group number that will be allocated to you.[1] The main agent class (containing an implementation of the methods `chooseAction`, `receiveMessage`, etc.) should be named `Agent`*n*, where *n* is the group number. For example, for group 5, the resulting fully qualified name of the agent object becomes `group5.Agent5`.[2].

You then need to produce a JAR file containing all the relevant binaries (`.class` files) to run the agent as well as the corresponding source code (`.java` files). You should *not* include any files or classes that are already part of the genius framework (such as `genius-xxx.jar`). The name of the JAR file does not matter as long as it has the jar extension.

[1] If you are not familiar with the concept of a java package, check out various online tutorials such as `https://www.w3schools.in/java-tutorial/packages/`.

[2] Note that the dot here indicates that object `Agent5` is in package `group5`, and the term "fully qualified name" refers to the complete path including all the packages.

Most IDEs such as Eclipse support the generation of JAR files. However, you can also do this on the command line by using the program `jar.exe`. If this executable is not found, this is typically located (on a PC) inside the Program Folder, e.g. `C:\Program Files\Java\jdk1.8.0_60\bin\jar`. The exact directory will vary depending on the version of java that you have installed.

For example, for group 1, if your source files are in a folder called `src` and the binaries in a folder called `bin`, then you can create the jar file by the following steps:

1. Create a new `group1.jar` file and add the source code files by using the *create (c)* option as follows:

```
jar cvf group1.jar -C src .
```

2. Next, add the binary files using the *update (u)* option:

```
jar uvf group1.jar -C bin .
```

3. Finally, check that all the necessary files are there:

```
jar tvf group1.jar
```

When executing these commands, the output should look something like this:

```
F:\OneDrive - University of Southampton\Workspace\genius-agent>jar cvf group1.jar -C src .
added manifest
adding: group1/(in = 0) (out= 0)(stored 0%)
adding: group1/MyAgent.java(in = 2446) (out= 971)(deflated 60%)

F:\OneDrive - University of Southampton\Workspace\genius-agent>jar uvf group1.jar -C bin .
adding: group1/(in = 0) (out= 0)(stored 0%)
adding: group1/MyAgent.class(in = 2830) (out= 1363)(deflated 51%)

F:\OneDrive - University of Southampton\Workspace\genius-agent>jar tvf group1.jar
     0 Tue Sep 17 16:45:24 BST 2019 META-INF/
    68 Tue Sep 17 16:45:24 BST 2019 META-INF/MANIFEST.MF
     0 Tue Sep 17 16:43:46 BST 2019 group1/
  2446 Tue Sep 17 16:43:44 BST 2019 group1/MyAgent.java
  2830 Tue Sep 17 16:43:46 BST 2019 group1/MyAgent.class
```

In particular, make sure that both the *java* and *class* files are included and they both within the *group1* directory (if your group is 1).

Finally, the JAR file needs to be submitted through the *handin* system by the deadline. **Only one agent per group should be submitted.** You can submit multiple times. The most recent valid submission will be the one used in the final competition, irrespective of who uploads the agent.

Submission Check List. When submitting your agent make sure that:

- The agent object has the appropriate name.
- The agent object is contained in the appropriate package.
- The JAR file contains both the sourcecode and binaries necessary to run your agent.
- You have NOT included any jar files or classes from the GENIUS platform.

4 The Tournament

The submitted agent will participate in a tournament with other agents from the module. Given the number of participants it may not be possible for each agent to play against all other agents. In that case, each agent will be randomly grouped with other agents to form a tournament. Within each tournament, each agent will play against each other on several domains in the configuration as mentioned above. Your agent

should be generic enough to play on different domain sizes and work with different degrees of preference uncertainty. In this year's tournament only domains with discrete issues will be used during the competition (see the GENIUS manual for what these terms mean). Domain sizes will vary from very small (less than 10 different offers possible) to very large (up to 25,000 possible offers).

Each agent will participate in thousands of negotiations against agents designed by other teams in the same cohort. The agent's mark is proportional to its average performance. This performance is based on two factors: (1) the individual utility achieved, and (2) the Euclidean distance of the agreement from the Nash bargaining solution. The latter is given by:

$$\sqrt{\sum_{i=1}^{m}(U_i(o*) - U_i(o_{NBS}))^2}$$

where $U_i(\cdot)$ is the utility for agent i for an offer (and the reserve price in case of a disagreement), $o*$ is the agreed offer (or a disagreement), o_{NBS} is the Nash bargaining solution (see Section 10.2), and m is the number of agents (normally 2).

