Collaborative AI (CSE3210): Negotiation Practical Assignment

Pradeep K. Murukannaiah and Catholijn M. Jonker

Interactive Intelligence, Delft University of Technology

March 13, 2025

Table 1: Deliverables and Deadlines

Deliverable	**Deadline**	**Notes**	**Weight**
Project Plan	March 21, 2025	Describes how you will tackle the problem and how each member of the team will contribute to the project. Ungraded, but required submission	–
Implementation and Group Report	April 4, 2025	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 based on multiple criteria and negotiation scenarios, and will show evidence that related literature has been read, understood and applied.	100%

Contents

1 Introduction **3**

2 The Assignment **3**
- 2.1 Negotiation Setup . . . 3
- 2.2 Assignment in Steps . . . 3
- 2.3 Analyzing the Performance of Your Agent . . . 4
- 2.4 Ethics and Plagiarism . . . 4
- 2.5 Open Science . . . 4

3 Getting Started **4**
- 3.1 Python Dependencies . . . 5
- 3.2 Detailed Documentation . . . 5

4 Submission Details **5**
- 4.1 Deliverables . . . 5
- 4.2 Report . . . 5
- 4.3 Requirements . . . 6
- 4.4 Submission . . . 6
  - 4.4.1 Project Plan . . . 6
  - 4.4.2 Implementation and Group Report . . . 7
- 4.5 Individual Contributions . . . 7

5 Evaluation **7**

6 Automated Negotiating Agents Competition (ANAC) **8**

7 Future Perspectives **8**

1 Introduction

Negotiation is a form of interaction in which a group of agents with conflicting interests strive to reach an agreement over an outcome [1, Chapter 4]. Negotiation is fundamental to collaborative AI because often agents cannot fully accomplish their objectives on their own and need to exchange resources with each other.

In this practical assignment, you will create your own negotiating agent in the GeniusWeb framework. This document provides the outline of the assignment, pointers to the literature that you can use as background information, links to the negotiation platform that we will use for this assignment, and usual information on deadlines, where to get help, how to submit your files, and so on.

For theoretical concepts, please refer to the handbook on negotiating agents [2].

Learning Objectives After completing this practical assignment, you should have learned to:

- Apply theories and techniques from multi-agent negotiation, including utility theory, preference modeling, communication protocols, strategy modeling, and opponent modeling.
- Create a negotiating agent in the GeniusWeb framework.
- Analyze the performance of your agent quantitatively and qualitatively.
- Describe the design and implementation of your agent concisely in a group report.

2 The Assignment

2.1 Negotiation Setup

The negotiation will run using the GeniusWeb framework and the setup is based on the rules from the International Automated Negotiating Agents Competition (ANAC). It will be a **bilateral** negotiation between two 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.

An agent will not have any knowledge of its opponent's preferences. Negotiation is time based and each negotiation lasts **10 seconds**. Further, the agents have reservation values (the utility received by the agents when no agreement is achieved). Reservation values of agents can be different and an agent does not know the reservation value of its opponent. In GeniusWeb, the reservation value is implemented as a *reservation bid*, i.e., any bid that is equal or better also has a higher utility than the reservation bid. Finally, there will not be any discounting (i.e., the discounting factors are set to 1).

2.2 Assignment in Steps

1. Develop an agent to negotiate in the setup above.
2. Experiment and evaluate your agent by running several negotiations (or tournaments).
3. Analyze the performance of your agent. See Section 2.3 for details.
4. Write a report. See Section 4.2 for details.

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

2.3 Analyzing the Performance of Your Agent

How well does your agent perform and how generic is it? That is, does your agent perform well against other agents? And, does it work as well on multiple domains? To verify this, have your agent negotiate against itself, and against as many other agents you can find and in as many domains you can find. For this, you can look in the GeniusWeb repository for agents that can run with SAOP protocol. You can also experiment with the agents of other groups in your class if the groups are willing to collaborate (of course, **without plagiarizing** each other's code). If you collaborate with other groups, explicitly name and thank these groups in your report. Report on the outcomes reached. Try to explain why the outcome is as it is.

