Assignment 3 - Distributed AI & Intelligent Agents

Perttu Jääskeläinen perttuj@kth.se
Group 1
Gabriele Morello morello@kth.se

November 30, 2022

Contents

1 Introduction

In this homework, we worked more with agent coordination and communication, with the goal to achieve greater knowledge in FIPA and agents in GAMA. This was done partially in isolation, and partially by extending the implementation from previous assignments

This homework includes solving the NxN queen problem for certain board sizes (up to 20x20), and introducing preferred attributes for festival guests, which determine which stage they will attend at the festival.

As an extra challenge, we also solved the problem of global utility - ensuring the best global scenario for all guests, which also takes into account guests preferences for avoiding or preferring crowds.

2 Basic Implementation

In the basic implementation, we implemented the N x N queen problem in a separate file,**NQueens.gaml**, and the festival guest utility in **FestivalStages.gaml**.

2.1 N x N Queen Problem

The strategy for achieving the goal of the NxN queen problem works the following way:

- Queens will receive their position from their predecessor. The first queen will determine it's own position on the first row.
- Each queen will be assigned to it's own row - i.e. queen 1 will be on row 1, queen 2 on row 2 etc. No queen will be placed outside it's own row.
- When assigned a position, the queen will check if the **diagonal**, **row** and **column** are safe (i.e. no queen is present on them).
- If the position is not safe, the queen will ask it's predecessor for a new position. If it is safe, the queen will position it's successor.
- If a queen cannot position it's successor to more squares, the queen will itself ask its predecessor for a new position.
- When the last queen is positioned, we print out the formation in the log - and continue looking for more positions.
- If no more positions are available (we arrive back at the first queen, who is on the last column), an error is thrown

We managed to get this working by reducing the set of squares that we attempt to place queens on as much as possible, which greatly reduces the computations we have to make. We managed to get this working for a 20x20 board with 20 queens at around 29k cycles in GAMA; and the first formation can be seen in figure 1. If the simulation keeps running, we will print more and more formations as the simulation continues.

2.2 Guest Utility - Preferred Stages

As a new concept, we introduce the **Stage** species in the simulation. Each **FestivalGuest** will attend some **Stage** depending on how well it fits their preferences. To determine this, we introduced utility for each guest - which will describe their preference at the festival. The **Stages** also have these same attributes. These include the following: **band**, **dancers**, **light show**, **sound**, **stage presence** and **vfx**- utility.

Each stage will have an attribute for how much they've spent on each of these attributes, and guests will attend stages that match their preferences the best. Below is a pseudo code example of how this is calculated, assuming stages only have two attributes (in reality, all six are taken into consideration):

- stage1Utility <- guest.bandUtility * stage1.bandUtility
- stage2Utility <- guest.dancerUtility * stage2.dancerUtility
- if stageUtility1 >= stage2Utility then: attend stage1 else: attend stage2

Stages will occasionally rotate artists or concerts, in which case they will receive new values for their attributes - this will causes guests to re-calculate their preferences and change stages if necessary. In total, there are 4 stages at the festival, and stages will inform guests of changes using informs in the FIPA protocol. If the stage the guest is currently attending changes performers, they will re-evaluate which stage to attend. If it is some other stage that does the rotation, the guest will compare the performer at their current stage to the new stage - and switch if the utility is greater.

3 Challenges

The challenge was done in the **GlobalUtility.gaml** file.

3.1 Global Utility

For this bonus task we introduced a new element: the crowd mass, this value is a global utility that agents have to consider when they choose where to go because some agents will prefer to go to a crowded stage while others prefer less populated ones. We tackled this problem using a leader actor that collects all the information from each agent and then decides where each agent has to go. Guests send to the leader a vector with all their preferences for each stage, the leader collects them, computes the total global utility, and sends to each guest a vector with the densities for each stage. The guests will now update their preferences and communicate their updates to the leader. This process will be repeated a definite number of times after which the leader picks the case where he computed the highest global utility and tells each guest to go to the corresponding stage, this case will not be optimal for each guest therefore some of them will have to sacrifice their own utility.