In addition, the agents will be tested on domains of different sizes. For each of the two categories and for each domain, the mark will be scaled such that the best agent (i.e. the highest individual utility for the first category, and the lowest distance for the second category) will receive the maximum score and the average score across the cohort will be around 65%. The score will be scaled separately for each category and domain. Therefore, one group might receive the highest score for the individual utility category, whereas another group might have the highest score for getting the lowest distance. Both categories have equal weight.

Agents may be disqualified and receive zero marks 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.

5 The Report

Each group needs to submit a report of up to 2500 words excluding abstract, tables, figures, references and appendix (this is an upper limit and should not be a goal) and no more than 6 double column pages in total including references and appendices. Any report exceeding the word limit will be penalised and those exceeding the page limit may not be marked. The report should be written in the style of a research paper and describe at a minimum:

1. The overall design of the agent including various aspects of the strategy (e.g. the concession strategy, how your agent deals with preference uncertainty, the acceptance strategy, etc), using pseudocode where applicable.

2. The reasons behind the specific design of the strategy that it employed, and how it compares to and/or varies from existing approaches. It should refer to relevant literature as appropriate.

3. An evaluation of the agents performance based on your chosen metrics, which compares your agent against some benchmarks (e.g. other agents).

4. A critical evaluation of the agent and how this can be improved.

5. A statement regarding the contribution of individual team members (in case of non-equal contributions).

You are encouraged to evaluate different variants of the agent strategy and use this to motivate the final design decisions, and/or how your submitted agent could be improved. You could also evaluate specific aspects of the strategy, e.g. the utility estimation approach. It is expected that the report will contain several graphs and/or tables showing the results of the analysis. It should also cite relevant literature especially for motivating the design choices but also comparing and contrasting the strategy to existing literature. Note that there is no need to describe the competition itself in detail. A brief overview may be included or if specific details of the competition are needed to motivate the strategy.

The report should contain the author names, student numbers, group number, word count, and be formatted as a double column academic paper. You will find a template for this on the course website. Please see Table 2 for an indicative marking scheme.

6 Individual Contribution

To recognize that the contributions of each individual in the team may vary, teams have the option to specify different individual contributions of each group member in terms percentages, which will have a corresponding effect on the individual marks. *If the contribution is non-equal then a brief statements needs to be supplied to justify this*. A separate percentage contribution can be specified for the report and evaluation. For example, if the contributions are specified as follows:

AGENT: Alice 30%, Bob 25%, Charlie 25%, Dominic 20%. NOTE: SHOULD ADD UP TO 100
REPORT: Alice 20%, Bob 30%, Charlie 20%, Dominic 30%. NOTE: SHOULD ADD UP TO 100

Then, if the agent performance obtained a mark of 80, and the report a mark of 60, then Alice will receive a mark of 30/25*80=96 for the agent. Marks for each component will be capped at 100 and will not go below the passmark of 50 unless the original mark was below 50. If applying the individual contributions results in marks outside the boundaries, the contributions will be adjusted accordingly. For example, if we apply the individual contributions to the report mark, Alice and Dominic would have received a mark of 48 which is below the passmark. The contributions for the report will therefore be adjusted to 21%,29%,21%,29% respectively.

If non-equal contribution are specified this needs to be supported by a short explanation, justifying any differences. The percentages along with the supporting statement should be placed in a section entitled 'Individual Contribution' and does not count towards the word count (although the entire document should be within the page limit). If this section is not included, then an equal contribution is assumed. Individual contributions need to be agreed by all team members. The module leader retains the option to deviate from the stated individual contributions (e.g. if no justification is included).

In deciding on individual contributions, note that contributions are not limited to programming or report writing. Reading the literature, attending the labs, running experiments, managing the team and organising meetings, also all count as contributions towards the coursework. Also, some teams may decide to develop several versions of the agent, and so even when the code is not used in the the final version, this can still be considered as a contribution. It is advisable that a record of effort is kept in case of disputes, e.g. in the form of commits and meeting minutes.

7 Plagiarism

Both the agent sourcecode and the report need to be the student's own work unless mentioned otherwise. For the report, also tables and figures etc should be your own (unless they are from an existing paper and source is cited appropriately). *Be careful with sharing your material (such as tables and figures) with other groups since enabling others to plagiarise is equally a breach of academic integrity.* Hence, if multiple reports are found with identical text/figures/tables/etc, ALL will be subject to plagiarism penalties and reported to the Academic Integrity Offer.

For the agent, you can use the code from the ExampleAgent 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** as part of your own agent, including agents who have previously participated in the international competition. 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. Also you can use code from existing agents as benchmarks in your evaluation to compare the performance of your agent. In case of doubt, feel free to ask! Any violations, deliberate or otherwise, will be reported to the Academic Integrity Officer with no exception.

