Automated Negotiation League (ANL) 2022

Join us on Discord!

1 Challenge

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

In previous years of ANL, either no learning across negotiation sessions was allowed, or learning options were limited. e.g. Learning over identically repeated negotiation sessions. This year we aim to provide as much flexibility as possible while. At every negotiation, agents obtain a directory path where they can 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 IJCAI 2022, July 2022, in Vienna, Austria. There will be $600 in prize money for ANL participants. This prize money distribution is provided in Table 1.

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. Accept the offer of the other agent
2. Make a counteroffer

	Utility	Social Welfare
First place	$200	$200
Second place	$100	$100

Table 1: Prize money (For evaluation criteria see section 2.2)

3. Walk away

To prevent agents from negotiating indefinitely, a deadline of 60 seconds is set.

Opponents. The opponents will be agents that are submitted by other competitors in the ANL.

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.

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

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 in which you can develop general negotiating agents as well as create negotiation domains and preference profiles. The platform allows you to simulate negotiation sessions and run tournaments. 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, then 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. This can occur for one of three reasons:

1. One of the agents walks away
2. The deadline is reached
3. One of the agents crashes

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 The social welfare is the sum of the utilities of the agents that were involved in a negotiation session. Both agents obtain the same score for social welfare. The agent with the highest average social welfare over all negotiation sessions wins the social welfare category.

The goal of this measure is to emphasise the cooperative challenge that negotiation agents face.

2.3 Rules of encounter

- Agents need to follow the SAOP protocol.
- Opponents are encountered multiple times.
- A single scenario is negotiated over **once**.
- 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 that is provided to the agent (see code).
- Every agent has a limit of 10GB of storage in the directory that is provided.
- 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 3 agents.
- The source code of agents must be submitted. This code will be included in the GeniusWebPython platform after the competition has 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 parallelisation, your agent will be run against every unique opponent concurrently. Take this into account when saving and loading data.**

Figure 1 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. Starting the next round is blocked until all the agents have finished their negotiation session. This allows all agents to have knowledge of all opposing agents at the start of every next session. A total of 50 negotiation sessions are run.

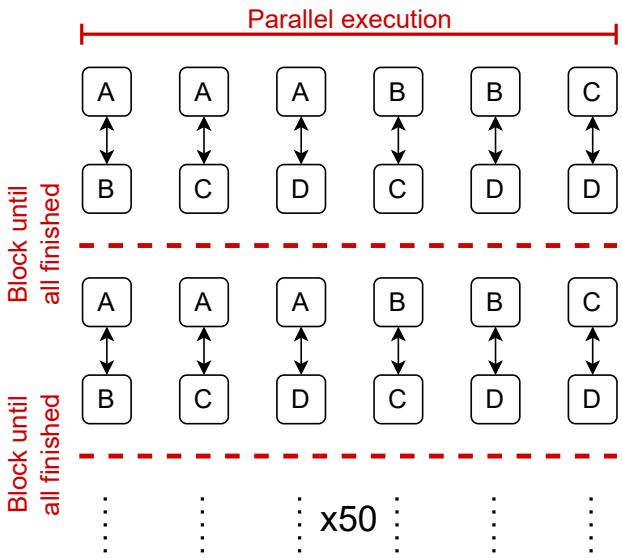

Figure 1: Example tournament with agents $\mathcal{A} = \{A, B, C, D\}$. The negotiation problem that the agents are negotiating over are randomly generated and never repeated.

4 Submission & questions

Participants submit their agent source code and academic report (optionally) in a zipped folder. For the code, just include the directory with your agent like the directory of the template_agent in the GitHub repository) in the zipped folder. Make sure that 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 has the option to prepare a 2-4 page report describing the design of their agent according to academic standards. The best teams that submit a report will be given the opportunity to give a brief presentation describing their agent at IJCAI 2022.

Furthermore, proceedings of the competition are planned to be published in a special issue. The report will be evaluated by the organisers of this league. For eligibility, the design should provide a contribution 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 decides to accept or reject a given 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 behaviour of the agent.

5 Fact sheet

Important dates

Submission deadline	10th of June 2022
Notification to finalists	1st of July 2022
Event	23st-29th of July 2022

Important links

Making an agent for ANAC 2022
Discord Server
Competition Website
Submission Form
Technical information GeniusWebPython

Contact

Bram Renting	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
Holger Hoos	Professor	Aachen University
Catholijn Jonker	Professor	Delft University of Technology

References

[1] R. Aydoğan, D. Festen, K. V. Hindriks, and C. M. Jonker, "Alternating offers protocols for multilateral negotiation," in *Studies in Computational Intelligence*, vol. 674, Springer, 2017, pp. 153–167, ISBN: 978-3-319-51563-2. DOI: 10.1007/978-3-319-51563-2_10.

[2] R. Lin, S. Kraus, T. Baarslag, D. Tykhonov, K. Hindriks, and C. M. Jonker, "Genius: An integrated environment for supporting the design of generic automated negotiators," *Computational Intelligence*, vol. 30, no. 1, pp. 48–70, 2014, ISSN: 08247935. DOI: 10.1111/j.1467-8640.2012.00463.x.