Automated Negotiation League (ANL) 2023

Join us on Discord!

1 Challenge

Design a negotiation agent for bilateral negotiation that can learn from every previous opponent while the tournament progresses. The highest-scoring agents based on individual utility and social welfare win the league.

While trying to reach a mutual agreement with any sophisticated agent, we have an initial thought thanks to our experiences from previous negotiation experiences with that agent or similar agents. Then, We take into account opponent preferences about current issues and opponent's behavior in new interactions. This year at the ANL challenge, we're looking forward to the agents you've developed precisely that behavior. The developed agent should be capable of making an opponent model in the current negotiation while learning from previous negotiations. Thus, we aim to provide as much flexibility as possible while. At every negotiation, agents obtain a directory path to save and load anything they want. This state is stored as a file on disk, and the path is passed to the agent every time a negotiation is initiated.

The competition takes place during AAMAS 2023, 29 May-2 June 2023, in London, UK. There will be $600 in prize money for ANL participants. This prize money distribution is provided in Table 1. We will also support a certain number of finalists participating in AAMAS for presenting their agents.

Ranking	Utility	Social Welfare
Winner	$200	$200
Runner-up	$100	$100

Table 1: Prizes

2 Preliminaries

A negotiation setup consists of a set of rules (protocol) to abide by, an opponent to negotiate against, and a problem (scenario) to negotiate over. We describe all three components in this section.

Negotiation Protocol. The Stacked Alternating Offers Protocol [1] (SOAP) is used as negotiation rules Here, the starting agent has to make an opening offer after which the agents take turns while performing one of 3 actions:

1. Make a counteroffer (thus rejecting and overriding the previous offer)
2. Accept the offer of the opponent agent
3. Walk away (e.g., ending the negotiation without any agreement)

This process is repeated in a turn-taking fashion until reaching an agreement or deadline. If no agreement has been reached at the deadline (e.g., 60 seconds), the negotiation fails. Please remember to test and report the performance of your agents at different deadlines. If there is a deal, a score is calculated for both

agents based on the utility of the deal at the end of a session. Otherwise, the score equals the reservation value of your preference profile. Each party will obtain a score based on its utility function. Since no deal has been reached, this is a missed opportunity for both parties.

Ideally, such an outcome has specific properties. One of these properties is that the outcome should be Pareto optimal. An outcome is Pareto optimal when there is no deal where one party can do better, and one can do worse or the same (i.e., they both score higher or equal, and therefore both would prefer this new deal over the old one). Another property is related to the Nash solution concept. A Nash solution is an outcome that satisfies certain bargaining axioms and may be viewed as a "fair outcome" which is reasonable to accept for both parties. An outcome is said to be a Nash solution whenever the product of the utility (in the range $[0,1]$) of the outcome for the first party and that for the second party is maximal (cf. [3]). A fair outcome is essential in the negotiation because it provides a reference point for what your opponent might be willing to accept. Typically, a negotiation is started when reaching an agreement is better than not reaching one. However, it would help if you collaborated with your opponent to get a deal. In a negotiation between self-interested parties, however, each party is determined to get the best possible outcome for itself, not regarding the utility of its opponent.

Scenario. The scenario consists of a discrete outcome space (or domain) $\omega \in \Omega$ and a preference profile per agent. This preference profile is used as a utility function $u(\omega)$ to map the problem space to a utility value for that agent (Equation 1). Here, 0 and 1 are the worst and best utility, respectively, obtained by the agent. The negotiations are performed under incomplete information, meaning that the utility function of the opponent is unknown.

$$u : \Omega \rightarrow [0,1] \tag{1}$$

The outcome space of negotiation can be plotted on a graph that indicates the utility of both parties on the x and y axes, respectively. An example is provided in Figure 1. This graph helps us understand and analyze bilateral negotiations.

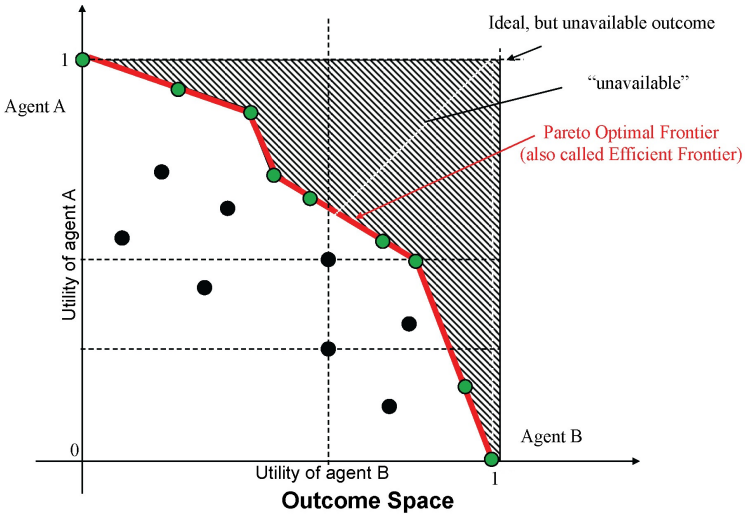

Figure 1: Negotiation Outcome Space

2.1 Platform

Entrants to the competition have to develop and submit an autonomous negotiating agent in Python that runs on GeniusWebPython [2]. GeniusWeb is a negotiation platform where you can develop general negotiating agents and create negotiation domains and preference profiles. The platform allows you to simulate

negotiation sessions and run tournaments. At every negotiation, agents obtain a file path to save anything they want. An agent gets a chance to process all that has been saved for that agent on multiple occasions during the tournament.