More information about academic integrity can be found here.

8 Late Submissions

Late submissions will be penalised according to the standard rules. If agents are submitted late, they may not participate in the full tournament. Note that, if the submission is late due to an invalid submission, this is still subject to late penalties. It is your responsibility that you submit on time and that the submitted agent validates correctly.

9 The International Automated Negotiating Agents Competition

An international agent negotiation competition (ANAC) is being held on a yearly basis, and the coursework is based on this competition. The University of Southampton is actively involved in shaping this competition. Last year was the 10th installment, which is typically held at a high profile conference. In the last few years, it was held at the International Joint Conference on Artificial Intelligence (IJCAI), one of the very top AI conferences. Groups are encouraged to submit their agent to next year's competition. Note that the international competition uses GeniusWeb framework, which is version where agents communicate to a central server over web sockets. Some modifications will need to be made for the GENIUS agent to be used in GeniusWeb, but these are quite minor. Some previous competitions can be found here (the relevant league is called the Automated Negotiation League):

- ANAC 2020
- ANAC 2019
- ANAC 2018

10 A Brief Negotiation Tutorial

This section provides a more detailed description of the negotiation process, the challenges, and points to some of the existing literature on this topic.

10.1 Overview

A negotiation is defined by its negotiation *domain*, which tells you what issues are negotiable and what value-range each issue can take. Negotiation can involve a range of issues, from quite personal ones such as deciding on a holiday destination to business deals such as trading orange juice in international trade. For example, the party domain, which is one of the pre-defined domains, is about organizing a party together with some friends and negotiating what you want to spend the available money on. The domain specifies a *preference profile* for each agent, that captures each agent's individual 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* in the range of 0 to 1.

Given the domain, the next 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 an opponent.

A negotiation instance 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 (which may be zero but is not always the case). 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 parties. In the tournament that will be played, each agent will negotiate with many other agents and the scores of each session are recorded and averaged to obtain an overall score for the agent. A ranking will be compiled using these overall scores.

Category	Sample Feedback	Indicative Marks
Structure and Writing (max. 5 points)	The report does not follow a proper layout and structure and contains many spelling and/or grammar mistakes	0
	The report conforms with the requirements, and contains few/no spelling and grammar mistakes	5
Description and Understanding (max. 30 points)	The agent description is inadequate/missing	0
	The agent is explained but some parts are incomplete, not sufficiently formal (e.g. using mostly words with few equations or algorithms), and with very little motivation	10
	The agent is explained and mostly complete, containing some formal algorithms, and motivation for some of the choices, showing a good level of understanding	20
	There is an excellent agent description which is complete and clear, yet concisely written and using formal notation and algorithms. The strategy is well motivated and demonstrates an excellent understanding.	30
Challenge, sophistication, originality (max. 15 points)	The strategy is very basic and shows no originality	0
	The strategy is largely based on a single paper with no/ few new elements	5
	The strategy is based on existing literature and has been adapted to show some innovation	10
	The strategy has many novel and sophisticated elements, mixing ideas from several papers	15
Literature (max. 10 points)	The report contains no references to the literature	0
	There are some references to the literature but it is not clear how these references are used to inform the strategy of the agent	5
	There is clear evidence that the literature was read and understood, and used to motivate and support the development of the agent strategy	10
Evaluation and Analysis (max. 40 points)	There is no evaluation of the agent performance	0
	There is some evaluation of the agent performance, showing tables and graphs, but little/no analysis of the results	10
	There is an adequate evaluation and some discussion of the results but little/no critical evaluation or ways to improve the agents	20
	There is a very good evaluation and the discussion shows a good understanding of the performance of the agent, and some ways to improve the strategy	30
	The evaluation is extensive considering various metrics and benchmarks, and there is an excellent critical discussion of the performance of the agent, and ways to overcome the deficiencies and improving the agent	40

Table 2: Indicative Report Marking Scheme. Indicative marks are given out of a 100 points.