4 Discussion

Figuring out how to solve the NxN queen problem proved to be problematic, since the simulation was taking way longer than expected. As an initial solution, we placed each queen on the (0,0) coordinate, and naively attempted to position them on any square on the board. This of course turned out to be very inefficient, since we *had to place each queen on its own row* - so attempting to place them on some other row was completely unnecessary. We also ended up in scenarios where we were testing all permutations of queen placements - so we had to reduce the set of squares that we place queens on.

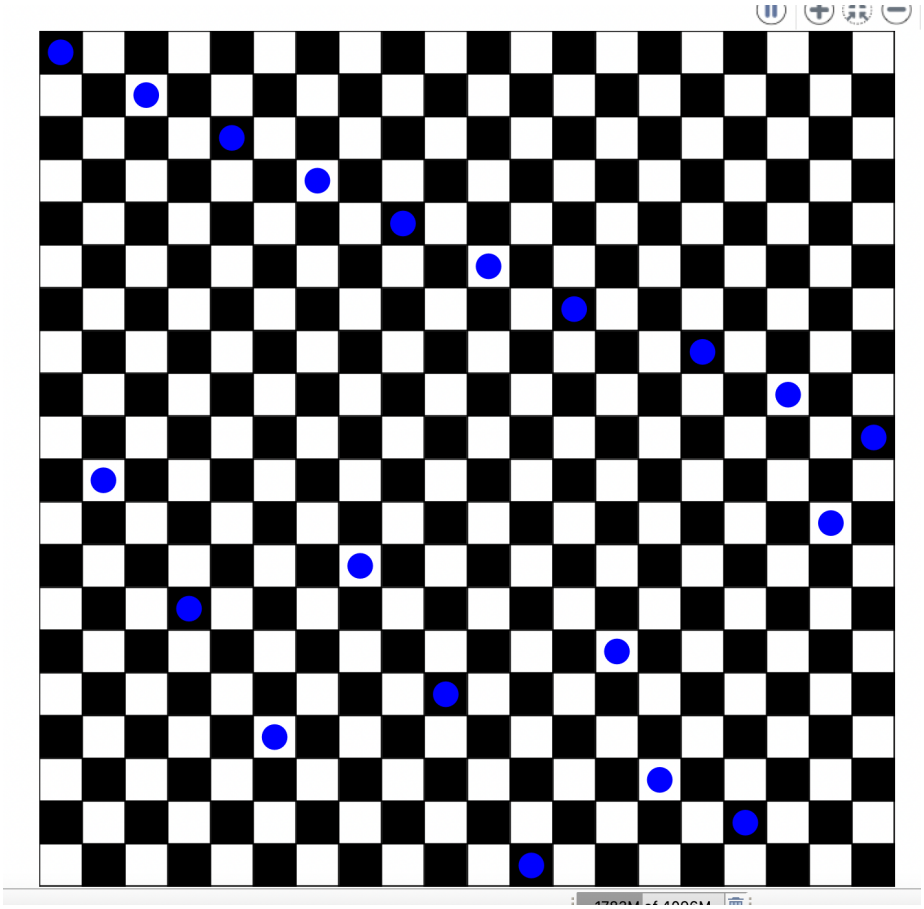

Figure 1: *The first queen formation found for a 20x20 board at around 29k cycles.*

When we figured out that we should simply place each queen on it's own row - and if we run out of columns, we just propagate to the previous row - it was fairly straight forward to implement, and we got the result much quicker.

The utility for guests was very straight forward to implement - since we were already familiar with agent communication and how to move toward locations in GAMA. However, the challenge portion proved to be more difficult than anticipated.

The difficulty with the crowd mass was to figure out how its supposed to work - making sure it eventually converges. We solved this by setting a limit to negotiation rounds - set to 5 - which seemed to be enough for 21 agents. Usually they came to an angreement after 2-3 rounds, sometimes requiring a bit more - and we could easily see the impact of the negotiation in the end result.

