ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Assignment 3 – Coordination & Utility

Group 25

Yunfan Yang

Pei Lun Hsu

27.11.2019

In this assignment, we were asked to implement 2 tasks, one is to make different agents coordinate with each other to solve a positioning problem. Another task is to implement the utility function to determine the behaviors of the agents.The purpose of the assignment is to let us be able to implement simple coordination model and utility function.

How to run

Run GAMA 1.8 and import lab3_basic.gaml/lab3_queens/lab3_challenge as a new project. Note that changing parameters nb_guest, nb_stage parameters in lab3_basic will affect the initial number of the agents guest, stage, respectively.

Task 1

For task 1, each queen will remember the queen in the last column and the next column. At each round, the queen will search all the cells on that column it is located and store all the available cells. After picking a random cell from the available cell list, it will calculate all the non-available cells and put them into the tabu cell list. The queen on the next column inherits the tabu list and implements those action above again. Once a queen finds there is no available cell at its column, it will communicate the previous queen to relocate its position based on its available cells list and recalculate the tabu list. If the previous cell also does not have any more available cells, it will again communicate its previous queen and ask it to do the same thing. This iteration will continue to go on until all the queens find their position.

Task 2

Species for task Utility

Agent stage

These agents are responsible to start the four different concerts, offer stage variable values to the guests using FIPA communication as concerts start.

Agent guest

These agents are responsible for deciding which stage to go based on the output of the utility function, which is unique for each agent.

Implementation

We first developed agent stage with **reflex send_variable** to start the concerts, randomly initialize the stage variables for every new concert, and send these stage variables, e.g., light, sound, crowd, to all agent guests. And then implement **reflex end_the_concert** to finish the concert after a certain time period.

We then developed agent **reflex decide_stage** to receive the FIPA cpfs from agent stages, calculate the utility based on the guest's preference parameters and the stage variables, which

are extracted from the cpfs from each stage, and then decide which stage to go according to the highest utility. Finally, reflex left_the_concert make guests left the concert and wait for next concerts start.

Results

As you can see in the following log for the process of deciding which stage to go:

(Time1.0): stage0 sends an cfp message to guests to provide the stage values of the upcoming concert
(Time1.0): stage1 sends an cfp message to guests to provide the stage values of the upcoming concert
(Time1.0): stage2 sends an cfp message to guests to provide the stage values of the upcoming concert
(Time1.0): stage3 sends an cfp message to guests to provide the stage values of the upcoming concert
(Time 2.0): guest0 will go to stage0, that has highest utility: 2.432326898071362
(Time 2.0): guest1 will go to stage3, that has highest utility: 1.3548679531823782
(Time 2.0): guest2 will go to stage2, that has highest utility: 1.8191703496119136
guest0 left the concert.
guest1 left the concert.
guest2 left the concert.

Challenge

The first guest that has been initialised will act as a leader to collect the decisions from other guests and return them feedbacks. The guests will pick their desired stages twice. After the first time they pick it, they will send the decision to the leader. The leader will count on the attendance of guests to each stage and then send back the information to the guests. Based on the information, guests who have a different tendency to extend of the crowd will recalculate their utilities for each stage and rechoose a best one with highest utility. To do this, global utility will be maximized.

Result

Overall the assignment is great. The logic to implement is straightforward and we like the way how the agents are constructed. The debug for the code is really interesting - to change the execution condition of different reflexes and see it working properly is great.