Offer	Utility Agent 1	Utility Agent 2
$o1 = \langle 0.5, 0.5 \rangle$	$U_1(o1) = 0.7 * 0.5 + 0.3 * 0.5 = 0.5$	$U_2(o1) = 0.3 * (1-0.5) + 0.7 * (1-0.5) = 0.5$
$o2 = \langle 1, 0 \rangle$	$U_1(o2) = 0.7 * 1 + 0.3 * 0 = 0.7$	$U_2(o2) = 0.3 * 0 + 0.7 * 1 = 0.7$
$o3 = \langle 0, 1 \rangle$	$U_1(o3) = 0.7 * 0 + 0.3 * 1 = 0.3$	$U_2(o3) = 0.3 * 1 + 0.7 * 0 = 0.3$
$o4 = \langle 0, 0 \rangle$	$U_1(o3) = 0.7 * 0 + 0.3 * 0 = 0$	$U_2(o3) = 0.3 * 1 + 0.7 * 1 = 1$
$o5 = \langle 1, 1 \rangle$	$U_1(o3) = 0.7 * 1 + 0.3 * 1 = 1$	$U_2(o3) = 0.3 * 0 + 0.7 * 0 = 0$

Table 3: Example of additive utilities for individual agents with 2 negotiation issues.

10.2 Understanding a Negotiation Outcome

A negotiation outcome can be assessed based on an agent's individual benefit, or it can be assessed at a joint level, e.g. to see whether a better outcome could have been achieved which would benefit *both* agents, and whether the outcome is *fair* and how this is defined.

In terms of the benefit of the outcome to the individual, this is described by a so-called *utility function*. As explained, negotiations involve multiple issues. In GENIUS and in this assignment, we assume utility functions *linearly additive*, i.e. they are a weighted function of the utility for individual issues, where the weight indicates the importance of an issue (e.g. price vs travel time when buying flight tickets). More formally, let o be an offer, where o_j is the proposed value for issue j (e.g. the price). Moreover, let $u_{i,j}(o_j)$ be the utility of agent i for that value. Then the overall utility of the offer, $U_i(o)$, is given by:

$$U_i(o) = \sum_{j=1}^{n} w_{i,j} \cdot u_{i,j}(o_j)$$

where n is the number of issues under negotiation, and $w_{i,j}$ is the weight of issue j for agent i. Crucially, both the utility function for each issue, and the weights can be different for different agents, which enables *mutually beneficial outcomes*.

For example, consider a setting with 2 agents and 2 issues, where $o_1, o_2 \in 0, 0.1, 0.2, \ldots, 1.0$ and we have that $u_{1,j} = o_j$ and $u_{2,j} = 1 - o_j$ for agents 1 and 2 respectively. That is, agent 1 prefers a higher value for each issue (e.g. the agent is a seller and the value is price), and agent 2 prefers a lower value (e.g. the agent is a buyer). Also, the agents have different weights: $w_1 = \langle 0.7, 0.3 \rangle$ and $w_2 = \langle 0.3, 0.7 \rangle$. That is, agent 1 finds the first issue more important and agent 2 the second issue. Consider 3 different offers: $o1 = \langle 0.5, 0.5 \rangle$ (both agents get something in the middle), $o2 = \langle 1, 0 \rangle$ (agent 1 is happy about the value of issue 1 but not of issue 2, and vice versa for agent 2) and $o3 = \langle 0, 1 \rangle$ (the opposite to the previous). Now consider the utility for each offer in Table 3. Note that there are many other possible offers/negotiation outcomes (in this case, with 2 issues and 11 values per issue, there are 11^2 possible outcomes). Take time to make sure you understand the table and how these values are calculated. As an exercise, try and derive the utility for some other offers not in the table.

All of these 3 offers are 'fair' in the sense that, for each offer, both agents get the same utility. However, some are clearly better than others. In particular, offer $o2$, where agent 1 gets the best value for his most preferred issue, and agent 2 gets the best value for her most preferred issue, has a utility of 0.7 for both agents, compared to the lower utility in other cases. This is called a *win-win* outcome, since both agents benefit.

It is important to note that, as seen in the example, these win-win outcomes exist *even* when the agents have diametrically opposing preferences for individual issues. Indeed, in such cases with diametrically opposed preferences, it is *only* possible to obtain win win situations by having multiple issues. If negotiation is only about a single issue, e.g. price, it is typically not possible to get a win-win outcome. Such negotiations are also referred to as *competitive*: the only way for one agent to gain, is for the other to lose. A multi-issue negotiation where win-win outcomes are available, as in the example, are called *integrative*. Often, in practice, additional issues are introduced with the specific aim to make the negotiations less competitive and more integrative. For example, a second-hand car dealer may offer a discount on the warranty, and so the warranty gets introduced as an additional issue alongside the price of the car.

