# GeniusWeb MOPaC Tutorial

Wouter Pasman

Interactive Intelligence, Delft University of Technology

September 27, 2021

This tutorial will help you using the MOPaC protocol in GeniusWeb. . It is assumed that you read the MOPaC protocol description and installed and basic knowledge of GeniusWeb, eg from the SAOP tutorial.

The main difference between SAOP and MOPaC is the events that parties need to respond to. Also the running of a session is slightly different and the log file contents differ.

## 1 Creating a MOPaC party

Creating a MOPaC party is almost identical to creating a SAOP party (see the SAOP tutorial). The differences:

1. a MOPAC-capable party has to return "MOPAC" as part of the Capabilities in its getCapabilities function.
2. a MOPAC capable party has to understand and properly handle MOPAC protocol events and create the appropriate actions. A short outline is on the wiki page, the details of this protocol are spelled out in the MOPAC Specification and MOPAC javadoc, but briefly:
   - handle incoming SessionSettings and fetch the profile
   - wait for YourTurn and send an Offer
   - wait for Voting and reply with Votes
   - wait for OptIn and reply again with Votes
   - loop back to wait for Yourturn above

The RandomParty already is MOPaC capable, so RandomParty can be used as a starting point ot create a MOPaC party.

## 2 Running a MOPaC session

After you put yuor compiled party on the TomCat server and started the server, navigate to the runserver webpage in your web browser. This should be available at a url that looks like: localhost:8080/runserver-x.y.z, where x.y.z is the version of the GeniusWeb server you have installed locally. Clicking on the new session link should take you to the Session creation webpage.

### 2.1 Options

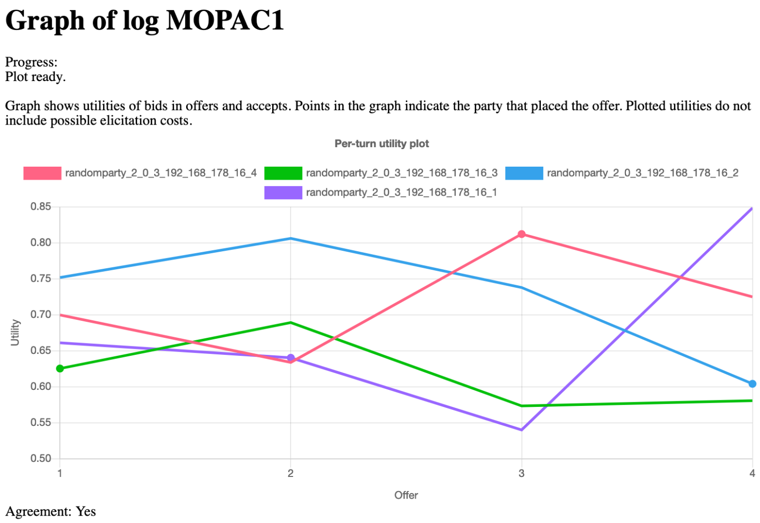
Here you have several options that you can tweak. We can broadly divide the options into two categories, the first is the set of options that are chosen for all the parties in the negotiation session. These are Protocol, Voting Evaluator, Deadline, and Domain. The rest of the settings are to be set per party. We discuss each of the options briefly below:

- Protocol: This option allows you to set the protocol for the negotiation session. Choose MOPAC since the assumption here is that you would like to run a MOPaC session.
- Voting Evaluator: This is method by which the protocol determines which deals form an agreement. There are currently two ways to do this:
  - Largest Agreement: Using this Voting Evaluation method, the earliest agreement with largest number of parties is chosen as the deal. Here, only 1 agreement is required and the negotiation stops soon after.
  - Largest Agreements and Repeat: Rather than stop after the first largest agreement, the rest of the parties that could not reach a consensus try to do so. The negotiation stops when either all parties have a consensus or 1 party is not part of a consensus (unless the deadline is reached, see below).
- Deadline: This is the time period for which the parties will negotiate. The scale of the time period can either be rounds or seconds. Since a second is a relatively large time scale for computers, using the scale of rounds would be easier to comprehend. For the MOPaC protocol, each round comprises of the four phases, namely Bidding, Voting, Opt-In and Termination/Continuation Phase.
- Domain: Here, you can choose the domain in which you would like the parties to negotiate. All domains that have been uploaded to the TomCat profilesserver and were parsed correctly are available here.
- Party: This allows you to choose the parties that participate in the negotiation from those available on the TomCat server.All parties that have been uploaded to the TomCat profilesserver are available here, but a few of these are not capable of doing MOPaC. Especially notice that randomparty is MOPaC capable but randompyparty is not. You can create, upload and run your own parties to your partiesserver.
- Profile: Here, you can choose the profile of the party. This describes the preferences of the party to various issues in the domain and affects how they will negotiate. All profiles that have been uploaded to the TomCat profilesserver and were parsed correctly are available here.
- Parameters: These are some extra protocol- and party- specific parameters that can be set for each party. For the MOPaC protocol, there is currently 1 parameter: power which controls the power that the party has in the negotiation. See Section 2 of the MOPaC Protocol document for more details about what the implications of the power of a party in the proceedings of the negotiation. The default power of a party, if not set explicitly, is 1. The range values that power can take is the set of natural numbers.

## 2.2 Steps to Run a Negotiation Session

Now to run a session, do the following

1. Go to the runserver with your browser and select ”new session”
2. Set the negotiation parameters as required.
3. Choose a MOPaC compatible party from the Party dropdown.
4. Choose a profile for the party.
5. Set the extra parameters for the party. This can be entered in the following way: ”a”: 1, ”b”: 2
6. Click on the add button to select the party with its profile for the negotiation session.
7. Repeat step 3-6 for all parties you want to add. Make sure to add at least 3 parties, each with a different profile, since MOPaC is a multi-lateral negotiation protocol.
8. Click on “Start Session” button to begin the session.



Graph of log MOPAC1

Progress:
Plot ready.

Graph shows utilities of bids in offers and accepts. Points in the graph indicate the party that placed the offer. Plotted utilities do not include possible elicitation costs.

Per-turn utility plot
randomparty_2_0_3_192_168_178_16_4
randomparty_2_0_3_192_168_178_16_3
randomparty_2_0_3_192_168_178_16_2
randomparty_2_0_3_192_168_178_16_1
Utility
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
1
2
3
4
Offer

Agreement: Yes

Figure 1: An Example Utilities Plot from a MOPaC negotiation session with 4 parties.

## 2.3 Utilities Plot

Once the session is completed, two links will be generated, one for the utilities plot and another for the log file. This plot is a graphical representation of the utilities of all the offers made in the session. The X axis captures the different offers made by parties in the session. The Y axis shows the utilties of the offers. An example utility plot is shown in figure 1. Here, each offer 1, 2, 3 and 4 is made by one of the parties (indeterminable in this plot). For each offer, the utility of each agent is plotted on the Y axis. Hence, the plot shows how the utilities for each agent differs for each offer. The plot makes no distinction between different rounds and so utilities offers made in subsequent rounds are plotted after the last offer in the previous round.

**Note** The line plot gives a false sense of sequence between the offers. Please keep in mind that this graphs shows utilities at each offer and not at each round. Each round comprises of 4 simultaneous offers, not sequential offers. We are still thinking of the most informative way to represent this multi-dimensional information. Please do write to us if you have any ideas.

## 2.4 Log File

The log file contains the final State - in this case a MOPACState - and is documented in the javadoc. It details how the session progressed - which parties were part of the session, who made which bid, what were each party's offers and the final agreements and their corresponding utilities. This state is stored as a JSON object. We describe the log file using an example log file as shown in listing 1:

