GeniusWeb Manual

Academic Year 2020-2021

1 Requirements

In order to successfully perform the following operations you need to:

- Have TomCat installed on your pc: you can do so from their website https://tomcat.apache.org/download-80.cgi
- Have copied inside the webapps directory in your TomCat home folder the most recent version of
  - partiesserver-x.y.z.war, which you can find in the folder hosted here[1]
  - profilesserver-x.y.z.war, which you can find in the folder hosted here
  - runserver-x.y.z.war, which you can find in the folder hosted here
- Have started TomCat at least once after updating the three files mentioned at the previous point by executing `startup.sh` in the `/bin` folder of the TomCat home directory.

If any of the above steps seem ambiguous, please find the tutorial videos at `https://ii.tudelft.nl/GeniusWeb/students.html` which cover how to setup WebGenius locally for development.

2 Setting up MOPaC Domains

To set up a MOPaC domain we start by creating a folder (wherever we want in our computer, we will move it later into the appropriate location). In our example we will call it "Wedding". In here we will save both the domain and profiles files (see next section for the latter).

We have to create a JSON file with the following structure:

[1]NOTE: your browser must have cookies enabled to access the artifactory

```
{
    "name": "name_of_the_domain",
    "issuesValues": {
        "issue_1_name": {
            "values": [
                "value_one",
                "value_two"
            ]
        },
        "issue_2_name": {
            "range": [start, end, step_size]
        },
        ...
    }
}
```

As you can see, it is also possible to use a range field to save some time/lines and specify a sequence of values just with the starting point, the ending point and a step size. For our example we have[2]:

```
{
    "name": "wedding",
    "issuesValues": {
        "menu": {
            "values": [
                "fish",
                "meat",
                "vegetarian",
                "vegan",
                "all"
            ]
        },
        "cost": {
            "values": [
                "2000 eur",
                "5000 eur",
                "9000 eur"
            ]
        },
        "participants": {
            "range": [100,200,20]
        }
    }
}
```

[2]The person writing this is not an expert wedding planner.

In the next section we will implement an example profile and then proceed uploading the folder to our TomCat server.

3 Setting up MOPaC Profiles

When implementing a profile, we will have to specify the utilities for every value we specified in our domain. Also, we will specify the weight of each attribute.

A general profile JSON file is shown on the following page. A couple of important things must be noted:

- All the profiles must have different names. They must also be different from the name of the domain file.
- The utilities of each issue, specified in "issueUtilities", might not add up to 1.
- The issue weights, specified in "issueWeights", must add up to 1.
- The domain name must be the same as the one specified in the domain JSON file.
- Issue names and values must be the same as specified in the domain JSON file.
- The domain field must be the same as the domain JSON file (you can just copy and paste the content of the domain file).

```
{
    "LinearAdditiveUtilitySpace": {
        "name": "name_of_the_profile",
        "issueUtilities": {
            "issue_1_name":{
                "discreteutils": {
                    "valueUtilities":{
                        "value_one": utility_1,
                        "value_two": utility_2
                    }
                }
            },
            "issue_2_name":{
                "numberutils": {
                    "lowValue": start,
                    "lowUtility": util_value,
                    "highValue": end,
                    "highUtility": util_value
                }
            },
            ...
        },
        "issueWeights": {

            "issue_1_name": weight_1,
            "issue_2_name": weight_2,
            ...
        },
        "domain":{
            "name": "name_of_the_domain",
            "issuesValues": {
                "issue_1_name": {
                    "values": [
                        "value_one",
                        "value_two"
                    ]
                },
                "issue_2_name": {
                    "range": [start, end, step_size]
                },
                ...
            }
        }
    }
}
```