We now analyse the properties of an outcome more formally in terms of the joint utility. A desirable property is for an outcome to be so-called *Pareto optimal* (also often referred to as *Pareto efficient*). An

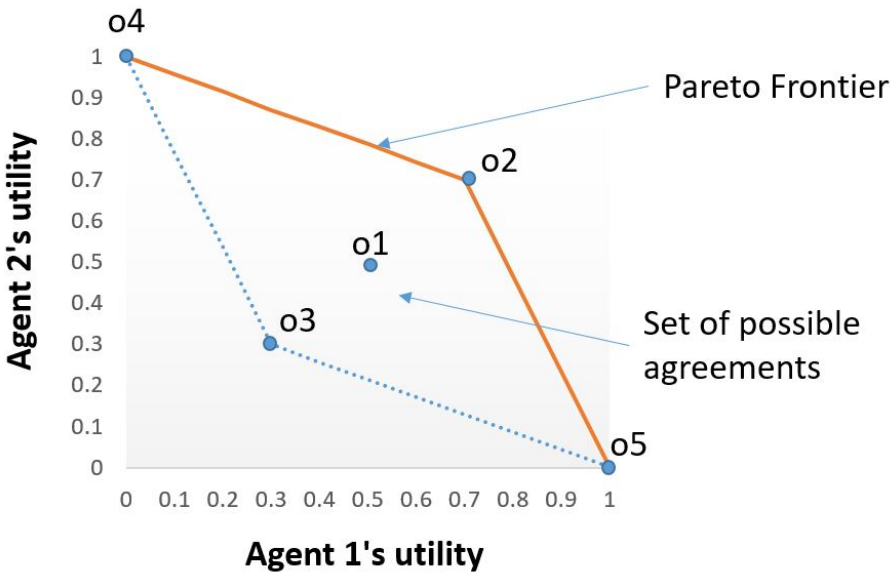

Figure 1: The outcome space and the Pareto frontier for the 2-issue negotiation domain example.

outcome is Pareto optimal when there does not exist another outcome where *both* agents can do better. Said differently, an outcome is Pareto optimal if, for any agent to be better off, the other agent has to be strictly worse off. In the example, $o1$ is not Pareto optimal since there exists another offer (e.g. $o2$) where both agents are better off. Here, $o2$ is Pareto optimal, but so are $o4$ and $o5$ (see Table 3). This is because there is no other outcome where both agents are better off. By connecting all the Pareto optimal solutions we obtain the so-called Pareto frontier. This is visualised in Figure 1, which shows the offers from the table in terms of both agent's utilities. Take time to understand these concepts before moving on.

Clearly, aiming to achieve an outcome on the Pareto frontier seems to be the sensible option (although this in itself is already challenging since there is a lot of uncertainty, as explained further below). However, a Pareto optimal solution is not unique since there can be many different outcomes on the frontier (as an exercise, try and determine how many outcomes in the example lie on the frontier other than the ones in the table). Some of these favour one agent over another. Also, in many of the negotiation domains there can be dozens of issues and hundreds of offers, and the Pareto frontier can look very complex. An example of a slightly more complex frontier is seen in Figure 2. So what agreement should we aim for? One possibility is to aim for a 'fair' outcome, i.e. one that is good for both parties, since such an outcome is likely to get accepted by both parties. There are many ways of defining this, but a well known concept is the Nash bargaining solution or simply Nash solution (note that this notion is NOT related to the Nash equilibrium other than the fact that they are both introduced by the late Nobel laureate John Nash). A Nash solution is an outcome that satisfies certain bargaining axioms and may be viewed as 'fair' outcome which is reasonable to accept for both parties. An outcome is said to be a Nash solution whenever it maximises the product of the utilities minus the disagreement utility (a.k.a. *reserve value*). More formally, let $U_i(d)$ denote the utility of a disagreement, then the aim is to find the outcome o which maximises (for 2 agents):

$$(U_1(o) - U_1(d)) \cdot (U_2(o) - U_2(d))$$

This solution has some interesting properties, e.g. it is invariant to how the utility functions are scaled (invariant to so-called affine transformations), it is Pareto optimal, and also it satisfied the symmetry property. The latter means that, if the two agents are symmetric (e.g. diametrically opposed and having the same reservation values), as in our example, then the agents should get the same utility. In the example with 2 issues, $o2$ is the Nash bargaining solution. Note that this solution is typically unique.