Listing 1: json contents of MOPAC.json log file

```
{
    "MOPACState": {
        "phase": { ...  },
        "actions": [
            {
                "Offer": {
                    "actor": "randomparty_2_0_3_3",
                    "bid": {
                        "issuevalues": { ...  }
                    }
                }
            },
        ...,
        {
            "Votes": {
                "actor": "randomparty_2_0_3_6_3",
                "votes": [
                    {
                        "Vote": {
                            "actor": "randomparty_2_0_3_3",
                            "bid": {
                                "issuevalues": {... }
                            },
                            "minPower": 2,
                            "maxPower": 2147483647
                        }
                    },
                    {
                        "Vote": {
                            "actor": "randomparty_2_0_3_3",
                            "bid": {
                                "issuevalues": {...}
                            },
                            "minPower": 2,
                            "maxPower": 2147483647
                        }
                    }
                ]
            }
        },
    ...
    ],
    "progress": {
        "ProgressTime": {
            "duration": 600000,
            "start": 1632732640424
        }
    },
    "settings": {
        "MOPACSettings": {
            "participants": [ ...],
            "deadline": {
                "DeadlineTime": {
                    "durationms": 600000
                }
            },
            "votingevaluator": {
                "LargestAgreement": {
                }
            }
        }
    },
    "partyprofiles": {
        "randomparty_2_0_3_4": {
            "party": {
                "partyref": "http://..../randomparty-2.0.3",
                    "parameters": {
                    }
                },
                "profile": "ws://.../websocket/get/party/party4"
            },
            ...
        }
    }
}
```

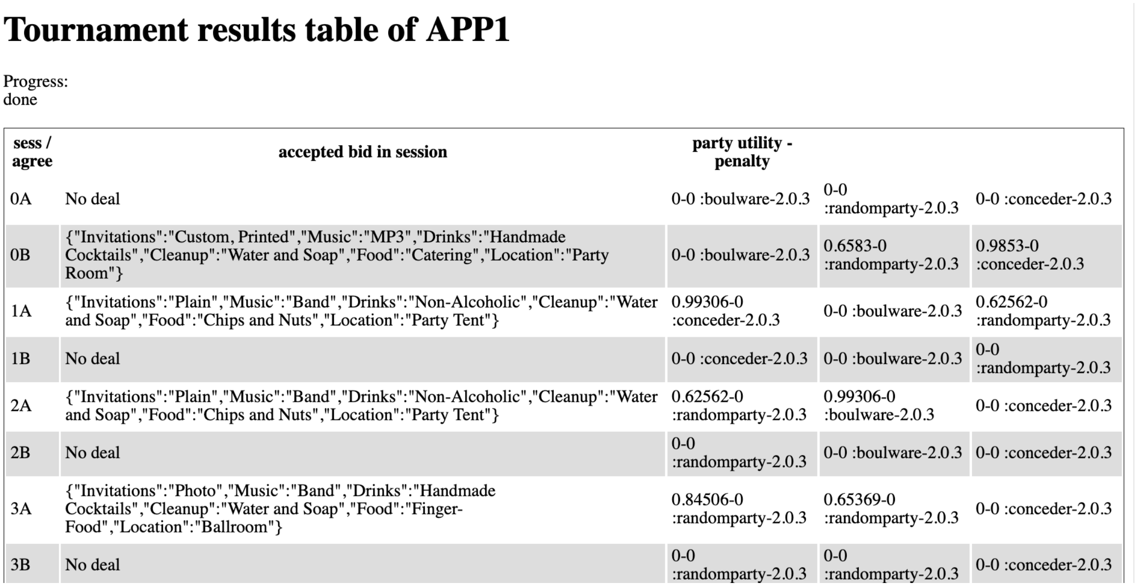
We see in the file:

- partyprofiles: Describes the parties that take part in the negotiation session and their preference profiles. Has a JSON object with a key for each party. Each party has
  - party.partyref: type of party,
  - party.parameters: extra parameters of the party,
  - profile: preference profile of the party.
- settings: Settings specific to the session that is being run. If running MOPaC protocol, the settings contains:
  - MOPACSettings.participants: The participants in the negotiation session. This is similar to the data in partyprofiles.
  - MOPACSettings.deadline: Contains the number of rounds and duration in milliseconds.
  - MOPACSettings.votingevaluator: Contains the type of voting evaluator used for the MOPaC session.
- progress: Has data about the duration of the session and when the ne- gotiation ended.
- connections: Has the reference to each party and the runtime errors if any including the stack trace.
- phase: The most important information from the phase is partyStates.agreements which contains the parties which participate in the largest agreement of the session. partyStates.actions contains information about the last action performed
- actions: This contains the actions performed by the parties in chronolog- ical order. Actions can be either of type Offer or Votes, corresponding to the bidding and voting phase of the MOPaC protocol.
  - Offer: Contains information about which party made the bid and the bid itself with its issue values.
  - Votes: Contains information about which party the votes belong to and a list of bids that the party accepted. Only the bids present in Votes.votes array were accepted by the party. Rest are rejected. In each vote, maxPower and minPower contains the minimum and maximum consensus threshold. See Section 2.1 of the MOPaC Protocol document for more information on this.