We want to emphasize that it is important to test your agent in order to improve the negotiation strength of your agent in *different* settings. **Important**: Do not assume that your goal is to outperform a 'stupid' agent. Test your agent against as many agents as you can and older versions of your own agent.

Think carefully on the criteria that you set to judge the performance of your agent. In normal negotiations, a good outcome is determined by criteria such as the efficiency of the outcome. That is, how close is the outcome to the Pareto frontier? How close is it to the Nash Product? Does it optimize Social Welfare?

2.4 Ethics and Plagiarism

We expect from our students that they adhere to the ethical standards of TU Delft, which includes that the work you submit is that of your group only, and you only collaborate with other groups or people in as far as is explicitly allowed by the assignment. In particular, the following behaviors are strictly prohibited:

- Designing an agent in such a way that it benefits some specific other agent.
- Hacking or exploiting bugs in the software.
- Communicating with the agent during time it is being evaluated by the teaching team.

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

2.5 Open Science

Please realize that your agent and its description may be made open source with your permission. Regardless of that, we expect our students to come up with non-offensive material according to TU Delft standards that is suitable for sharing with the scientific community.

3 Getting Started

You should develop your agent in Python 3.9. Note: Some of the other documents may mention that you can use Python 3.8 or 3.9, but we require you to use Python 3.9 to avoid potential conflicts. We do not support Python 3.10 or 3.11, yet. There is also a Java version in which you can develop agents, but for this course we require all submissions to be developed in Python 3.9.

The following GitHub repository: https://github.com/brenting/ANL-2023-example-agent contains a simple runner setup for Python 3.9 and contains everything that you need to develop and test a negotiation agent. A template agent is already provided that you can use as a starting point of your agent.

The repository above also has a set of agents (standard agents, agents from previous year of the course, and agents from the competition) and a set of preferences profiles available for evaluating your agent. The agent you develop through this platform will be compatible with GeniusWeb [3]. However, the deliverables should comply with the structure described in Section 4.1.

3.1 Python Dependencies

If you develop your agent in Python and wish to use additional packages, please use the versions indicated in this file: https://github.com/brenting/ANL-2023-example-agent/blob/main/requirements_allowed.txt. Please use only the minimum necessary number of packages, you do not need to install all the listed packages. This is important since we do not want to run into the issue of conflicting package version when running your agent on our set up. This is also important for you to run a tournament with other agents.

If there are additional dependencies that you **must** use, please email `collabai-cs-ewi@tudelft.nl` with a subject line "Project 3: Request to Add a Library." We will then decide if we can add the library and the version number to the `requirements.txt` file, after which your agent can use it. This is necessary because we want to maintain one `requirements.txt` file that works for all projects. Please do not ask for this unless it is absolutely necessary. Also, please note that **we may not be able to honor all requests**.

3.2 Detailed Documentation

The details on how to write an agent in Python are available in https://tracinsy.ewi.tudelft.nl/pubtrac/GeniusWebPython/wiki/WikiStart. Note: an agent is called a party in GeniusWeb. The Python repository that is supplied for this course in based on the stand-alone running version of GeniusWeb (https://tracinsy.ewi.tudelft.nl/pubtrac/GeniusWebPython/wiki/WikiStart#Stand-alonerunning). Therefore, you can **ignore** all documentation on setting up the **webservers**. Note that the documentation on the classes is compatible with the Java version of GeniusWeb and is therefore not fully Pythonic.

4 Submission Details

4.1 Deliverables

- You should deliver a zip file containing your report, and the source code of your project. Your group number N should be clearly stated in every document and every file you submit. The assigned group number is the number of your group on Brightspace. Your end report should list the names of the group members.
- You should create a new package under `ANL-2023-example-agent/agents/groupN_agent` in the project provided to you, and include all your source and resource files under this package (you can create additional sub-packages under this package, if you want to).
- Your submission should include all source files (and only source files, not compiled files) and any resource files (e.g., text files with settings or data) that you created in the package you created.