Other types of solution concepts include: maximising social welfare, which is the *sum* of utilities

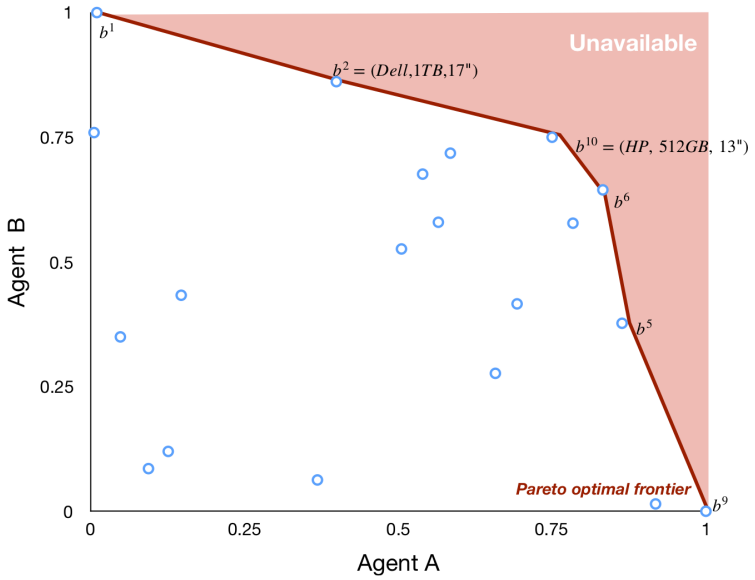

Figure 2: A more complex Pareto frontier example from the Laptop domain.

(instead of the product); the egalitarian solution, which maximises the utility of the agent with the lowest utility; the Kalai-Smorodinski bargaining solution, which chooses the point on the Pareto frontier which is closest to linear line which goes through the U_1^{max}, U_2^{max}, where U_i^{max} is the maximum utility agent i could achieve individually (see e.g. [8] for more details).

10.3 Deadlines and Discount Factors

Besides considering the utility of the outcomes of a negotiation, time pressure often plays an important role. Often, negotiations have firm deadlines. For example, if the date of the party has already been fixed, then the agents should come to an agreement on how to spend the money in organizing the party well in advance of that day. In the assignment there is a deadline for reaching an agreement. Both agents have exactly the same deadline and this is common knowledge (both agents 'know' when the deadline is). If they fail to reach an agreement each participant gets their disagreement utility.

Deadlines ensure that negotiations do not last forever, but often result in a deal being clinched at the last minute. Hence, in addition to a deadline, negotiations can have a different type of time pressure in the form of a *discount factor*. Discount factors model the fact that the desirability of the good being traded may decline with time or other types of urgency (e.g. lost opportunities). This happens when the good is perishable, for example fruits or other types of perishable food.

The formal modelling of discount factors in the context of automated negotiation and the GENIUS platform is as follows. Let $\delta \in [0,1]$ be the discount factor of a domain, and o be a certain outcome. Let t in $[0,1]$ be the current normalized time, as defined by the timeline. We compute the discounted utility $U_{i,t}^{dis}$ as follows:

$$U_{i,t}^{dis}(o) = U_i(o) \cdot \delta^t \tag{1}$$

At the deadline $t = 1$, the original utility is multiplied by the discount factor. If $\delta = 1$ or not specified at all, the utility is not affected by time, and such a domain is considered to be undiscounted, while if δ is very small there is high pressure on the parties to reach an agreement quickly. Note that in GENIUS, discount factors are part of the preference profiles and therefore can differ between parties.

10.4 Opponent Modelling

Although achieving a Pareto efficient and fair outcome may be desirable, the challenge is that a negotiation agent has no knowledge of the opponent's preferences, nor does it have knowledge of the negotiation strategy. Even the discount factor may be different and unknown. To address this, there is a wide range of literature on *opponent modelling*. These papers propose algorithms which try to infer the preferences and negotiation strategies of the opponent. In terms of additive utility preferences, this means inferring the weights as well as inferring the utility for individual values for each issue.

There are a wide range approaches, from simple heuristic to machine learning. A simple approach is to assume that the opponent will start with what is the best offer for them, and will slowly concede. This will give some indication of which issues the opponent finds more important. A related approach is by using frequency analysis, which has been successfully used by the Hardheaded agent [18]. The assumption here is that values for issues who appear more frequently in the offers received by the opponent are more likely to have a high utility for the opponent. Similarly, if certain values for certain issues persist compared to other issues where there are more frequent changes, this might mean that the weight for that particular issue is higher.

Examples of agents using more sophisticated approaches include *IAMHaggler* [25, 26, 24], which uses Gaussian process regression technique to predict the opponent's behavior and *OMAC Agent* [4, 3, 5, 2], which models the opponent using wavelet decomposition and cubic smoothing spline. In [14], a guessing heuristic is introduced to infer the opponent preferences.

