

COMP6203: Lab #1

Intelligent Agents
University of Southampton

This lab introduces the concepts domains, issues, preference profiles and outcome spaces in detail. In addition it will introduce the GENIUS platform.

If you are unable to finish the lab in time, make sure you finish it in your own time before starting the next lab. Conversely, if you finish early you can start on the next one already. Note these labs are not marked but are designed to help you with the coursework.

Contents

1 Theory Crash Course

- 1.1 Domain, Issues, Values and Outcomes
- 1.2 Preference Profiles and Utilities

2 Task 1: Understanding Preference Profiles

- 2.1 Exercise 1:
- 2.2 Exercise 2:

3 Task 2: A first look at GENIUS

- 3.1 Creating the Laptop Domain in GENIUS
- 3.2 Add the preference profiles to the domain
- 3.3 Double check the created XML

4 Task 3: Running a Negotiation Session

1 Theory Crash Course

This section provides the essential theory to understand a negotiation process in GENIUS platform.

1.1 Domain, Issues, Values and Outcomes

Suppose we have a setting with two agents, a buyer (Agent A) and a seller (Agent B), wanting to negotiate on a laptop. A domain D specification determines what type of offers that can be made by the agents during negotiation. For instance, if agents can negotiate the price, hard disk size and monitor size of a laptop, a domain that describes this negotiation could look like this:

