ID2209 - Distributed Artificial Intelligence and Intelligent Agents
Assignment 3 - Coordination and Utility

Kishore Kumar, kishorek@kth.se Abel Valko, valko@kth.se

November 2021

1 Introduction

The objective of this assignment was to solve the N-Queens problem and a utility maximizing problem using a multi-agent approach.

1.1 How To Run

1.2 N-Queens

Parameters that can be adjusted are:

- *numberOfQueens*
- *queen0Start* determines the placement of the first queen. Changing this results in different results.

1.3 Utility

Parameters that can be adjusted are:

- *numberOfAudience*
- *myPreferences* this is an attribute of the Audience which controls the preferences of each person from the audience.

2 Species for N-Queens

2.1 Queen

These are the queens that must be placed on teh board. They have the following actions:

- *initiate* initiates the queen. This is called only on the first queen at the start of the simulation. The queen is placed and then informs its successor that it is their turn to find a spot.
- *identifyConflict* identifies if there are any conflicts with other placed queens from the current position of the queen. It keeps track of this using a queen variable called *conflict*.
- *makeMove* Tries to place the queen in a position without conflicts. If it is successful it notifies its successor. If it can not find a position without conflict, or has received conflict messages from its successor for all available positions, it will inform its predecessor that there is a conflict and it must move.

and the reflex:

- *resolveRequest* receives inform messages from successors and predecessors and calls the makeMove action.

2.2 Board

This is a grid species that makes up the board with *numberOfQueens* times *numberOfQueens* cells.

3 Species for Utility

3.1 Stage

These are the Stages to which the Audiences go to. They play different acts from time-to-time, with different attributes. They have the following reflexes:

- $change_attributesThiskeepschangingtheattributesoftheactcalledasactAttributeevery50timesteps.Theattributesarechose$

3.2 Audience

These are the guest agents which move to the Stages based on their preferences, *myPreferences*. They have the following reflexes:

- *requestActValues* This requests the Stages for their act attributes every 50 time steps using the inform protocol.
- *goToTarget* this reflex is activates whenever the Audience agent has a value in targetLocation which holds the location of the desired stage and the position of the agent is not equal to the target's position. It commands the agent to go towards the target.
- *reachedTarget* this reflex is activates whenever the Audience agent has a value in targetLocation and the position of the agent matches with the target's position. It resets the targets and the *utilityValues* calculated by the agent for each stage.
- *receiveAndSetGoal* this reflex is activates whenever the Audience agent receives messages from the Stages using the inform protocol. It receives the *actAttribute* from each Stage and calculates the *utilityValues* for each stage. It finds the stage with the maximum *utilityValue* and sets the *targetLocation* and the *targetStage* values to reach.

4 Results

Run code for full simulation and printout results. Below are results for 4, 8, and 12 queens.

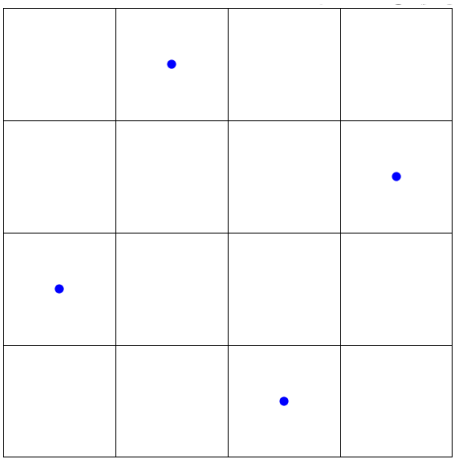

Figure 1: One example of a solution for the 4 queens problem.

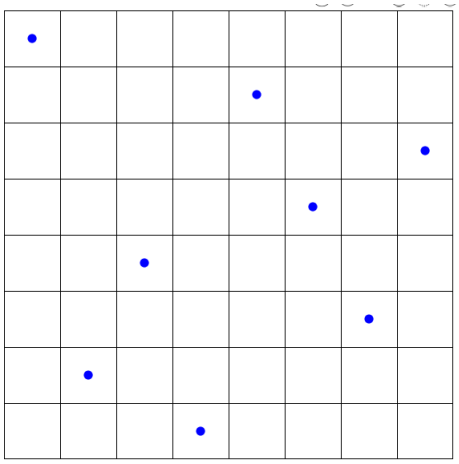

Figure 2: One example of a solution for the 8 queens problem.

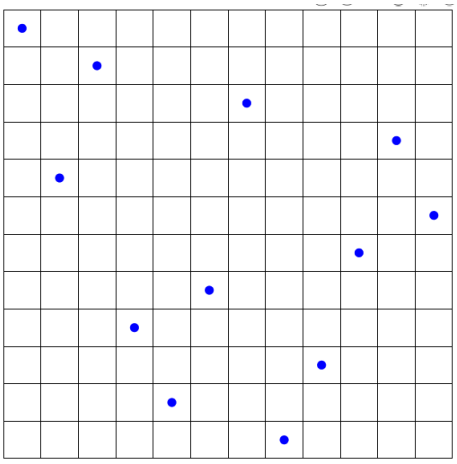

Figure 3: One example of a solution for the 12 queens problem.

Run code for full simulation and printout results. Below are results for the utility simulation.