Extensive references are available in section 5, where we provide links to additional information. A basic example agent that can handle the current challenge can be found on GitHub. We aim to provide a quick-start and FAQ on this GitHub page, which we will improve incrementally based on feedback received via e.g. Discord).

2.2 Evaluation

If the agents reach an agreement, the outcome's utility is the agent's score. This utility is usually different for both agents. A utility of 0 is obtained if no agreement is reached. There is two different categories:

Individual Utility At the end of the tournament, the average utility of every agent is calculated. The agent with the highest average utility wins the utility category.

Social Welfare Social welfare is the sum of the utilities of the agents involved in a negotiation session. Both agents obtain the same score for social welfare. The agent with the highest average social welfare overall negotiation sessions wins the social welfare category. The goal of this measure is to emphasize the cooperative challenge that negotiation agents face.

2.3 Rules of encounter

- Agents need to follow the SAOP protocol.
- Opponents are encountered multiple times.
- Agents will not encounter a negotiation with the same preference profile against the same opponent.
- In total, 50 randomly generated scenarios will be played against each opponent. These scenarios will all be fully discrete (so no continuous issues).
- Data can only be saved to a path provided to the agent (see code).
- Every agent has a limit of 10GB of storage in the provided directory.
- Violating the spirit of fair play will result in disqualification. The ANAC board will be the judge in these matters.
- The competition rules allow multiple entries from a single institution but require each agent to be developed independently.
- No participant can be a co-author of more than three agents.
- The source code of agents must be submitted. This code will be included in the GeniusWebPython platform after the competition is finished for future use.

3 Learning & data storage

You are allowed to store data in a directory that is provided to your agent. You can use this storage for learning purposes.

NOTE: Due to parallelization, your agent will be run against every unique opponent concurrently. Take this into account when saving and loading data.

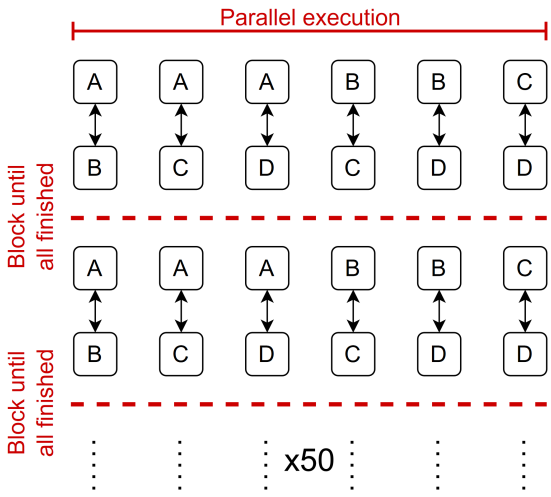

Figure 2: Agents are $\mathscr{A} = \{A, B, C, D\}$ in an example tournament where they negotiate randomly and sessions never repeated.

Figure 2 illustrates the execution procedure of the tournament. As already described in the frame above, all agent combinations will negotiate in parallel on a randomly generated negotiation problem. The next round is blocked until all the agents have finished their negotiation session. This allows all agents to know all opposing agents at the start of every following session. A total of 50 negotiation sessions are run.

4 Submission & questions

Participants submit their agent source code and academic report (optionally) in a zipped folder. For the code, include the directory with your agent like the directory of the template_agent in the GitHub repository) in the zipped folder. Ensure your agent works within the supplied environment by running it through "run.py". Please submit your application through the following link:

Submission Form

4.1 Academic report

Each participant should prepare a 2-4 page report describing the design of their agent according to academic standards. The best teams can give a brief presentation describing their agent depending on the available slots at AAMAS 2023 .

Furthermore, the selected agent papers are planned to be published in the proceedings of coming AAMAS. The organizers of this league will evaluate the report. For eligibility, the design should contribute to the negotiation community. The report should address the following aspects:

- Bidding Strategy: How the agent generates bids each turn.
- Acceptance Strategy: How the agent accepts or rejects a bid.
- Opponent Modelling: How the agent models the opponent (e.g., the opponent's strategy, preferences, etc.).
- Learning method: What does the agent learn and why? Show the influence of the learning behavior of the agent.

5 Fact sheet

Important dates

Final Deadline for submissions: 1st May, 2023
Announcement of finalists: 15th May, 2023 (tentative)
ANAC 2023: 29th May-2nd June 2023, in London; jointly held with AAMAS 2023

Important links

Making an agent for ANAC 2023
Discord Server
Competition Website
Submission Form
Getting started with GeniusWeb
Technical information GeniusWebPython

Contact

Mehmet Onur Keskin	Main contact for questions about the challenge
Wouter Pasman	Developer of GeniusWebPython

Advisers (alphabetic)

Name	Title	Affiliation
Reyhan Aydoğan	Assistant Professor	Ozyegin University
Tim Baarslag	Scientific Staff Member	Centrum Wiskunde & Informatica
Katsuhide Fujita	Associate Professor	Tokyo University of Agriculture and Technology
Catholijn Jonker	Professor	Delft University of Technology

References

[1] Reyhan Aydoğan, David Festen, Koen V. Hindriks, and Catholijn M. Jonker. Alternating offers protocols for multilateral negotiation. In *Studies in Computational Intelligence*, volume 674, pages 153–167. Springer, 2017.

[2] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, 30(1):48–70, 2014.

[3] Roberto Serrano. Bargaining, November 2005.