10.5 Preference Uncertainty and Elicitation

In addition to having no initial knowledge of the opponent, another challenge introduced in the competition is uncertainty about the agent's own preferences and the possibility to elicit additional information at a cost. To understand why preference uncertainty occurs in practice, it is important to understand where the agent preferences come from in the first place. By definition, autonomous agents act on behalf of people or organisations. Hence, the preferences need to reflect what people and/or organisations want. The process of obtaining these preferences is called *preference elicitation* and is generally costly, both in terms of time, and hiring experts to do so. Preferences can be very complex (a good classic book that describes this nicely is by Raiffa [21]). Typically, people find it very difficult to give exact utility values for certain outcomes, or understanding what the weights might mean. It is typically much easier for people to compare offers in a pairwise manner and to say which of the two they would prefer [9]. By doing many pairwise comparisons one can get closer to understanding the true utility function, but in complex negotiations there are simply too many pairs to compare. As a result, often what you end up with is a partial ordering of preferences.

In the competition, preference uncertainty means that the agent only has ordinal information about different offers, and only of a subset of the full offer space. Therefore, it may know that it prefers A over B, but not by how much (i.e. it knows the *order* but not the *utility*) and it may not know anything about C. There are two main ways in which to deal with such situation. One way is to derive an estimated (additive) utility function from the information given, and then use this utility function during the negotiation. Some built in functions are provided to help with this (see the GENIUS manual for more detail) but you are encouraged to come up with your own approach or use an existing approaches from the literature (e.g. that are used for opponent modelling). For example, a more sophisticated approach to derive the preferences has been developed in [23] using linear programming. Another way is to negotiate with the incomplete set of ordinal preferences directly, without the 'translation' step.

In addition, there is the option to elicit or 'buy' additional information about an offer. That is, the agent can ask to add any unranked offer as part of the ordinal preference ranking, but this comes at a fixed cost. The cost is set as part of the domain. This simulates the idea of having to ask the user, and that preference elicitation is costly in terms of effort. The elicitation aspect raises many interesting research questions to decide how many offers to elicit information about, and which ones.

10.6 Negotiation Strategies and Techniques

As already mentioned, a negotiation strategy needs to determine what to bid (the offering strategy), and whether to accept if an offer is received (the acceptance strategy). These are typically related. Assuming the agent knows or has estimated its own utility, a common approach is to split the strategy into two separate components: (1) determining the utility threshold level at which to make and accept offers, often referred to as the *concession strategy*, and (2) finding the 'best' offer at or above the chosen utility threshold.

Regarding the concession strategy, this can be a time-based schedule or adaptive which responds to the opponent. Examples of time-based concession strategies are Boulware (concede slow to begin with, and then concede faster as you get closer to the deadline), Conceder (concede fast from the beginning), Linear (concede linearly), Hardheaded (don't concede until the very end). See e.g. [6]. A well known adaptive strategy is tit-for-tat, which concedes a similar amount to what the opponent is perceived to concede ([7, 1]). Many approaches use one of the time-based strategies but then change the parameters of this strategy in response to what the apponent is doing, e.g. changing the target utility (which is the utility threshold at the deadline). A notable ANAC winner agent is *Agent K* [16, 17], which calculates its target utility based on the average and variance of previous bids and employs a sophisticated acceptance strategy. Furthermore, the *CUHK Agent* [10, 11] adaptively adjusts its acceptance threshold based on domain and opponent analysis.

Another crucial factor is finding the best offer given the current utility level, i.e. deciding on a specific value for each issue. An early approach is by Faratin et al. [7] which chooses an offer that is closest in terms of similarity to the opponent's previous offer, but at the same time is above the utility threshold of the proposing agent. A similar approach is taken by [22], who propose the Orthogonal strategy which minimises the Euclidean distance between the most recent offer made by the opponent, and offers at the agent's utility threshold level. They show that, for certain types of domains, if concession is sufficiently slow, and both agents use this approach, the agreed offer is guaranteed to be Pareto optimal. Another common approach is to first estimate the opponent model (as discussed before), and then choosing amongst all the offers above the utility thresholds the one which has the highest utility for the opponent. This maximises the chances that the offer is above the opponent acceptance threshold, and therefore will be accepted.

There is a vast range of other negotiation agents in the literature that have been developed over the past decades, e.g. Zeng and Sycara [27], who introduce a generic agent called *Bazaar*; Karp et al. [15], who take a game-theoretic view and propose a negotiation strategy based on game trees; Jonker et al. [14], who propose a a concession oriented strategy called *ABMP*; and Lin et al. [19], who propose an agent negotiator called *QOAgent*; *The Fawkes*, which combines the best bidding, learning, and accepting strategy components; *Meta-Agent* [12, 13], which, for any given negotiation domain, dynamically selects the most successful ANAC agent to produce an offer, and many more. ANAC benchmark strategies that consider most aspects of the competition, including preference uncertainty, are discussed in [20].