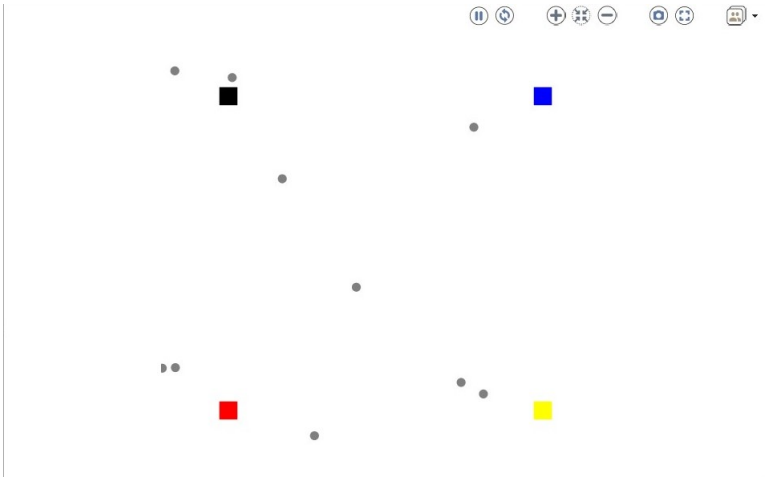

Figure 4: The simulation begins with the audiences yet to find their desired stages.

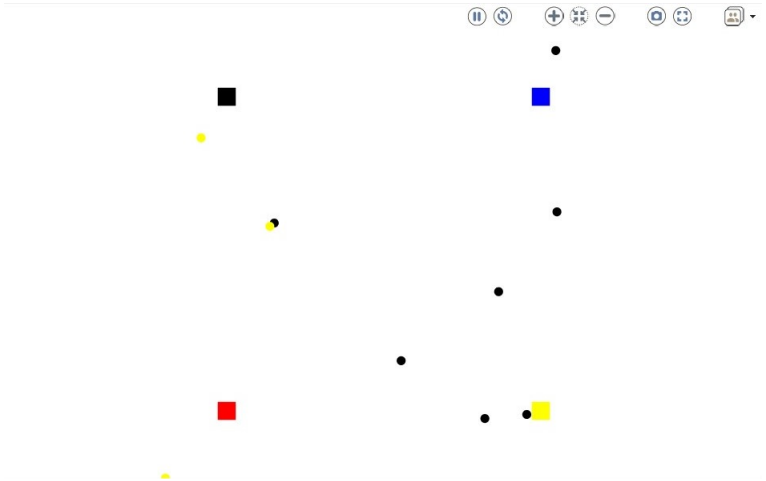

Figure 5: The audiences after finding their desired targets.

```
Stage0: Flipping attributes for new act
Stage0: Attributes for new act[0.6,0.1,0.0,0.5,0.2,0.3]
Stage1: Flipping attributes for new act
Stage1: Attributes for new act[0.9,0.1,0.0,1.0,0.7,1.0]
Stage2: Flipping attributes for new act
Stage2: Attributes for new act[1.0,0.1,0.6,0.4,0.2,0.2]
Stage3: Flipping attributes for new act
Stage3: Attributes for new act[0.3,0.2,0.7,0.5,0.0,0.2]
Audience0: Asking each stage for attributes of the new acts
Audience1: Asking each stage for attributes of the new acts
(Time 1.0): Stage0 received requests
    Stage0 receives an inform message from Audience0 with content ['Need Attributes']
    Stage0 receives an inform message from Audience1 with content ['Need Attributes']
(Time 1.0): Stage1 received requests
    Stage1 receives an inform message from Audience0 with content ['Need Attributes']
    Stage1 receives an inform message from Audience1 with content ['Need Attributes']
(Time 1.0): Stage2 received requests
    Stage2 receives an inform message from Audience0 with content ['Need Attributes']
    Stage2 receives an inform message from Audience1 with content ['Need Attributes']
(Time 1.0): Stage3 received requests
    Stage3 receives an inform message from Audience0 with content ['Need Attributes']
    Stage3 receives an inform message from Audience1 with content ['Need Attributes']
(Time 1.0): Audience0 receives inform messages
    Audience0 receives an inform message from Stage0 with content [0.6,0.1,0.0,0.5,0.2,0.3]
    Audience0 receives an inform message from Stage1 with content [0.9,0.1,0.0,1.0,0.7,1.0]
    Audience0 receives an inform message from Stage2 with content [1.0,0.1,0.6,0.4,0.2,0.2]
    Audience0 receives an inform message from Stage3 with content [0.3,0.2,0.7,0.5,0.0,0.2]
(Time 1.0): Audience0 Utility values are:map(['Stage0'::0.4649999999999997,'Stage1'::1.25,'Stage2'::0.6200000000000001,'Stage3'::0.48500000000000004]) Max value is:1.25
Audience0: Target found!:Stage1 Location:{85.0,15.0,0.0}
(Time 1.0): Audience1 receives inform messages
    Audience1 receives an inform message from Stage0 with content [0.6,0.1,0.0,0.5,0.2,0.3]
    Audience1 receives an inform message from Stage1 with content [0.9,0.1,0.0,1.0,0.7,1.0]
    Audience1 receives an inform message from Stage2 with content [1.0,0.1,0.6,0.4,0.2,0.2]
    Audience1 receives an inform message from Stage3 with content [0.3,0.2,0.7,0.5,0.0,0.2]
(Time 1.0): Audience1 Utility values are:map(['Stage0'::0.6399999999999999,'Stage1'::1.235,'Stage2'::1.0,'Stage3'::0.52]) Max value is:1.235
Audience1: Target found!:Stage1 Location:{85.0,15.0,0.0}
```

Figure 6: The console output while executing the script

5 Discussion

The N-Queens assignment was interesting but unfortunately most of the time was spent solving the problem in code and finding a functioning algorithm as opposed to understanding coordination itself.

The Utility assignment was comparatively easier than the N-Queens problem. However the bonus challenge was a lot more demanding due to the dynamic nature of the problem with the stage attributes varying with the crowd which further influences the crowd's utility values.