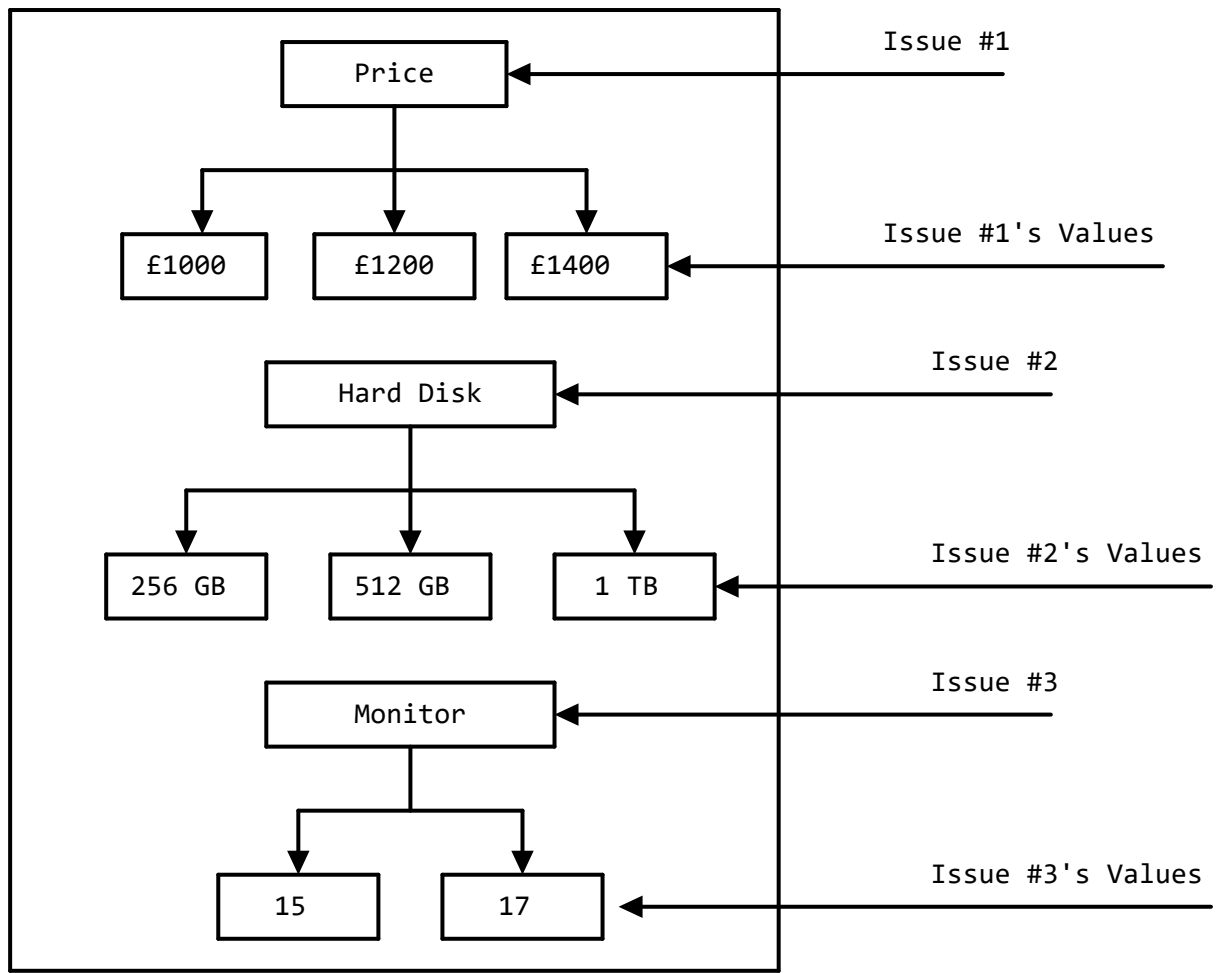


Figure 1: An example domain for laptop negotiation

Next we introduce the formal notation needed to describe a domain.

A domain contains n issues:

$$D = \{I_1, \dots, I_n\} \quad (1)$$

Each issue i consists of k_i values:

$$\forall I^i \in D \quad I_i = \{v_i^1, v_i^2, \dots, v_i^{k_i}\} \quad (2)$$

The outcome space, i.e. set of all possible negotiation outcomes/bids, is then the given by:

$$O = I_1 \times I_2 \times \dots \times I_n \quad (3)$$

Example: With this notation, the negotiation scenario in Figure 1 can be formally defined as:

$$D_{laptop} = (Price, HardDisk, Monitor) \quad (4)$$

$$I_{Price} = (£1000, £1200, £1400)$$

$$I_{HardDisk} = (256GB, 512GB, 1TB)$$

$$I_{Monitor} = (15, 17)$$

An example of an *outcome* or *bid* $o \in O$ of the negotiation is:

$$o = (Price: £1200, HardDisk: 1TB, Monitor: 15) \quad (5)$$

1.2 Preference Profiles and Utilities

Agents have different preferences for different outcomes. For example, the buyer agent A prefers a lower price but also prefers a larger screen. The seller agent B prefers to sell for a higher price and the price matters most, while the screen size does not matter as much.

To describe these preferences, each agent has so-called a *preference profile* that represents the trade offs between values and between issues, and in GENIUS this preference profile is described using an **additive utility function**:

Additive Utility Function

The utility for agent j of an outcome $o = (o_1, \dots, o_n)$ is calculated as:

$$U_j(o) = \sum_{i=1}^n w_j^i \frac{eval_j(o_i)}{\max(eval_j(I_i))} \quad (6)$$

where o_i is the value of issue i , $eval_j(o_i)$ is agent j 's evaluation of issue i , $\max(eval_j(I_i))$ is a normalization factor ensuring the value is between 0 and 1, and w_j^i is j 's weight for issue i where $\sum_{i=1}^n w_j^i = 1.0$

Considering again the Laptop domain, suppose the preference profile of agent A (the buyer) is as follows:

Issue	Weight	Evaluations
Price	0.3	$eval(\pounds 1000) = 100, eval(\pounds 1200) = 50, eval(\pounds 1400) = 0$
Hard disk	0.4	$eval(256GB) = 0, eval(512GB) = 6, eval(1TB) = 10$
Monitor	0.3	$eval(15) = 0, eval(17) = 1$

Table 1: Preference profile of Agent A (buyer).

Furthermore, the preference profile of Agent B is:

Issue	Weight	Evaluations
Price	0.6	$eval(\pounds 1000) = 0, eval(\pounds 1200) = 50, eval(\pounds 1400) = 100$
Hard disk	0.3	$eval(256GB) = 3, eval(512GB) = 2, eval(1TB) = 1$
Monitor	0.1	$eval(15) = 1, eval(17) = 0$

Table 2: Preference profile of Agent B (seller).

Often a goal in the negotiation is to achieve a mutually beneficial agreement. This is captured by the notion of Pareto optimality.

Definition: Pareto Optimality

An outcome is said to be *Pareto optimal* (a.k.a *Pareto efficient*) if there exists no other outcome where all agents are better off and no agent is worse off.

We will explore Pareto optimality in the exercise that follows.

2 Task 1: Understanding Preference Profiles

Make sure you have read and understood the theoretical concepts described above before moving on.

2.1 Exercise 1:

Calculate the utility of outcome $o = (\text{Price: } \pounds 1200, \text{ HardDisk: } 1\text{TB}, \text{ Monitor: } 15)$ for Agents A and B given Table 1 and Table 2 respectively.

Once you have completed the exercise, press below to reveal the answer.

[Click Here to Reveal/Hide the Answer](#)

2.2 Exercise 2:

In the next exercise we are going to determine which outcomes in the outcome space are Pareto optimal (see definition above). To help with this, we first plot the outcomes based on the utilities of the two agents. This is also called the **utility space** and is visualised in Figure 2 below.