For our example, we will call our profile `wedding1` and it will be:

```json
{
    "LinearAdditiveUtilitySpace": {
        "name": "wedding1",
        "issueUtilities": {
            "menu":{
                "discreteutils": {
                    "valueUtilities":{
                        "fish": 0.3,
                        "meat": 0.5,
                        "vegetarian": 1.0,
                        "vegan": 0.4,
                        "all": 0.2
                    }
                }
            },
            "cost":{
                "discreteutils": {
                    "valueUtilities":{
                        "2000 eur": 0.6,
                        "5000 eur": 1.0,
                        "9000 eur": 0.3
                    }
                }
            },
            "participants":{
                "numberutils": {
                    "lowValue": 100,
                    "lowUtility": 0.2,
                    "highValue": 200,
                    "highUtility": 1.0
                }
            }
        },
        "issueWeights": {

            "menu": 0.4,
            "cost": 0.5,
            "participants": 0.1
        },
        "domain":{
            "name": "wedding",
            "issuesValues": {
                "menu": {
                    "values": [
                        "fish",
                        "meat",
                        "vegetarian",
                        "vegan",
                        "all"
                    ]
                },
                "cost": {
                    "values": [
                        "2000 eur",
                        "5000 eur",
                        "9000 eur"
                    ]
                },
                "participants": {
                    "range": [100,200,20]
                }
            }
        }
    }
}
```

You should of course create more than one profile in order to properly run a negotiation session (and a tournament). You can duplicate one of the profiles and change each value that needs to be changed: remember to also change the profile name.

Now that we created our profile(s) and domain JSON files, we can move our folder to `webapps/profilesserver-x.y.z/domainsrepo/` located in our TomCat home path. You should now be able to see them when creating a new session or tournament in your local GeniusWeb instance. If you cannot see them, we suggest you to check the `catalina.out` file in the `log/` folder available at TomCat home directory.

Note: We do not have a great way to interactively edit and debug the errors in the domain and profiles files as of yet. So we suggest that you run the following command: `tail -f logs/catalina.out` while editing the domains and profiles file with your preferred text editor. It should give you a live view of the file with the latest errors as you save the profile/domain configurations.

4 Running a Session with MOPaC

After the TomCat server is running, 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. 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:

1. **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.

2. **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).

3. **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.

4. **Domain**: Here, you can choose the domain in which you would like the parties to negotiate. All parties that have been uploaded to the TomCat profilesserver are available here. Please check section 2 to see how to set this up.

5. **Party**: This allows you to choose the parties that participate in the negotiation from those available on the TomCat server. As of now, only the randomparty-x.y.z (x.y.z depends on your version of the partiesserver), is compatible with MOPaC protocol. Please create and upload your own parties on to your hosted servers.

6. **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 are available here. Please check section 3 to see how to set this up.

7. **Parameters**: These are some extra protocol 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.

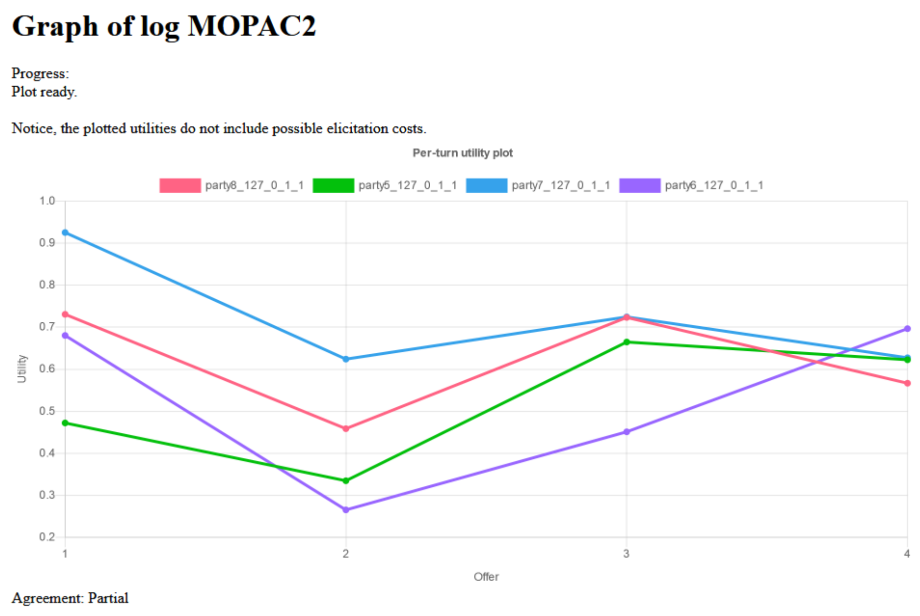
Steps to Run a Negotiation Session:

1. Set the negotiation parameters as required.
2. Choose a MOPaC compatible party from the Party dropdown.
3. Choose a profile of preferences for the party.
4. Set the extra parameters for the party. This can be entered in the following way: `"a": 1, "b": 2`
5. Click on the add button to select the party with its profile for the negotiation session.
6. Repeat from step 2 3 or more times. Make sure to add at least 3 parties, each with a different profile, since MOPaC is a multi-lateral negotiation protocol.
7. Click on "Start Session" button to begin the session.
8. Once the session is completed, two links will be generated, one for the utilities plot and another for the log file.

Utilities Plot: 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.

Log File Since the Utility Plot gives only a single perspective of the session, we make available a log file which contains all the low level details of 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. The log is organized as a JSON object. We describe the log file using an example log file as shown in listing 1:



Graph of log MOPAC2

Progress:
Plot ready.

Notice, the plotted utilities do not include possible elicitation costs.

Agreement: Partial

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

```
{
    "partyprofiles": {
        "party1": {
            "party": {
                "partyref": "http://127.0.1.1:8080/partiesserver-x.y.z/run/randomparty-x.y.z",
                "parameters": { "power": 1 }
            },
            "profile": "ws://127.0.1.1:8080/profilesserver-x.y.z/websocket/get/party/party1"
        },
        ...
    },
    "settings": {
        "MOPACSettings": {
            "participants": [...],
            "deadline": {
                "deadlinerounds": {
                    "rounds": 3,
                    "durationms": 10000
                }
            },
            "votingevaluator": {
                "LargestAgreementsLoop": {}
            }
        }
    },
    "progress": {
        "rounds": {
            "duration": 8,
            "currentRound": 1,
            "endTime": 1601223817378
        }
    },
    "connections": [
        {
            "reference": "http://127.0.1.1:8080/partiesserver-x.y.z/run/randomparty-x.y.z",
            "error": {
                "cause": null,
                "stackTrace": []
            }
        }
    ],
    "phase": {
    },
    "actions": [
        {
            "offer": {
                "actor": "party1",
                "bid": {
                    "issuevalues": {...}
                }
            }
        },
        ...
        {
            "Votes": {
                "actor": "party1",
                "votes": [
                    {
                        "actor": "party1",
                        "bid": {
```

```json
                            "issuevalues": {...}
                        },
                        "minPower": 2,
                        "maxPower": 4
                    },
                    {
                        "actor": "party1",
                        "bid": {
                            "issuevalues": {
                                ...
                            }
                        },
                        "minPower": 2,
                        "maxPower": 4
                    },
                    ...
                ]
            }
        },
        ...
    ],
    "phase": {
        "partyStates": {
            "actions": [
                ...
            ],
            "agreements": {
                "party1": {
                    "issuevalues": {
                        ...
                    }
                },
                "party2": {
                    "issuevalues": {
                        ...
                    }
                },
                ...
            },
            ...
        }
    }
}
```

Listing 1: Example Log File after running a MOPAC session

- **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 negotiation 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 chronological 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.

5 Running a Tournament with MOPaC

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.

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.