4.2 Report

The report should be **five pages or less** (excluding references and description of individual contributions) in the IJCAI submission format. Five pages is an upper limit, not a goal. We encourage concise reports. Thus, **five pages is a strict upper limit**. You may be able to produce a lot of results, but often it is easy to combine these results. Further, it is also important to select the main messages that you want to convey.

The report 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, too). This means that the main methods used by your agent should be explained in the report. In particular, you should include:

- A high-level description of the agent and its structure, including the main Python methods (mention these explicitly!) used in the negotiating agent 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.

- A section documenting the strong and weak points of your agent, the tests you performed to analyze (and improve) the negotiation strength of your agent. You must include scores of various tests over multiple sessions that you performed. Describe how you set up the testing situation and how you used the results to modify your agent, and motivate why you choose these testing methods, metrics, or graphs. See Section 2.3 for more information.
- Cover all the points of Section 2.2.

You may include **appendices** in addition to the main paper, but it is not required. There is no page limit or formatting requirement for the appendices. **Important:** Appendices are not a way to bypass the five-page limit of the main report. Appendices will not be graded the same way as the main report. Appendices should only include supplemental material and they must be referred to from the main report. For example, a figure in the main paper may summarize the main results from an experiment, but the detailed results, e.g., in a table format, can be included in an appendix.

4.3 Requirements

- The agent should implement a negotiation strategy to generate offers, and an acceptance strategy to decide whether to accept an offer or not.
- The agent must aim at the best negotiation outcome possible, while taking into account that it may need to concede to the other agents.
- Show how your agent takes the preferences of the other agents into account.
- The agent meets reasonable time and memory constraints. Any agent that uses more than a minute in order to initiate the negotiation or to reply to an opponent's offer will be disqualified.
- Make sure your agent at least works with linear additive profiles.
- The source code should contain explanatory comments that will allow people not involved in programming the code to understand it. A clear explanation of a method must be provided at the beginning of important methods. Parts of the code that are not self-explaining should receive additional documentation. Please incorporate enough information in comments in the code to ensure this. Finally, make sure that you have removed all debug printouts from your code and that your agent does not unnecessarily burden the servers.
- Your code should be tested to ensure that it works on multiple operating systems, and requires nothing other than the files contained in your group's submission.
- The submission should include all the elements listed in Section 4.1.

4.4 Submission

4.4.1 Project Plan

By **March 21, 2025**, you should submit a project plan as a PDF file. The plan should describe how you will tackle the problem and how each member of the team will contribute to the project. The project plan will not be graded, but submission is mandatory. We require you to submit a project plan for two reasons.

(1) We want you to start working on the project early. Since this is an open-ended project, starting late (e.g., in the week of the deadline) will very likely lead to high stress and poor results.

(2) Since this is a group project, it is important to set clear expectations from each group member. Writing a project plan is an opportunity for you to set such expectations.

4.4.2 Implementation and Group Report

By **April 4, 2025**, each group should deliver two files: (1) your report as a PDF file, and (2) a zip file of your project, following the structure and naming convention described in Section 4.1. Do not submit incomplete assignment solutions; only a complete assignment solution containing all deliverables will be accepted. In case of any problems, please contact us well in advance of the deadline.

4.5 Individual Contributions

At the end of the report, you must describe how each member of the team contributed to the project. This part of the report is not counted in the five page limit. However, we would like concise description of the contributions. We will seek additional details if necessary.

5 Evaluation

Evaluation is based on your implementation and report. Table 2 shows an indicative grading scheme.