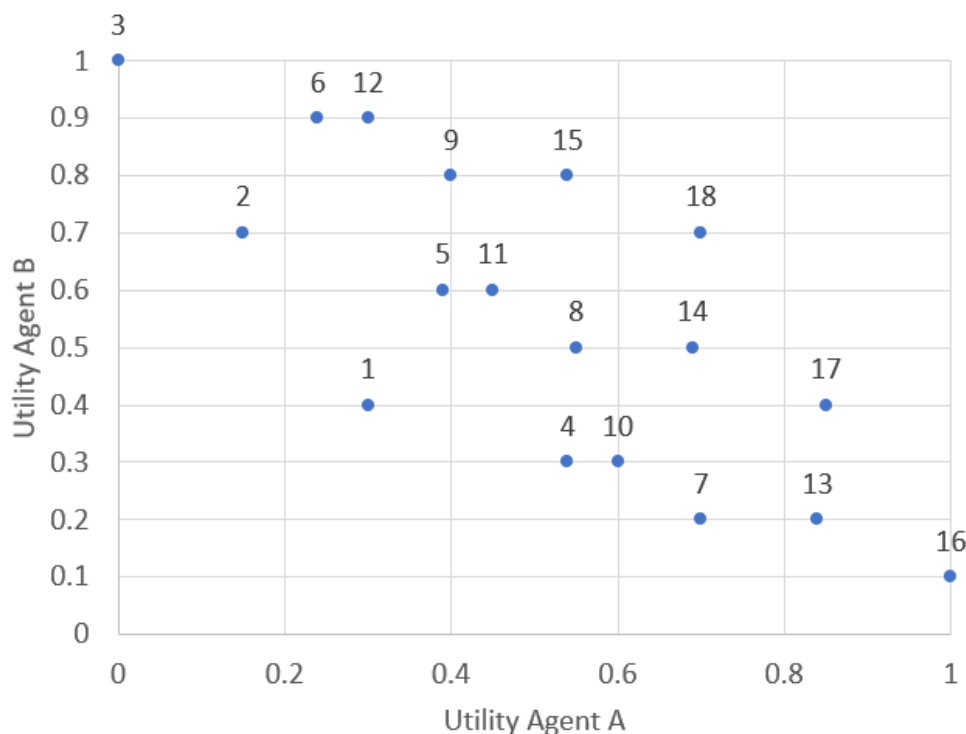


Figure 2: Negotiation outcomes for the laptop domain.

Note that each point in the figure corresponds to a point from Table 3. The labels above the data points show the outcome IDs which match those in the table.

Once you have understood this figure, write down which of the outcomes are Pareto optimal using the outcome IDs.

[Click Here to Reveal/Hide the Answer](#)

3 Task 2: A first look at GENIUS

At this point, if you are expected to be familiar with the theoretical concepts. Make sure that you understand the topics above before you continue.

Download the latest version of the GENIUS platform from the website:

<http://ii.tudelft.nl/genius/>

Unzip the content and run GENIUS by double clicking on the `genius-9.x.xx.jar` file (where x depends on the version).



Save all files in a place you can access the later, e.g. in the H: drive or on your own laptops.



Make sure Java 1.8 (JRE or JDK version 8) is installed on the computer you are using. **GENIUS will not work with other versions of Java.**

Binary distributions for all platforms are available from [AdoptOpenJDK](#). Both the install version and zip version, which does not require installation, are available (use the latter if you do not have admin rights to the computer you are using).

Note that for later labs you will need the Java Development Kit (JDK). The Java Runtime Environment (JRE) only allows you to run java programs, which is fine for now, but will not enable you to compile code which you need when developing your own agent.



If double clicking does not execute GENIUS, try using the command line as follows. First, go to the GENIUS working directory and then enter:

```
{JAVA_DIRECTORY}/java -jar genius-9.x.xx.jar
```

where {JAVA_DIRECTORY} refers to the directory where the java binaries are located for java 1.8. Note that if the PATH environment variable already includes this directory then you can simply use the `java` command without the directory. However, check that it points to the correct Java version by entering `java -version`.

Notice that, at start up, GENIUS will initialise by generating a number of `.xml` files. In particular, in the `etc` directory is where the domain profiles are kept. We will return to this later.

3.1 Creating the Laptop Domain in GENIUS

We are now going to create the laptop domain as in [Table 1](#) and [Table 2](#) above. You will see the existing domains in the Domains tab. First, right click on the domains and choose to create a new domain.



Note there already exists a laptop domain in GENIUS. Hence, use e.g. **lab1_laptop** as the name for the new domain.