In designing your own agent, you should search for and read relevant papers, including those from past ANAC competitions. The papers in this document can be a starting point, but you are encouraged to search for your own papers e.g. using Google Scholar and keywords such as 'ANAC' and 'negotiation' and look for recent papers for the last 3-4 years.

References

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

[2] Siqi Chen, Haitham Bou Ammar, Karl Tuyls, and Gerhard 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*, AAMAS '13, pages 707–714, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[3] Siqi Chen and Gerhard Weiss. An efficient and adaptive approach to negotiation in complex environments. In Luc De Raedt, Christian Bessiere, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J.F. Lucas, editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 228–233. IOS Press, 2012.

[4] Siqi Chen and Gerhard Weiss. A novel strategy for efficient negotiation in complex environments. In Ingo J. Timm and Christian Guttmann, editors, *Multiagent System Technologies*, volume 7598 of *Lecture Notes in Computer Science*, pages 68–82. Springer Berlin Heidelberg, 2012.

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

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

[7] Peyman Faratin, Carles Sierra, and Nicholas R. Jennings. Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence*, 142(2):205 – 237, 2002.

[8] Enrico H Gerding, DDB van Bragt, and JA La Poutre. Scientific approaches and techniques for negotiation. *Report. SEN: Software engineering/Centrum voor wiskunde en informatica*, 2000.

[9] Shengbo Guo and Scott Sanner. Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 289–296, 2010.

[10] Jianye Hao and Ho-fung Leung. ABiNeS: An adaptive bilateral negotiating strategy over multiple items. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, volume 2 of *WI-IAT '12*, pages 95–102, Washington, DC, USA, Dec 2012. IEEE Computer Society.

[11] Jianye Hao and Ho-fung Leung. CUHK agent: an adaptive negotiation strategy for bilateral negotiations over multiple items. In Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, Takayuki Ito, Minjie Zhang, Quan Bai, and Katsuhide Fujita, editors, *Novel Insights in Agent-based Complex Automated Negotiation*, volume 535 of *Studies in Computational Intelligence*, pages 171–179. Springer, Japan, 2014.

[12] L. Ilany and Y. (K.) Gal. The simple-meta agent. In I. Marsa-Maestre, M.A. Lopez-Carmona, T. Ito, M. Zhang, Q. Bai, and K. Fujita, editors, *Novel Insights in Agent-based Complex Automated Negotiation*, volume 535 of *Studies in Computational Intelligence*, pages 197–200. Springer, Japan, 2014.

[13] Litan Ilany and Yakov Gal. Algorithm selection in bilateral negotiation. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2013)*, 2013.

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

[15] Alan H. Karp, Ren Wu, Kay-yut Chen, and Alex Zhang. A game tree strategy for automated negotiation. In *Proceedings of the 5th ACM conference on Electronic commerce*, EC '04, pages 228–229, New York, NY, USA, 2004. ACM.

[16] Shogo Kawaguchi, Katsuhide Fujita, and Takayuki Ito. Compromising strategy based on estimated maximum utility for automated negotiating agents. In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-based Complex Automated Negotiations, Seriesof Studies in Computational Intelligence*, pages 137–144, Berlin, Heidelberg, 2012. Springer-Verlag. URL = http://link.springer.com/content/pdf/10.1007

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

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

[19] Raz Lin, Sarit Kraus, Jonathan Wilkenfeld, and James Barry. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *Artificial Intelligence*, 172(6-7):823 – 851, 2008.

[20] Kotone Ninagawa, Yasser Mohammad, and Amy Greenwald. Baseline strategies for the anac automated negotiation league. `https://preflib.github.io/gaiw2021/papers/GAIW_2021_paper_25.pdf`.

[21] Howard Raiffa. *The Art and Science of Negotiation*. Belknap Press, reprint edition, 2005.

[22] DJA Somefun, Enrico H Gerding, and Johannes A La Poutré. Efficient methods for automated multi-issue negotiation: Negotiating over a two-part tariff. *International Journal of Intelligent Systems*, 21(1):99–119, 2006.

[23] Dimitrios Tsimpoukis, Tim Baarslag, Michael Kaisers, and Nikolaos G Paterakis. Automated negotiations under user preference uncertainty: A linear programming approach. In *International Conference on Agreement Technologies*, pages 115–129. Springer, 2018.

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

[25] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. Iamhaggler: A negotiation agent for complex environments. In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*, pages 151–158, Berlin, Heidelberg, 2012. Springer-Verlag. URL = http://eprints.soton.ac.uk/271662/1/acan2010.pdf.

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

[27] Dajun Zeng and Katia P. Sycara. Bayesian learning in negotiation. *International Journal of Human-Computer Studies*, 48(1):125 – 141, 1998.