ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Assignment 1 – GAMA and Agents

Course Coordinator

Mihhail Matskin

Course Assistants

Mehdi Satei

Óttar Guðmundsson

Shatha Jaradat

Group 21

Khushdeep Singh Beant Singh Mann

Ketan Sunil Motlag

DATE.OF.ASSIGNMENT

29 Novemeber 2018

Agent Communication and Coordination

In this assignment, we were tasked to solve the famous NQueen problem via agent communication and coordination. For a NxN matrix there will be queens placed in such a way that no queen can be present in the same row & column or even on the diagonal. We were also tasked to create an event scenario in which there will be different events going on in different stages. An agent will pickup the location based on the calculated utility.

How to run

Run GAMA 1.7/1.8 and import filename DAIIA_Group_21_Assignment3_Nqueen as a new project. Press main to run the simulation. Note that changing parameters 1,2 and 3 will affect how A, B and C will work. Unzip DAIIA_Group_21_Assignment3_Nqueen.zip and import the resulting directory in GAMA 1.8 as a new project.

Navigate to creativity Model.gaml and press on the experiment button above (guests assignment) to run a simulation of the basic assignment, of the challenges. There are many parameters that can be tweaked to modify the behaviour of the simulation, and that will be explained in the next sections. Some examples are the number of participants in the festival, the number of places that offer drinks and the number of those that offer food.

Species

Task1

Queen

In the NQueen problem we have only species which has a skill FIPA. Each Queen is communicating with its predecessor. It has reflexes where it receives the position based on the position of previous queen. If it doesn't find a position because of the constraints it will request its predecessor to move so as it can find the position based on the constraints.

Task 2

Participants

Participant is an agent which will be taking part in the different events based on the several attributes of its own. It decides based on the utility which is simply calculated by multiplying the utility of this agent with the utility offered by particular event.

Stages

Stages is a specie which has different attributes which are informed to the participant. Based on the information provided by stages the participant will decide whether to join the event or not.

Leader

This specie helps in regulating the crowd by informing each participant about the crowded event in the festival. Depending on the information provided by the leader participant will change its decision to go to crowded place or not depending upon its preference.

Implementation

Task1

In task 1 first we implemented the constraints for the queens. The queens will move in the available matrix. If it's not able to find the location while meeting the constraints it will send a request to the previous queen and it will keep on iterating the same until the condition is satisfied. Every time the available locations list is updated based on the position of the queen.

Task2

In task 2 we started by creating the stages with random utilities. Each participant will go to the event based on the total utility based on six different attributes. Each agent will have different values of utilities for different attributes. We displayed stage crowd count in console for every event and based on the number of people, crowd is manipulated by the special agent called as leader.