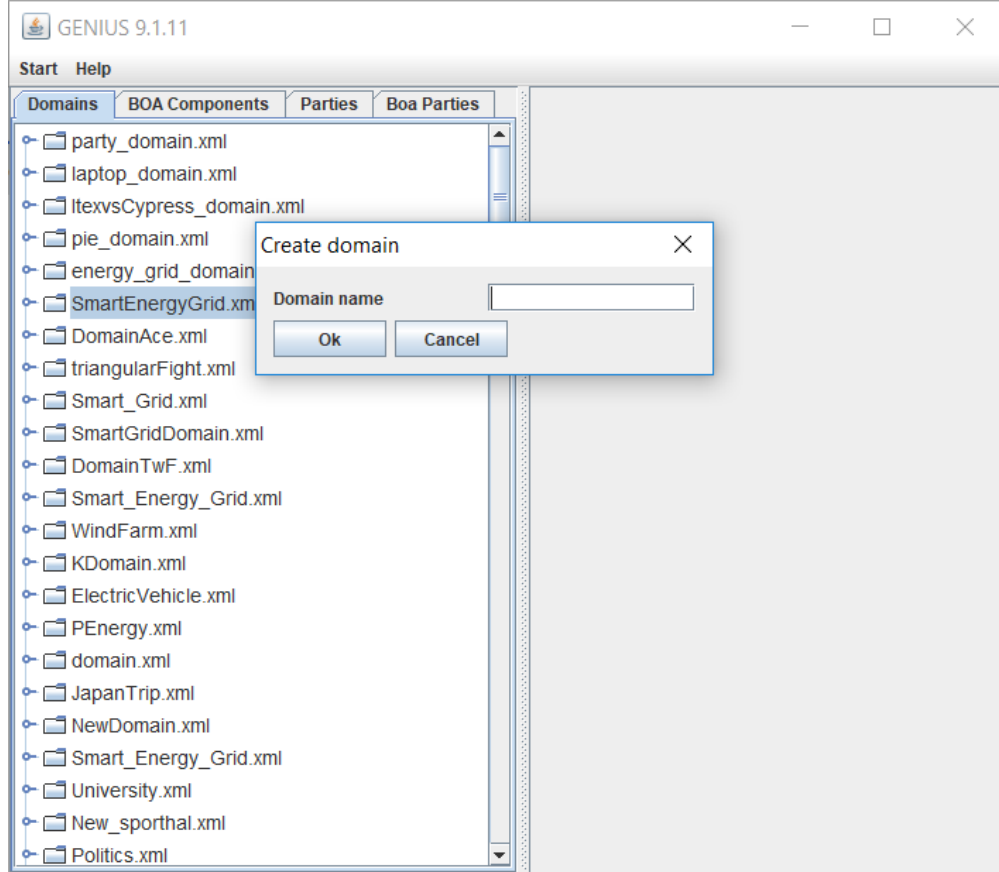


Figure 3: Right click in the left bar

Next, add the issues to the domain by clicking the +Add Issues button. Enter the name of the issue and the values. Each value should be on a new line.