# 3 Running a MOPAC tournament

To run a MOPaC tournament, navigate to the run server page and click on the “new tournament” link. It should take you to a page with options to configure the settings for running a new tournament. To run a tournament:

1. Choose the number of times you want to repeat the tournament. Please note that each tournament could take a considerable amount of time if there are more than 3 parties in each session.



**Tournament results table of APP1**

Progress:
done

| sess / agree | accepted bid in session | party utility - penalty | | |
|---|---|---|---|---|
| 0A | No deal | 0-0 :boulware-2.0.3 | 0-0 :randomparty-2.0.3 | 0-0 :conceder-2.0.3 |
| 0B | {"Invitations":"Custom, Printed","Music":"MP3","Drinks":"Handmade Cocktails","Cleanup":"Water and Soap","Food":"Catering","Location":"Party Room"} | 0-0 :boulware-2.0.3 | 0.6583-0 :randomparty-2.0.3 | 0.9853-0 :conceder-2.0.3 |
| 1A | {"Invitations":"Plain","Music":"Band","Drinks":"Non-Alcoholic","Cleanup":"Water and Soap","Food":"Chips and Nuts","Location":"Party Tent"} | 0.99306-0 :conceder-2.0.3 | 0-0 :boulware-2.0.3 | 0.62562-0 :randomparty-2.0.3 |
| 1B | No deal | 0-0 :conceder-2.0.3 | 0-0 :boulware-2.0.3 | 0-0 :randomparty-2.0.3 |
| 2A | {"Invitations":"Plain","Music":"Band","Drinks":"Non-Alcoholic","Cleanup":"Water and Soap","Food":"Chips and Nuts","Location":"Party Tent"} | 0.62562-0 :randomparty-2.0.3 | 0.99306-0 :boulware-2.0.3 | 0-0 :conceder-2.0.3 |
| 2B | No deal | 0-0 :randomparty-2.0.3 | 0-0 :boulware-2.0.3 | 0-0 :conceder-2.0.3 |
| 3A | {"Invitations":"Photo","Music":"Band","Drinks":"Handmade Cocktails","Cleanup":"Water and Soap","Food":"Finger-Food","Location":"Ballroom"} | 0.84506-0 :randomparty-2.0.3 | 0.65369-0 :randomparty-2.0.3 | 0-0 :conceder-2.0.3 |
| 3B | No deal | 0-0 :randomparty-2.0.3 | 0-0 :randomparty-2.0.3 | 0-0 :conceder-2.0.3 |

Figure 2: An Example Results Table after running a MOPaC negotiation tournament.

2. Choose the number of teams in each session. It denotes the number of parties that will be chosen from a list of available parties. The number of teams in a session can vary from a minimum of 2 to a maximum of number of parties available.

3. Choose session protocol as "MOPAC", and Voting Evaluator and deadline. These three options are discussed in the previous section.

4. Next you choose the domain in which you want your parties to negotiate and add different preference profiles that you want to have available. Here you can choose any number of profiles, but it is suggested you have at least as many profiles as number of teams in a session since you don't want the profiles of different agents to repeat.

5. Next you choose the parties and their parameters. Please check the previous section for more details about these options. You should have added as many parties as the number of teams in a session.

6. Click the "Start Tournament" button.

7. When the tournament is started, you get options similar to what you get when you run a session at the bottom of the screen: link to the results table and the log file.

