Imperial College London

Final Report

Self-Organising Multi-Agent Systems

Department of Electrical and Electronic Engineering
Imperial College London

SOMAS Class 2021-2022

January 2022

Lecturer

Prof. Jeremy Pitt

Abstract

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Contents

1 Introduction **1**

1.1 Problem Statement 1

1.2 Our Goals 1

2 Organisation Structure **2**

3 Simulation Structure **3**

3.1 Simulation Environment 3

3.2 Simulation Flow 3

3.3 Message Passing 3

3.4 Health Decay System 3

4 Visualisation **4**

5 Team 2 Agent Design **5**

6 Team 3 Agent Design **6**

6.1 The Agent 6

6.1.1 Agent knowledge 6

6.1.2 Agent decisions 7

6.1.3 Agent variables 7

6.1.4 Agent generation 8

6.1.5 Agent Strategy 8

6.2 Agents emotions 8

6.2.1 Read function 8

6.2.2 Friendship Function 8

6.2.3 Mood, Morality and Stubbornness helper functions 9

6.2.4 Changes to environment 9

7 Team 4 Agent Design 11

8 Team 5 Agent Design 12

9 Team 6 Agent Design 13

10 Team 7 Agent Design 14

11 Experiments 15

12 Results 16

13 Discussion 17

14 Conclusion 18

15 Future Work 19

A Appendix 20

Bibliography 21

Chapter 1

Introduction

1.1 Problem Statement

Section on the tower and how we have interpreted the spec. Should include a basic spec for features that we wanted to achieve

1.2 Our Goals

Getting a simulation environment working

Chapter 2

Organisation Structure

Could introduce each of the teams and a one liner about their agent strategy. This will obviously be expanded upon in the agent sections. Should contain an org chart and describe important parts of it. Justify certain organisational choices. Maybe consider what could have been done better?

Chapter 3

Simulation Structure

3.1 Simulation Environment

Goes over the backend design

3.2 Simulation Flow

The order in which events happen Need a nice diagram for this

3.3 Message Passing

Need a nice diagram for this

3.4 Health Decay System

Need a nice diagram for this

Chapter 4

Visualisation

Important things about front end.

Chapter 5

Team 2 Agent Design

Chapter 6

Team 3 Agent Design

6.1 The Agent

The agent is designed to show different combinations of behaviour depending on three variables we have used to define any agent: stubbornness, morality and mood. These three variables, defined in a range of 0 to 100 will change as the agent encounters a series of scenarios, with its personality shifting as events unravel. The value of the different variables will affect how the agent takes decisions and acts, all of which will be explained in the following pages.

As the main function from which all the functionality of the agent is bases we have the `func (a *CustomAgent3) Run()`. The agent follows then next steps:

1. The agent starts by updating its personality variables at the beginning of each day calling the necessary function.
2. If the agent was just 'born' it will learn an initial value for the average food consumed.
3. In the case the agent has changed floors the personality variables are changed depending on the characteristics of the change.
4. In the case the platform is not empty the agent will eat, the quantity depending on `a.takeFoodCalculation()`.
5. Finally the agent will read and send any messages it considers.

6.1.1 Agent knowledge

Agent 3 will be able to remember facts in order to make decisions. The variables in their memory are:

1. `floors []int`: stores the floors the agent has been in before. This information is used on reshuffles to set our mood.
2. `lastHp int`: stores the last recorded HP value. Currently used to check a new day has started.
3. `friends map[uuid.UUID]float64`: the agent is aware of the people they have met during their time in the tower. We assigned the agents uuid with our affection for them. Affection range is 0-1 (0 = dislike, 1 = like).

4. `foodLastEaten food.FoodType`: stores food last eaten.
5. `foodLastSeen food.FoodType`: stores food last seen in the platform
6. `foodMovingAvg float64`: stores moving average of food consumed
7. `agentAge int`: stores how old (in days) the agent is.
8. `treatyProposed messages.Treaty`: stores whether we already have a treaty with the person above
9. `reshuffleEst int`: stores our estimation of reshuffle
10. `hpAbove int`: stores the HP our neighbour above last told us they had
11. `hpBelow int`: stores the HP our neighbour below last told us they had

6.1.2 Agent decisions

Agent 3 will be able to take decisions when receiving messages or after signing treaties and store these until they eat. This information is used to symbolize a predetermined decision has been taken instead of "impulsively" eating food depending solely on our hunger and state of mind.

1. `foodToEat int`: stores how much food we have decided to eat in the next eating period.
2. `foodToLeave int`: how much food we have decided to leave. It is necessary in case we want to: E.g eat 6 food and leave at least 10 food but when the platform arrives we see that it has 13 food. Both statements cannot be fulfilled and so we must prioritise one.

6.1.3 Agent variables

Agent 3 will act differently depending on three different variables that define them, which will give the agent different personalities.

1. `stubbornness int`: defines the likelihood of the agent to read a message. E.g: Stubbornness = 20 means there is a 20% chance that the agent will ignore the message.
2. `morality int`: defines the willingness of the agent to help others, in other words, how much you care about other agents in the tower.
3. `mood int`: defines how likely the agent is to do things, it will affect its decision-making process.