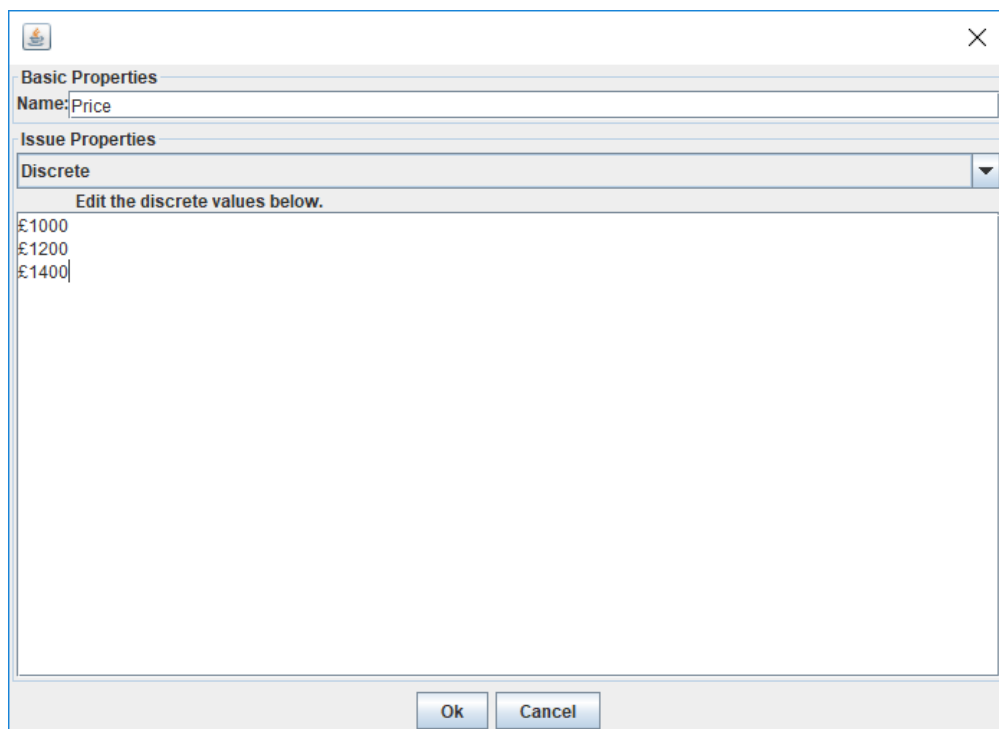


Figure 4: Right click in the left bar

Do this for all issues and then save. After you finished it should look like this:

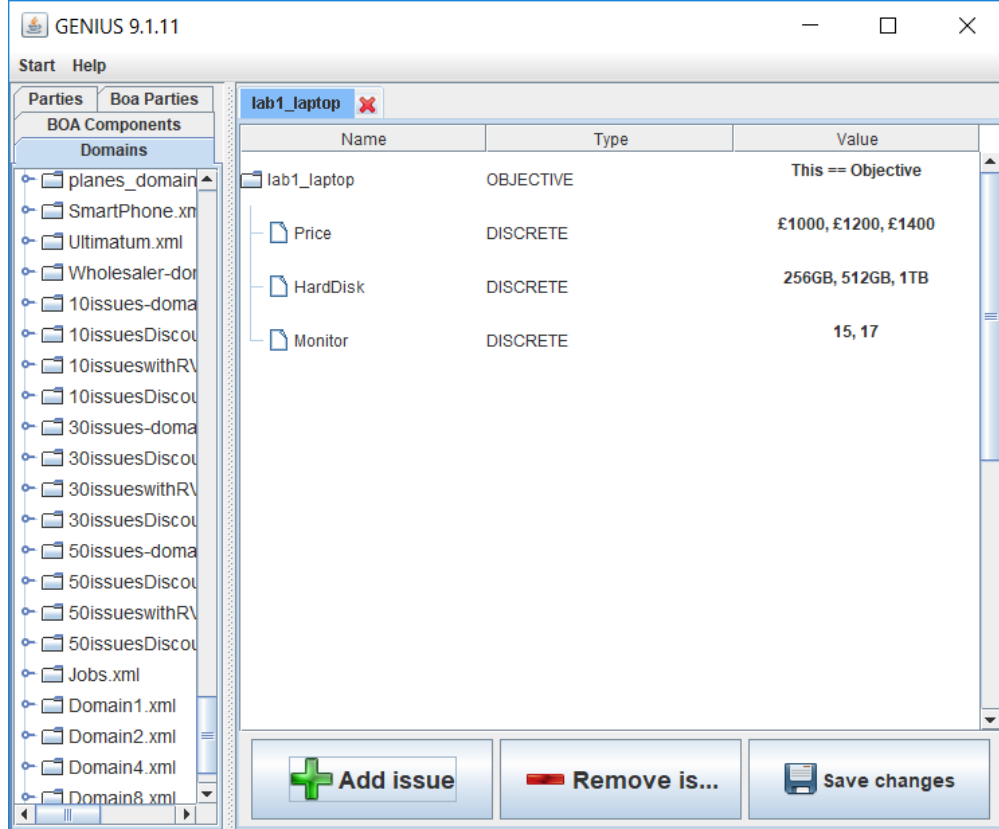



Figure 5: Right click in the left bar

3.2 Add the preference profiles to the domain

Now create a preference profile for Agents A and B as follows. First, right click on the domain's name and choose New preference profile. Then set the weights according to the preference profile of Agent A first (see Table 1 above). This should look like figure 6 below.

 Tick the box next to the issue to fix the weights of that issue.

