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

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 Modeling

Add diagram with a possible scenario of several days in a row?

3.4.1 Global Description

The health of the agents living in the tower is represented by Health Points (HP). Two mechanisms affect an agent's HP: how much food they eat, and their "cost of living". The cost of living represents how many calories a human needs to eat each day to stay healthy. These two mechanisms are implemented using the functions updateHP and hpDecay, respectively. These two functions are described below.

At the end of each day, agents are assigned an HP value based on how much food they have eaten and their cost of living. This HP value is an integer and has a maximum value of MaxHP, and a minimum value of HPCritical. As its name suggests, HPCritical is a critical HP value for the agents: they can only survive a certain number of days (MaxDayCritical) at this level. When in the critical state, if agents can increase their HP by HPReqCToW ("HP Required to move from Critical To Weak"), then they move into the "weak state" (Fig. 3.1), and their HP takes the value of WeakLevel. The amount that an agent's HP increases from eating is determined by the function updateHP().

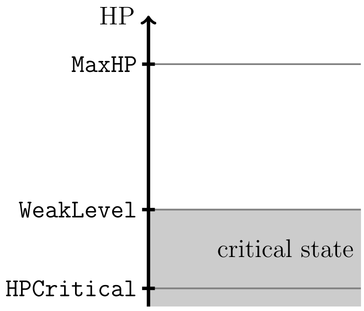

Figure 3.1: The health of the agents is represented by a HP value between HPCritical and MaxHP. All HP values which are below WeakLevel are classed as critical. The diagram is not drawn to scale.

3.4.2 Food and Health: updateHP

To increase their HP, agents need to eat. However, the amount an agent's HP improves can saturate in a single day; eating more than a certain amount will provide an agent with no extra benefit to their HP. Moreover, eating more food will lead to diminishing returns in terms of HP change. Mathematically, the ideas of diminishing returns and saturation are well captured by the step response of a 1st-order system (3.1):

$$\mathrm{newHP} = \mathrm{currentHP} + \underbrace{w(1 - e^{\frac{-\mathrm{foodTaken}}{\tau}})}_{\mathrm{HPChange}} \tag{3.1}$$

The two parameters w and τ are defined at the beginning of the simulation. The shape of this curve is given in Fig. 3.2 together with some important parameters.

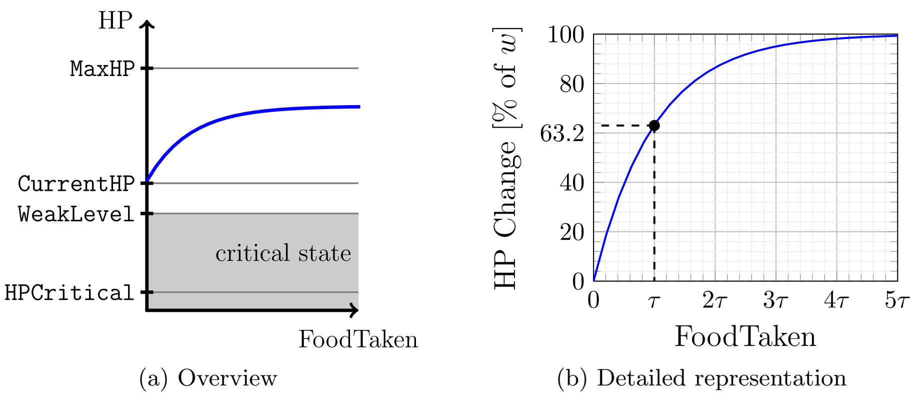

Figure 3.2: `updateHP` as a function of the amount of food eaten ("FoodTaken").

It is not possible to gain more HP than w over the duration of one day; this is an intentional limit to prevent an agent's health from improving too quickly. As an example, we can think of an agent that starts from the weak level and wants to reach the maximum HP value. It would take several days for this agent to "recover" from this weak level and stabilise its health to a high HP value.

Note that it is possible for an agent to achieve an HP value that is larger than `MaxHP` inside hpDecay(). At the end of each day, the hpDecay() function will apply the cost of living and then bound the final HP value by MaxHP.

Agents in the critical state are treated differently. For these agents, HP is updated according to equation (3.2):

$$\text{newHP} = \min\left\{\text{HPCritical} + \text{HPReqCToW}, \text{currentHP} + w(1 - e^{\frac{-\text{foodTaken}}{\tau}})\right\} \tag{3.2}$$

3.4.3 Cost of Living: hpDecay()

At the end of each day, the HP value of the agents will be reduced by the cost of living. The cost of living is larger for an agent with larger HP value than for an agent with lower HP value. This fact is motivated by a simple observation: humans that have stronger bodies and immune systems also need more food to sustain their level of health. The exact relation between HP value, cost of living, and HP value after applying the cost of living is given by the linear relation (3.3):

$$\text{newHP} = \text{currentHP} - [b + s(\text{currentHP} - \text{WeakLevel})] \tag{3.3}$$

The parameter b is a (constant) base cost, and s is the slope of the linear function. These parameters are initialised at the beginning of the simulation. Add a diagram for this?

To ensure that the HP value at the end of the day is bounded by `MaxHP`, we slightly modify (3.3) to produce (3.4):

$$\text{newHP} = \max\left\{\text{MaxHP}, \text{currentHP} - [b + s(\text{currentHP} - \text{WeakLevel})]\right\} \tag{3.4}$$

For agents in the critical state that gain `HPReqCToW` HP in a single day, i.e. their HP after eating is

$$\text{currentHP} \geq \text{HPCritical} + \text{HPReqCToW}, \tag{3.5}$$

their HP will be set to `WeakLevel`:

$$\text{newHP} = \text{WeakLevel} \tag{3.6}$$

Agents in the critical state which do not manage to improve their HP by HPReqCToW will be kept in the critical state:

$$\text{newHP} = \text{HPCritical} \tag{3.7}$$

with the `daysAtCritical` counter incremented by 1. If `daysAtCritical` reaches `MaxDayCritical`, the agent dies and is replaced. This counter is reset to 0 if an agent exits the critical state.

Chapter 4

Visualisation

Important things about front end.

Chapter 5

Team 2 Agent Design

Chapter 6

Team 3 Agent Design

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

S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, "Projective dynamics: Fusing constraint projections for fast simulation," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 1–11, 2014.