ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Assignment 3 – Coordination and Utility

Group 19

Abdullah Abdullah
Aksel Uhr
24/11-2021

Coordination and Utility

This assignment consists of two tasks where the first task is based on solving the coordination problem by arranging N Queens on the chessboard in such a way that they cannot attack each other and the second task is based on having some utility function as a deciding factor for the event guests to decide which stage they should move to according to their preferences.

The task 1 N Queens problem is actually related to a festival. The speakers in a festival are considered as Queens because the noise emission of speakers in a festival is the same as the queens movement in a chess. If the speakers are not placed at their correct positions it can cause weird sound problems which will be a bad experience for the festival guests.

The task 2 visit high utility stage is also related to a festival. The main aim of the task is to move guests according to their preferred attributes of a stage. The attributes of a stage can be better sound, light, area and other factors. Each stage has different values for these attributes which range between 0.1 and 1. The guests have their own preferences for each of these attributes like for one guest they may prefer low sound so a value of 0.2 maybe while for another guest it can be a 0.9 which means they love high sounds. So the task is to have a utility calculation function which can calculate the highest utility value for a guest and they inform the guest about that particular stage with highest utility value so that they can move towards the stage and enjoy the festival.

How to run
Run GAMA 1.8, open the folder attached to the submission, and run the experiments.

Species Task 1

Agent Queen, Shape = Green Sphere

This agent represents the queen or the speaker in the festival which will be arranged on the chessboard in such a way that no two queens will be able to attack each other. For each n queen we have a predecessor and successor queens, a position, target point, two booleans to see if they should move or not and a hashmap of <int, bool> to see if a queen has visited a point or not in the board. Each queen communicates with its predecessor using the FIPA request protocol. This is done by each queen, asking its predecessor if it can attack at the current position (safe or not). If yes, the current queen must change position, and ask its predecessor again. If not, it will proceed with asking the predecessor’s predecessors via each predecessor if it is safe at the current position. In case the requesting queen is in a safe position (all predecessors acknowledge the requesting queen that it is safe), it may stay at that position, and if not, it must change the position and try again.

Agent Chessboard, Shape = Black or White Square

The purpose of this agent is to print a chess board. For each even index in the board, a black square is drawn, and for each odd index in the board, a white square is drawn.

Species Task 2

Agent EventStage, Shape = Square (color = different colors for different stages)

This agent represents the event stages of the festival where a concert or another kind of activity might be happening which can be attended by the guests of the festival. Each stage has a hashmap of attributes along with their values which ranges between 0.1 and 1.0 and are randomly assigned as it was a requirement of the assignment. The event stage agent also has some variables like concert duration and restart concert which are just counters used for ending an act at a stage and then restarting it. When a new act at a particular stage is started the attributes values are randomized each time to have dynamic attributes values for the stages. In event stage agents there are only three functions, one to start the act at a stage, second to end the act at the stage and restart it after some duration and third to answer the requests from guests about the stage attributes or location of the stage after the guest has decided that it wants to move to a particular stage.

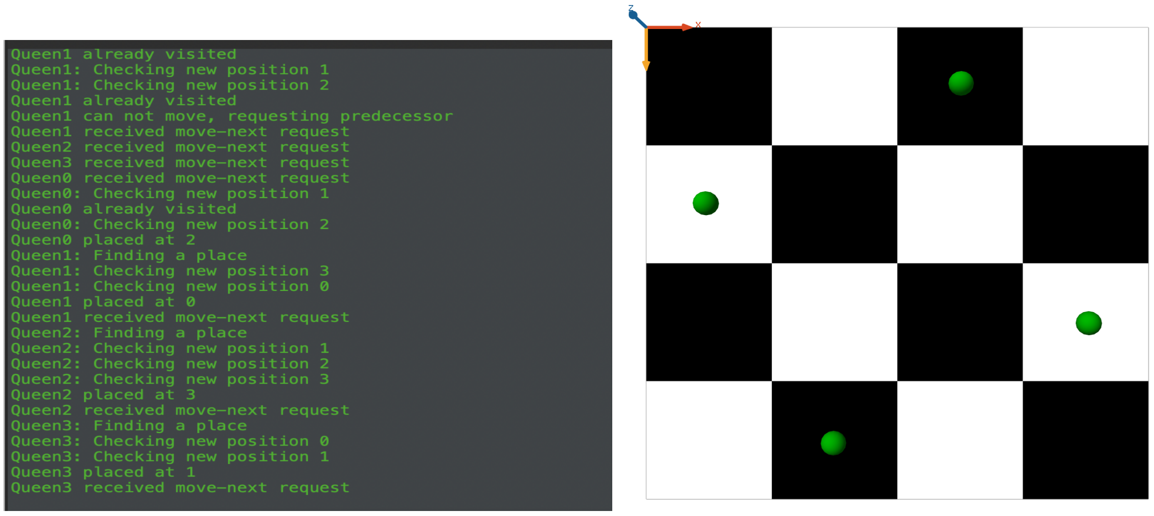
Agent EventGuest, Shape = Sphere (color green, changes to the stage color where it want to move)

This agent represents the guests of the festival, who wanders initially in the festival and at the same time asks for attributes values from the different stages agents in the festival. The guest agents have a hashmap of their preferences with values ranging between 0.1 and 1.0 randomly. These values are initiated once for a guest and they remain the same throughout the festival as a requirement of the assignment. The guests initiate the conversation using request performative and asks for attributes from the stage. The stage agent responds with a hashmap of stage attributes along with values. The guest agent takes the attributes values received from the stages and calculates the utility using the calcUtility function and stores it as a utility variable. This utility variable is then compared with guest current utility and if it's greater than guest current utility value we update the current utility and this process carries on until we get the highest utility value for a guest so that the guest can move towards that stage and enjoy the festival. After finding the highest utility value, the guest asks for the location of the stage and the stage responds with the location and guests updates the location for the stage and moves towards it.

calcUtility Function
The calcUtility function takes stage attributes hashmap as argument and loops over the keys of the hashmap which are basically the attributes names as strings and for each value of the key in hashmap it is multiplied with value of key in guest preferences and then added to a utility variable which is returned at the end when loop over the whole hashmap ends.