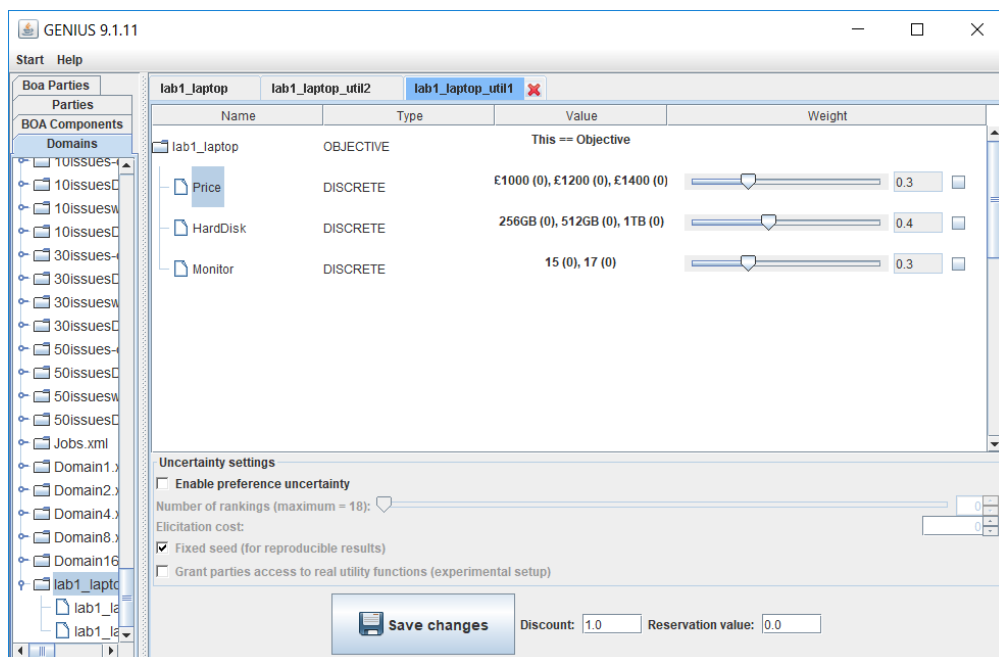


Figure 6: Right click to domain's name

Next, to set the evaluation values for each issue, double click on the issue you want to set the values of. Again set the values according to Table 1. Now create another preference profile for agent B using the weights and evaluation values in Table 2.



Make sure you keep the box Enable preference uncertainty unticked. We will consider preference uncertainty in a later lab. Also, we will only work with discrete issues. Finally, note that you can set the Discount factor and Reservation value. Keep these to the default values.

Don't forget to save any changes.

3.3 Double check the created XML

The domain and the preference profiles are stored as xml files. In fact, you can create your domain directly from the xml file rather than using the GENIUS interface.

Once you create the domain and the preference profiles and the domain, have a look at the created XML files (e.g. using Notepad++). They are by default located at:

```
{WORKING_DIRECTORY}/genius{version}/etc/templates/lab1_laptop
```

Check whether lab1_laptop_util1.xml is similar as below:

```
<utility_space>
  <objective name="lab1_laptop" index="0" description="" etype="objective"
type="objective">
    <issue vtype="discrete" name="Price" index="1" etype="discrete"
type="discrete">
      <item evaluation="100.0" index="1" value="£1000">
      <item evaluation="50.0" index="2" value="£1200">
      <item evaluation="0.0" index="3" value="£1400">
    </item></item></item></issue>
    <issue vtype="discrete" name="HardDisk" index="2" etype="discrete"
type="discrete">
      <item evaluation="0.0" index="1" value="256GB">
      <item evaluation="6.0" index="2" value="512GB">
      <item evaluation="10.0" index="3" value="1TB">
    </item></item></item></issue>
    <issue vtype="discrete" name="Monitor" index="3" etype="discrete"
type="discrete">
      <item evaluation="0.0" index="1" value="15">
      <item evaluation="1.0" index="2" value="17">
    </item></item></issue>
    <weight index="1" value="0.29816346554144846">
    <weight index="2" value="0.4043304598130849">
    <weight index="3" value="0.2975060746454666">
  </weight></weight></weight></objective>
  <discount_factor value="1.0">
  <reservation value="0.0">
</reservation></discount_factor></utility_space>
```

Figure 7: Contents of lab1_laptop_util1.xml

4 Task 3: Running a Negotiation Session

A *negotiation session* is a single bilateral or multilateral negotiation. In this task you will use the GENIUS interface to run a negotiation session. Note that it is also possible to run negotiations using the command line and without using the interface. This is useful when testing the agent you create and running many experiments. We will return to this in later labs.