Table 2: Grading scheme showing indicative grades for different aspects of your report. If your report is completely missing information on a category, you receive zero points for that category

Category (weight)	**Sample feedback**	**Grade**
Structure and Writing (5%)	The report does not follow a proper layout and structure and contains many spelling and/or grammar mistakes	1–6
	The report conforms with the requirements, and contains few/no spelling and grammar mistakes	6–10
Description and Understanding (20%)	The agent description is inadequate/missing	1–6
	The agent is explained but some parts are incomplete; and, not all design choices are motivated	6–8
	The agent is explained and mostly complete; design choices are well motivated; shows a good level of understanding	8–10
Literature (10%)	There are some references to the literature but it is not clear how these references are used to inform the strategy of the agent	1–6
	There is clear evidence that the literature was read and understood, and used to motivate and support the development of the agent strategy	6–10
Sophistication and Originality (25%)	The strategy is very basic and shows no originality	1–6
	The strategy is based on existing literature and has been adapted to show some innovation	6–8
	The strategy has many novel and sophisticated elements, e.g., mixing ideas from several papers	8–10
Evaluation and Analysis (40%)	There is some evaluation of the agent performance, showing tables and graphs, but little/no analysis of the results	1–6
	There is an adequate evaluation and some discussion of the results but little/no critical evaluation or ways to improve the agents	6–8
	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	8–10

6 Automated Negotiating Agents Competition (ANAC)

Did you like thinking about efficient negotiating strategies, or implementing a successful negotiating agent? You can enter in the international Automated Negotiating Agents Competition (ANAC) in 2025 which will be colocated with the International Joint Conference on Artificial Intelligence (IJCAI) to be held in Montreal, 16th to 22nd August, 2025. The task and the negotiation platform in the competition (see [4] for example) may be different from those in this assignment. However, the ideas you use in this assignment can be easily developed further for ANAC. If you are interested in this, we would like to help you. You could join a writing team on the scientific aspects of this assignment. Please ask the teachers about this. Student teams that submitted their agent after this assignment have received prizes in the competition (and won $1000 in 2011 and 2013) and were awarded student scholarships to attend international conferences to present their agent.

7 Future Perspectives

Negotiation has become a major and interesting research area within Artificial Intelligence. Negotiation is important in many domains ranging from business applications such as Internet market places to incident and crisis management. Negotiation is one of the main tools an autonomous agent has to agree about a course of action or to resolve conflicts with other agents.

The agents in the negotiation environment have limited capabilities in order to achieve a negotiated deal. What extensions would be required to use your agent in real-life negotiations to support (or even take over) negotiations performed by humans? Can you think of additional capabilities (e.g., other actions or forms of communication) for your agent that would make it more practical and help achieve better deals?

There are many angles that you can take:

- Design of negotiation strategies and better opponent models (using machine learning techniques);
- Design of a negotiation support system, either advising humans in a particular domain, or even taking over negotiation all together;
- Design of richer negotiation protocols, such as protocols allowing for partial bids, use a form of communication between the agents, auctions;
- Design and add explanation facilities for negotiation support systems, negotiating agents, and/or GeniusWeb;
- Design and implement agents that can handle non-linear utility functions, add dependencies, and enrich GeniusWeb to support people using that functionality;
- Design self-learning agents that improve their negotiation strategies with experience;
- 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?

This can all be combined with a Literature Survey course, in order to kick-start your research project (e.g., an MSc thesis a year later). If you are interested, don't hesitate to ask us.

Acknowledgement

Thanks to Enrico Gerding for sharing a negotiation assignment, and to Wouter Pasman and the Teaching Assistants of Collaborative AI and AI Techniques courses for providing comments on this document.

References

[1] Gerhard Weiss. *Multiagent Systems*. The MIT Press, Cambridge, MA, 2013.

[2] Catholijn M. Jonker, Reyhan Aydoğan, and Tim Baarslag. Negotiating agents. Brightspace (Resources and Tools), 2021.

[3] GeniusWeb Team. GeniusWeb: An open architecture for negotiation via the internet. https://tracinsy.ewi.tudelft.nl/pubtrac/GeniusWeb, 2023.

[4] ANAC 2024. Automated negotiating agents competition (ANAC): Automated negotiation league. https://scml.cs.brown.edu/anl, 2024.