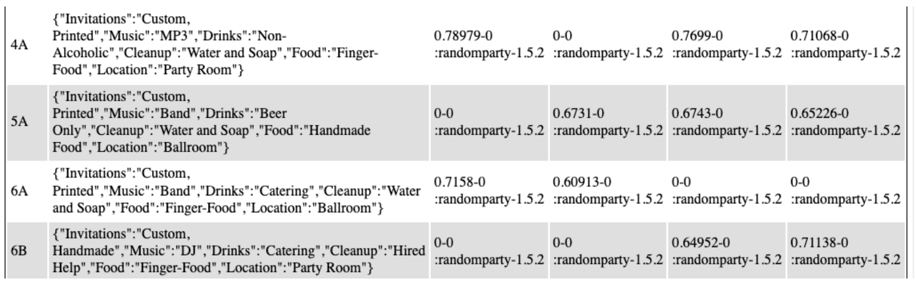
3. Choose session protocol as "MOPAC", and Voting Evaluator and deadline. These three options are discussed in the previous section.

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.



4A	{"Invitations":"Custom, Printed","Music":"MP3","Drinks":"Non-Alcoholic","Cleanup":"Water and Soap","Food":"Finger-Food","Location":"Party Room"}	0.78979-0 :randomparty-1.5.2	0-0 :randomparty-1.5.2	0.7699-0 :randomparty-1.5.2	0.71068-0 :randomparty-1.5.2
5A	{"Invitations":"Custom, Printed","Music":"Band","Drinks":"Beer Only","Cleanup":"Water and Soap","Food":"Handmade Food","Location":"Ballroom"}	0-0 :randomparty-1.5.2	0.6731-0 :randomparty-1.5.2	0.6743-0 :randomparty-1.5.2	0.65226-0 :randomparty-1.5.2
6A	{"Invitations":"Custom, Printed","Music":"Band","Drinks":"Catering","Cleanup":"Water and Soap","Food":"Finger-Food","Location":"Ballroom"}	0.7158-0 :randomparty-1.5.2	0.60913-0 :randomparty-1.5.2	0-0 :randomparty-1.5.2	0-0 :randomparty-1.5.2
6B	{"Invitations":"Custom, Handmade","Music":"DJ","Drinks":"Catering","Cleanup":"Hired Help","Food":"Finger-Food","Location":"Party Room"}	0-0 :randomparty-1.5.2	0-0 :randomparty-1.5.2	0.64952-0 :randomparty-1.5.2	0.71138-0 :randomparty-1.5.2

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

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, starting 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.

Log File:
The log file of a tournament is slightly different compared to the log file for the session. 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. Each element has a list of participants (party + profile), deadline, and voting evaluator type.
- `AllPermutationsState.results` is an array of results for each session. For each session, it 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.

```
{
  "AllPermutationsState": {
    "toursettings": {
      "AllPermutationsSettings": {
        "teams": [
          {
            "Team": [
              {
                "partyref": "http://127.0.1.1:8080/
                    partiesserver-x.y.z/run/randomparty
                    -x.y.z",
                "parameters": {}
              }
            ]
          },
          ...
        ],
        "profileslists": [
          {
            "ProfileList": [
              "ws://127.0.1.1:8080/profilesserver-x.y.
                 z/websocket/get/party/party1"
            ]
          },
          ...
        ],
        "sessionsettings": {
          "MOPACSettings": {
            "deadline": {
              "deadlinerounds": {
                "rounds": 10,
                "durationms": 10000
              }
            },
            "votingevaluator": {
              "LargestAgreement": {}
            }
          }
        },
        ...
      }
    },
    "sessionsettings":  [
        {
            "MOPACSettings": {
              "participants": [
                {
                    ...
                },
                ...
              ],
              ...
            }
          },
          ...
    ],
    "results": [
        {
            "participants": {
              "party1": {
                ...
              },
              ...
            },
            "agreements": {
              "party1": {
                "issuevalues": {
                    ...
                }
              },
              "party2": {
                "issuevalues": {
                    ...
                }
              }
            },
            ...
        },
      ...
    ]
  }
}
```

Listing 2: Example Log File after running a MOPAC tournament