For the purpose of this agent, we will define personality as the individual differences in characteristic patterns of thinking, feeling, and behaving, as defined by the American Psychology Association. The traits that define an agent's personality are mutable, making our agent adapt its personality depending on the circumstances.

6.1.4 Agent generation

The function generates a new agent by initializing its variables, knowledge and decisions. All knowledge and decisions are defined initially as empty except for our last HP, which is initialized as 100 (as all agents are).

The variables of our agent, which define the personality it has, are randomly allocated to generate different agents. All three variables are defined to be in the range 0 to 100. Therefore morality and mood are randomly allocated to a value between 0 and 100. To ensure our agent is not completely deaf from the start we limit the initial stubbornness to 75. This does not limit the agent once it starts interacting and taking decisions. Which means it could possibly end up with a stubbornness value of 100 if the situations it goes through are auspicious for it.

6.1.5 Agent Strategy

Our agent follows the following strategy depending on the three agent variables:

1. When morality is high, agents are more willing to listen and obey other agents.
2. When morality is low, agents are less willing to listen and choose to eat more than they would when they have high morality.
3. When stubbornness is high, agents will be "deaf" to most messages and not react to them.
4. When stubbornness is high, agents are more willing to listen to messages.
5. When mood is high, the actions taken are beneficial for more people.
6. When mood is low, the actions taken are more beneficial for ourselves.

6.2 Agents emotions

The following functions act as helpers for the rest of the agent code, enabling changes in the agents variables and reactions due to the value of these variables, as well as reactions to friendships.

6.2.1 Read function

`func (a *CustomAgent3) read() bool`: our agent's stubbornness will limit the amount of messages it might read and react to. Each time we receive a message we call the `read()` function. This function will take into account the current stubbornness of the agent and return if the message must be read, and reacted to, or just discarded. This function works by randomly generating a number between 0 and 100. If this number is higher to the current stubbornness the message is read, if not it will be discarded.

6.2.2 Friendship Function

`func (a *CustomAgent3) updateFriendship(friend uuid.UUID, change int)`: changes the value of the friendships using Q-Learning. Change is either -1 or 1.

In the case we encounter a new agent we record this new ‘friendship’ in our memory. From the start we have an initial opinion on the agent we have just met. As for now we just know about its existence, we predefine a friendship level depending on our current morality. To ensure we don’t create a huge predisposition of the agent to others we limit this range on initial impression to be between 0.4 and 0.6

6.2.3 Mood, Morality and Stubbornness helper functions

`func (a *CustomAgent3) changeInMood(pointsMin, pointsMax, direction int)`: Helper function to change the mood. There is an aspect of randomisation and the mood changes between pointsMin and pointsMax. Depending on the direction, 1 or -1, the mood will change in a positive or negative sense, increased or reduced.

`func (a *CustomAgent3) changeInMorality(pointsMin, pointsMax, direction int)`: same as before but changes morality.

`func (a *CustomAgent3) changeInStubbornness(change, direction int)`: same as before but changes stubbornness.

6.2.4 Changes to environment

Changes to the environment will change the variables our agent is defined by. We have defined two possible environmental changes our agent will be affected by:

1. A new day `func (a *CustomAgent3) changeNewDay()`: changes our morality and mood depending on our HP that morning. Changes depend on our current HP, which we have divided in four possible situations: 0-19, 20-39, 40-74 and 75-100.

2. A change in the floor we are located (reshuffle) `func (a *CustomAgent3) changeNewFloor()`: a change in the floor we are at will make the mood of our agent to change accordingly. We have designed for 4 possible scenarios to happen:

   (a) We arrive at the highest floor we have ever been to: we get a boost of moodas we should be able to eat more than any other time, our mood can change positively between 5 and 10 points.

   (b) We arrive at a higher floor than before, but not the highest we have ever been: We are higher than before which is positive for us. As we could have done better our mood increases but less than in the previous scenario, our mood can change positively between 0 and 5 points.

   (c) We have arrived at a lower floor than before, but we have been lower: Our situation is worse than before but it could have been even worse. We lose mood points, our mood can change negatively between 0 and 5 points.

   (d) We have arrived at the lowest floor ever: We have never been in a worse situation and our agent will suffer a blow from this change. Our mood can be specially affected in this scenario, our morale can change negatively between 5 and 15 points.

   Each scenario can affect our agent in a different way. To make the scenario as realistic as possible we have added an element of randomness to the change in mood, depending on the range given on each of the scenarios.

Inside the function we store our new floor in our knowledge, we restart the HP of our neighbors, we don't know who they are, and call `a.reshuffleEstimator()`.

`func (a *CustomAgent3) reshuffleEstimator()`: ensures we do not accept treaties longer than two times the reshuffle period so there are less chances a treaty kills us

Chapter 7

Team 4 Agent Design

Chapter 8

Team 5 Agent Design

Chapter 9

Team 6 Agent Design

Chapter 10

Team 7 Agent Design

Chapter 11

Experiments

Experiemntal design - What are we looking to find – Conditions that lead to stability/instability - Independent and dependent variables – Experimental parameters - Measurement methods

Chapter 12

Results

Chapter 13

Discussion

Chapter 14

Conclusion

Conclusion section.

Chapter 15

Future Work

Futur Work section.

Appendix A

Appendix

Appendix Section. Org chart could go here. Potentialy could include implementation details.

Bibliography