Results for Task 1

In the example below, one can see that no queens are interfering with each other, i.e. are able to attack.

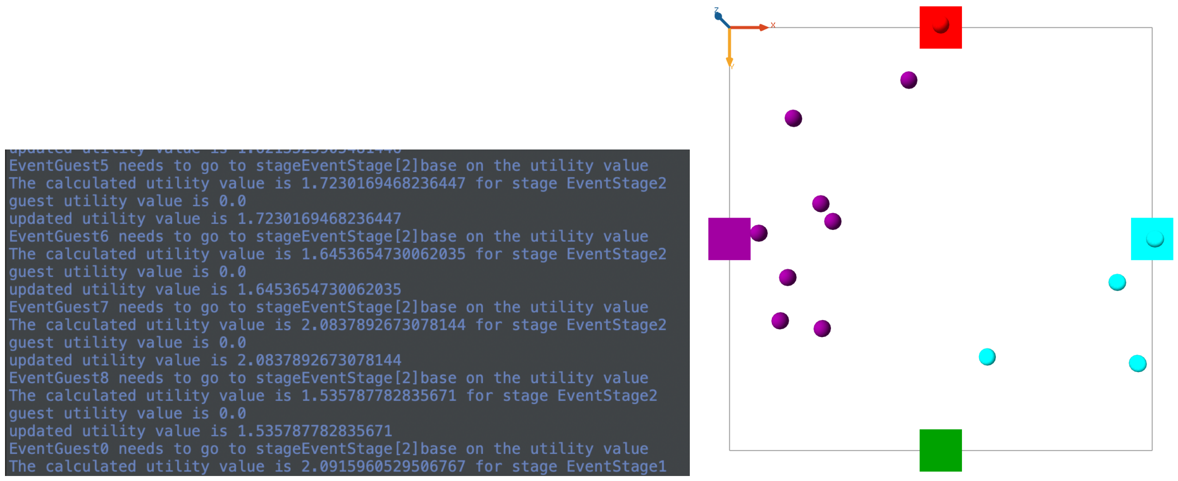


```
Queen1 already visited
Queen1: Checking new position 1
Queen1: Checking new position 2
Queen1 already visited
Queen1 can not move, requesting predecessor
Queen1 received move-next request
Queen2 received move-next request
Queen3 received move-next request
Queen0 received move-next request
Queen0: Checking new position 1
Queen0 already visited
Queen0: Checking new position 2
Queen0 placed at 2
Queen1: Finding a place
Queen1: Checking new position 3
Queen1: Checking new position 0
Queen1 placed at 0
Queen1 received move-next request
Queen2: Finding a place
Queen2: Checking new position 1
Queen2: Checking new position 2
Queen2: Checking new position 3
Queen2 placed at 3
Queen2 received move-next request
Queen3: Finding a place
Queen3: Checking new position 0
Queen3: Checking new position 1
Queen3 placed at 1
Queen3 received move-next request
```

Figure 1: Snapshot of task 1, n queen problem.

Results for Task 2

The second task works properly, as depicted below. From the snapshot below, we can see that the event guests calculate the utility value for each stage and on the basis of highest values then moves towards that stage.



```
updated utility value is 1.62155[illegible]
EventGuest5 needs to go to stageEventStage[2]base on the utility value
The calculated utility value is 1.7230169468236447 for stage EventStage2
guest utility value is 0.0
updated utility value is 1.7230169468236447
EventGuest6 needs to go to stageEventStage[2]base on the utility value
The calculated utility value is 1.6453654730062035 for stage EventStage2
guest utility value is 0.0
updated utility value is 1.6453654730062035
EventGuest7 needs to go to stageEventStage[2]base on the utility value
The calculated utility value is 2.0837892673078144 for stage EventStage2
guest utility value is 0.0
updated utility value is 2.0837892673078144
EventGuest8 needs to go to stageEventStage[2]base on the utility value
The calculated utility value is 1.535787782835671 for stage EventStage2
guest utility value is 0.0
updated utility value is 1.535787782835671
EventGuest0 needs to go to stageEventStage[2]base on the utility value
The calculated utility value is 2.0915960529506767 for stage EventStage1
```

Figure 2: Snapshot of task 2 visit high utility stage.

Discussion / Conclusion / Reflection

The n queen problem was very tricky (logic and coordination) compared to the previous assignments. It was somewhat similar to the distributed system course assignments, with the major difference being that we had to implement the algorithm from scratch. The complexity was quite enjoyable until a point where we felt that it overrode the association to the distributed AI theory and hence relevance to the course. More specifically, we spent much time “debugging” with the ‘write’ command and solving index problems. However, it was quite enjoyable to draw parallels to the distributed system course, and we ultimately acquired a better understanding of coordination and utility.