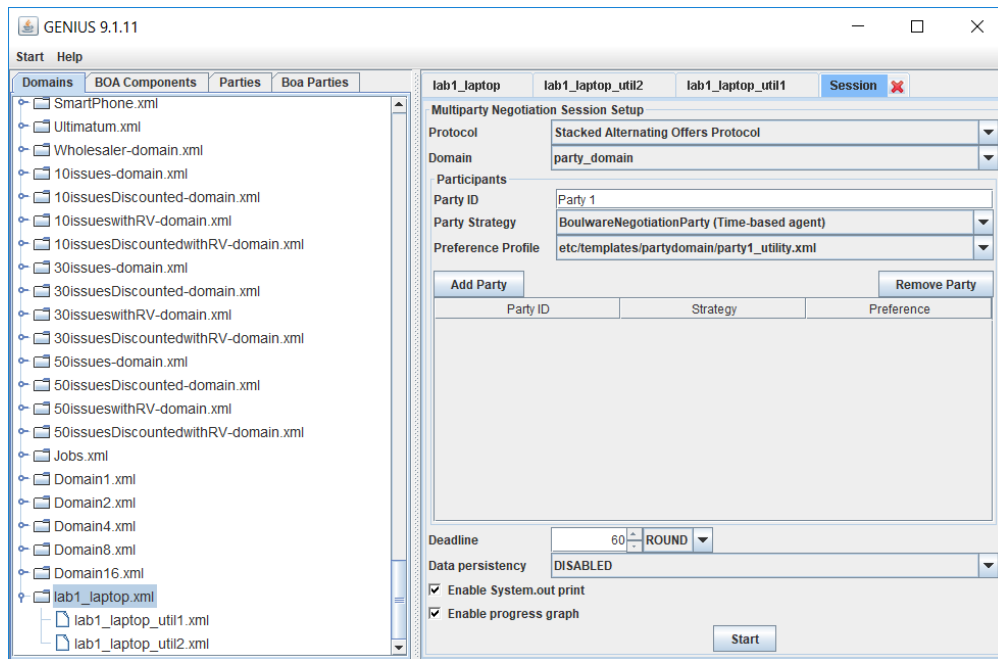


Figure 8: Running a Negotiation Session

- Under Start, choose Negotiation. The Session tab should now appear as shown in Figure 8.
- For the Protocol, keep the default Stacked Alternating Offers Protocol SAOP. You can read the GENIUS user guide to learn about the different protocols, we will use SAOP as the main protocol throughout.
- Choose the lab1_laptop domain from the drop down list (it will be at the end).
- Next, you will need to configure the participants. Each participant has its own negotiation strategy and preference profile.
- Select e.g. the BouldwareNegotiationParty for Agent A and party1_utility as the preference profile (the default). Choose Add Party.
- Select a different strategy for Agent B and make sure the preference profile is party2_utility.

The result should look something like Figure 9.

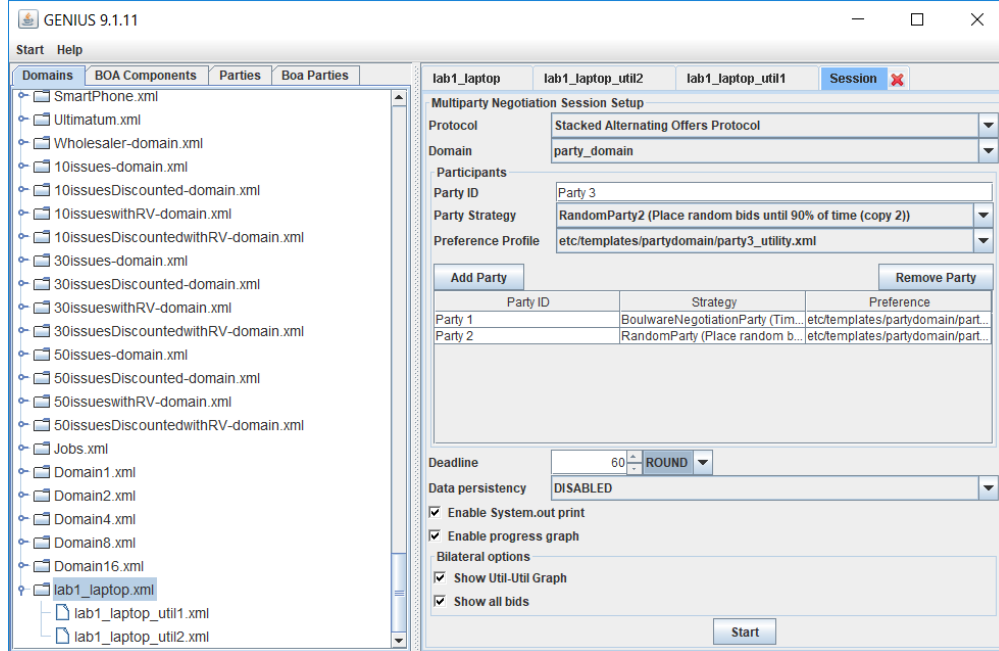


Figure 9: *Running a Negotiation Session*

Rounds vs Time

Note that the Deadline can either be set to ROUND, which means that the number refers to the maximum number of negotiation rounds. A round is when each agent has been able to make a (counter) offer. Alternatively, if the Deadline is set to TIME instead, then the value indicates the negotiation period in terms of number of seconds. This means that the number of negotiation rounds depend on the execution speeds of the negotiation agents.