Results

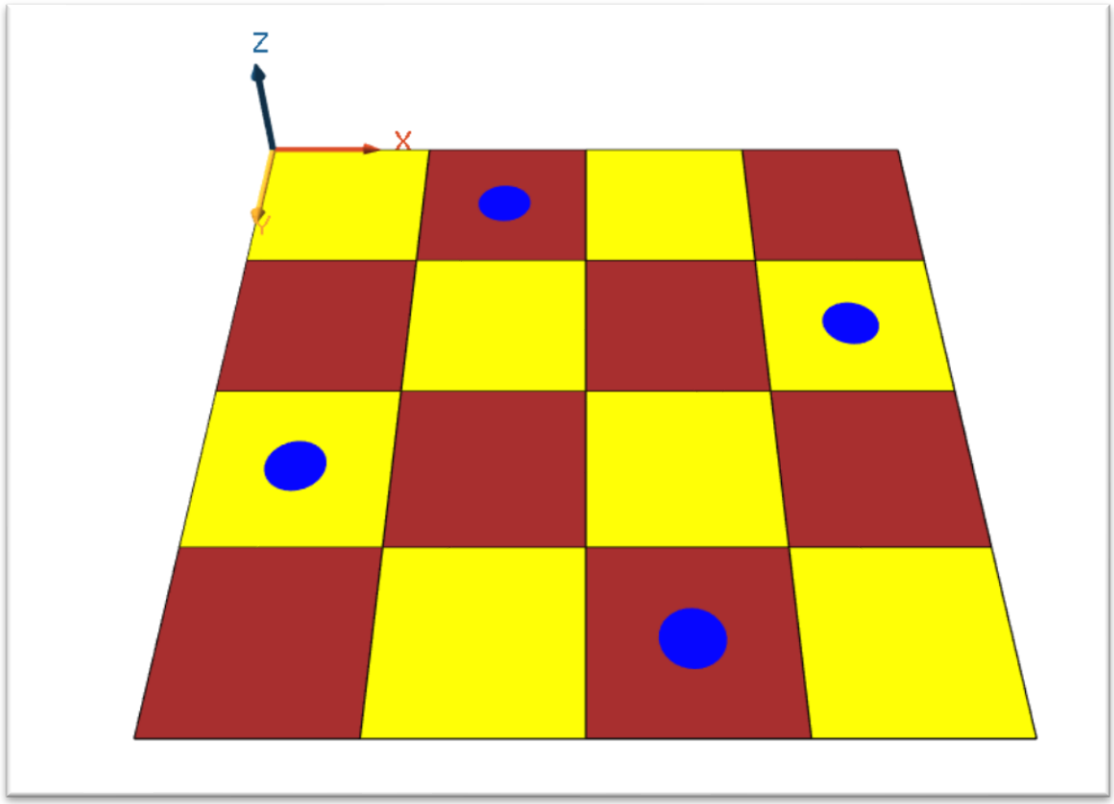
Task1

The communication between each queen is displayed on the console as shown below

```
llowed positions are  [0,1,2,3]
ndex is 0
ueen list is [queen(0)]
ueen creation no. 1
revious queen was queen(0)
ew queen added is [queen(1)]
 now nqueens list contains [queen(0),queen(1)]
oing to target position 0
k-positions contains [0,1,2,3]
. agree request message is message[sender: queen1; receivers: [queen0]; performative: request; content: [positions]; content]
. inform request message contains message[sender: queen1; receivers: [queen0]; performative: request; content: [positions]; content]
eceivers message is message[sender: queen0; receivers: [queen1]; performative: inform; content: [[0], [0], [0]]; content]
eceivers ok_position is [2]
eceivers ok_position is [2,3]
oing to target position 2
ueen creation no. 2
revious queen was queen(1)
ew queen added is [queen(2)]
 now nqueens list contains [queen(0),queen(1),queen(2)]
oing to target position 0
oing to target position 2
k-positions contains [2,3]
. agree request message is message[sender: queen2; receivers: [queen1]; performative: request; content: [positions]; content]
. inform request message contains message[sender: queen2; receivers: [queen1]; performative: request; content: [positions]; content]
eceivers message is message[sender: queen1; receivers: [queen2]; performative: inform; content: [[0, 2], [0, 1], [0, 3]]; content]
k-position is zero
oing to target position 0
oing to target position 2
k-positions contains [2,3]
ive an new position now
. agree request message is message[sender: queen2; receivers: [queen1]; performative: request; content: [newPositions]; content]
ow ok-positions become [3]
. inform request message is message[sender: queen2; receivers: [queen1]; performative: request; content: [newPositions]; content]
eceivers message is message[sender: queen1; receivers: [queen2]; performative: inform; content: [[0, 3], [0, 2], [0, 4]]; content]
```

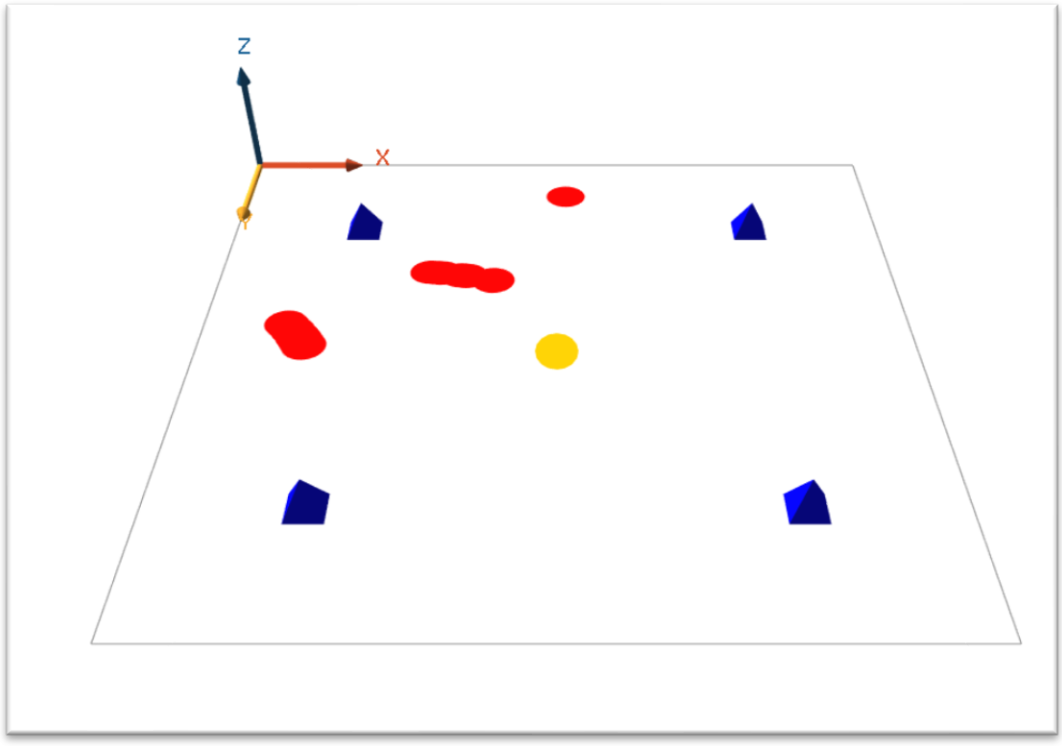
As shown every time its being checked by queen whether it fits all the constraints or not and it chooses the position in the matrix based on that. If there is not available position for the queen in such way that it can fit in any matrix while fulfilling the conditions it will simply communicate with the previous queen and pas the request to it for changing the position. This is done until the latest queen can take one of the positions from the available one.

One of the solutions for 4x4 Matrix is shown below where the queens are placed as per constraints



Task 2

In task 2 agents are going to different events based on the utility function calculated and the priority of each agent. For example, if few of the agent is interested in place where there is less crowd it will go to the respective location.



Discussion / Conclusion

Solving the N-queen problem was very interesting as we have to understand the concept of iterations in the backpropagation. It was possible for us to create the solution for different values of N. In the festival implementing the leader concept was very interesting although there could have been different for implementing the same concept of crowd without having to actually communicate between leader and all other agents.