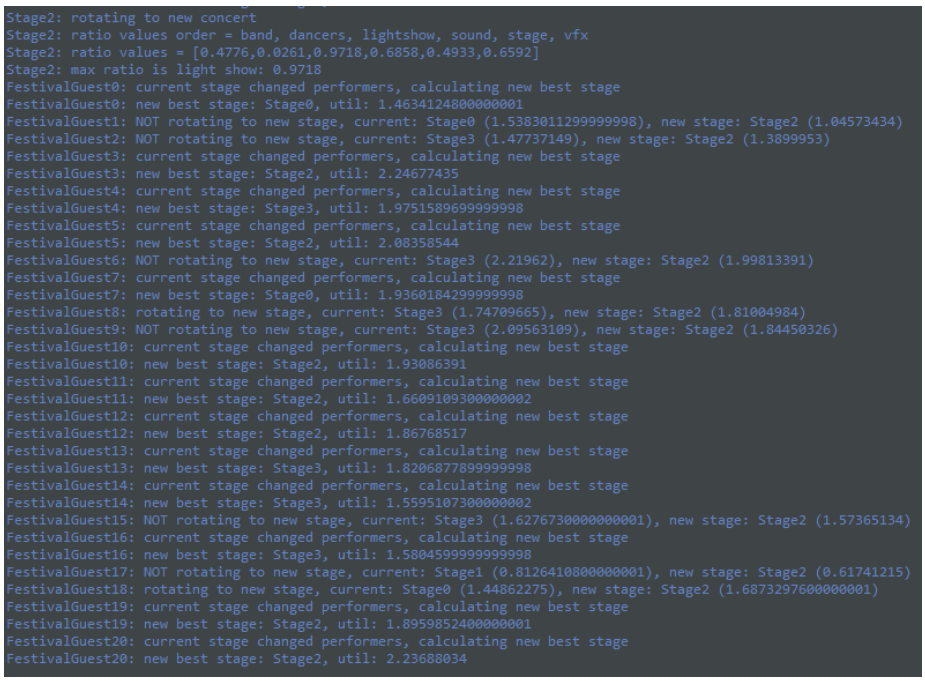

Figure 2: *Festival utility for guests when stages are rotated.*

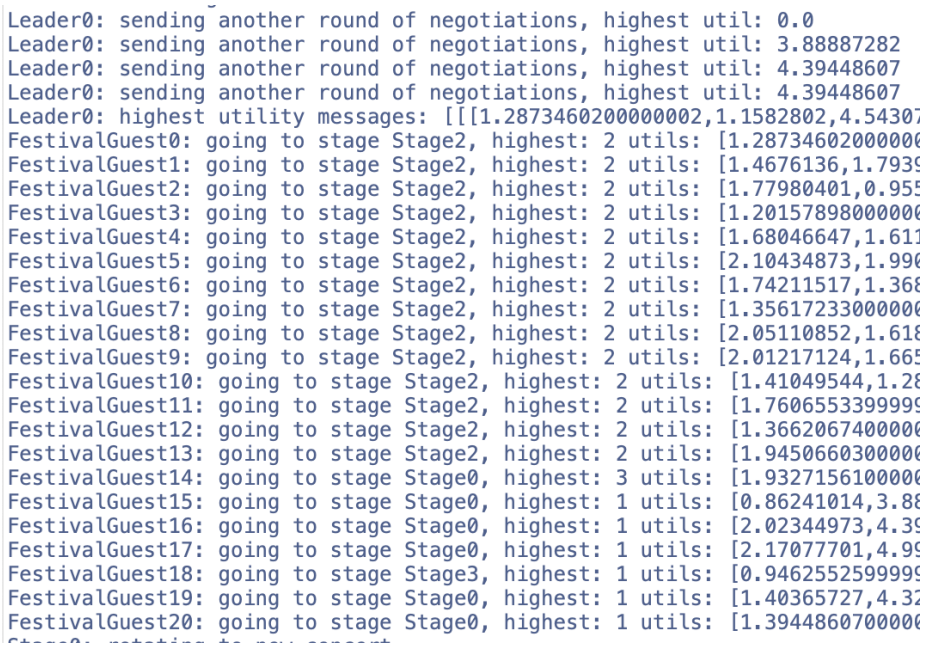

Figure 3: *Global utility coordination between agents - here we can see that agents 18, 19 and 20 go to stages that are different from their highest utilities, in order to prioritize the global utility.*