Multi-Party Negotiation

Note that it is possible to have more than 2 parties in the negotiation. The SAOP protocol is designed such that it can handle multi-party negotiations. However, for the assignment we will only consider bilateral negotiations.

Choose either a time-based or round-based deadline and press Start.

At this points the agents will exchange offers and counter offers which are visualised in the negotiation progress graph. In addition, you will be able to see the negotiation log (you might need to move the divider across to see it). The results should look something like this:

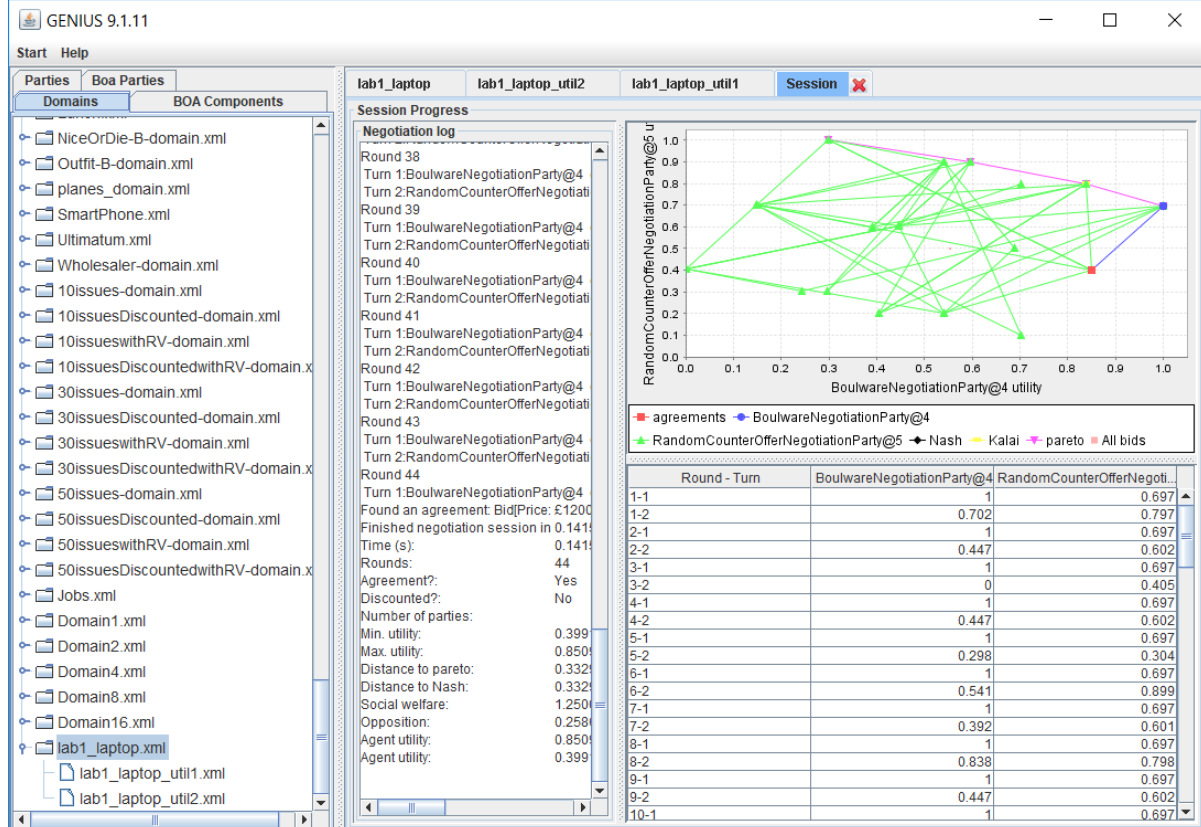


Figure 10: Results of the Negotiation Session

Note that an agreement is likely to be reached in this setting. The exact agreement will depend on the negotiation strategies chosen.

Now try the interface with other strategies and domains. Make sure you understand the interface and the negotiation log. We will cover the various evaluation measures in later labs. Also, more information can be found in the coursework assignment and the GENIUS user guide. Ask the demonstrators if you have any questions.

Note that, in addition to the negotiation log viewed in the GENIUS interface, this log can be found in the directory:

`{WORKING_DIRECTORY}/genius{version}/etc/templates/lab1_laptop`

END OF LAB 1