## 3.1 Results Table

For a tournament, there is no Utilities Plot. Instead, you have a results table, which is an overview of the results. It tries to access the original profiles and compute the utilities of the final bid. If there was no accepted bid, the results table shows the utility of the reservation bid in the profile. We show an example result table in figure 2. The table can be described column wise as follows:

- The first column shows the session number and agreement number, start- ing from 0A, where 0 is the session number and A is the first agreement. If there were more agreements in session 0, they would be shown in 0B, 0C and so on.

- The second column shows the issue values of the bid that was accepted.

- Columns 3 and onward show the utility of the agreement for each party and the penalty incurred by the party. This is represented as follows: ¡utility¿-¡penalty¿. If a party was not part of an agreement, the column contains 0-0, i.e. no penalty and no utility.

## 3.2 Log File

The log file of a tournament is slightly different compared to the log file for the session: it now conains the final state of the tournament: an AllPermutationsState. We show an example log file in listing 2. The following points summarize the structure of the log file:

- AllPermutationsState.toursettings.AllPermutationsSettings contains teams which has the full set of parties available for the tournament, profilelists which has the full set of profiles available and sessionsettings which has the settings chosen in the run tournaments webpage.

- AllPermutationsState.sessionsettings is an array of the various permutations of profiles and parties chosen according to the run tournament settings. This is generated by the AllPermutationState, according to the ”toursettings”, and added to the log file for convenience. Each element has a list of participants (party + profile), deadline, and voting evaluator type.

- AllPermutationsState.results is an array of SessionResults, one for each session. The log file does not contain the fiull MOPACState because the log file would become too large. Each SessionResult contains participants which is a list of parties with profiles, and agreements which is the set of parties that have come to an agreement along with the issue values.

Listing 2: json contents of APP.json log file with tournament result

```
{
    "AllPermutationsState": {
        "toursettings": {
            "AllPermutationsSettings": {
                "teams": [
                    {
                        "Team": [
                            {
                                "partyref": "http://.../randomparty-2.0.3",
                                "parameters": {
                                }
                            }
                        ]
                    },
                ...
                ],
                "profileslists": [
                    {
                        "ProfileList": [
                            "ws://.../party/party1"
                        ]
                    },
                    ...
                ],
                "reuseTeams": false,
                "teamsPerSession": 3,
                "sessionsettings": {
                    "MOPACSettings": {
                        "participants": [

                        ],
                        "deadline": {
                            "DeadlineRounds": {
                                "rounds": 10,
                                "durationms": 10000
                            }
                        },
                        "votingevaluator": {
                            "LargestAgreement": {   }
                        }
                    }
                },
                "numberTournaments": 1
            }
        },
        "sessionsettings": [
            {
                "MOPACSettings": {
                    "participants": [
                        {
                            "TeamInfo": {
                                "parties": [
                                    {
                                        "party": {
                                            "partyref": "http://...randomparty-2.0.3",
                                            "parameters": {
                                            }
                                        },
                                        "profile": "ws://...party/party3"
                                    }
                                ]
                            }
                        },
                    ...
                    ],
                    "deadline": {
                        "DeadlineRounds": {
                            "rounds": 10,
                            "durationms": 10000
                        }
                    },
                    "votingevaluator": {
                        "LargestAgreement": {   }
                    }
                }
            },
        ........
        ],
        "results": [
            {
                "participants": {
                    "boulware_2_0_3_3": {
                        "party": {
                            "partyref": "http://...boulware-2.0.3",
                            "parameters": { }
                        },
                        "profile": "ws://.../party/party4"
                    },
                    "randomparty_2_0_3__1": {
                        "party": {
                            "partyref": "http://192.168.178.16:8080/partiesserver/run/randomparty-2.0.3",
                            "parameters": {     }
                        },
                        "profile": "ws://.../party/party3"
                    },
                    "conceder_2_0_3_2": {
                        "party": {
                            "partyref": "http://..conceder-2.0.3",
                            "parameters": {     }
                        },
                        "profile": "ws://...party/party2"
                    }
                },
                "agreements": {
                    "randomparty_2_0_3__1": {
                        "issuevalues": {...}
                    },
                    "conceder_2_0_3_192_168_178_16_2": {
                        "issuevalues": {...}
                    }
                },
                "penalties": {
                },
                "error": null
            },
            ......
        ]
    }
}
```