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

Co-operative survival games refer to a subset of political choice games wherein 'players' must work together to overcome disaster, else suffer the consequences through both personal and communal damage. Furthermore, without the ability to monitor a system and in the absence of a centralised authority, it becomes impossible to form enduring self-governing institutions, resulting in the proposed self-organising mechanisms for 'solving' such games potentially functioning in harmful ways. This project utilises a self-organising, multi-agent system to simulate an iterated collective action, co-operative survival game comprising a resource management problem, the 'solution' to which involves a system of localised governance. Through a series of experiments where the behaviour of the players in the system is increasingly randomised and the presence of self-organising mechanisms is toggled, we investigate the causes of instability. Ultimately, we conclude that through the integration of governance using treaties, it is possible for such a system to effectively self-organise to minimise the total deaths of the players involved.

Contents

Chapter 1

Introduction

Co-operative survival games refer to a subset of political choice games wherein 'players' must work together to overcome disaster, else suffer the consequences through both personal and communal damage. Instances of co-operative survival exist in varying media: computer games (e.g., "Don't Starve," "Rust," "Minecraft"), board games (e.g., Ravine) or sociological constructs (e.g., Reducing viral transmission) all of which exhibit a necessity for collective action, with such notions of survival supported by extensive literature [1]. Survival games are often identified within the context of "Common Pool Resource Management" (CPR) [2] [3] problems, where to facilitate co-operative survival, a pool of shared resources must be maintained to help mitigate disaster. In the general case of such survival games, following a disaster, if 1 player dies, all players die.

This project proposes a twofold bifurcation from conventional CPR problems: firstly, there are no provisions made to the common pool, meaning that we cannot refer to this problem as a linear public goods (LPG) game. Instead, players must maintain a collective resource without offering replenishment from their personal pool. This effectively constructs a scenario where all players make appropriation with insufficient provision, yielding a system of *N free-riders*.

Secondly, Ostrom notes that common-pool management problems are provably solvable through the introduction of self governing institutions [2], offering a contradiction to the tragedy of the commons and zero contribution thesis [4]. This project instead offers a scenario without centralised governance and an inability to monitor other players, effectively preventing this research from complying with Ostrom's "design principles for enduring self-governing institutions."

The absence of a centralised authority, coupled with an inability to form self-governing institutions proposes a seemingly unsolvable problem: a common pool of resources must be maintained across a network of players, each acting rationally to maximise local utility [5], without the prospect of forming enforceable rules. This necessitates the presence of a form of self-organisation sufficiently powerful as to overcome the challenges set by the nature of this problem, whilst accepting the risk that such a self-organising mechanism may act perniciously under these circumstances [6].

Apropos of Artificial Societies, to combat the difficulties poised by such an unconventional CPR game we introduce a notion of self-organisation through governance in the form of treaties. Serving both "as a juristic act and as a rule" [7], treaties act as a form of institutionalised power between players, with society coming to a mutual agreement upon the importance of such a legal device. Overall, it is the topic of this research to evaluate if the use of treaties is sufficient for self-organising this system, so as to achieve a level of stability when this game is played

iteratively in an *N-player* scenario.

Chapter 2

Simulation Structure

2.1 Scenario and Simulator

2.1.1 Context

The Platform presents an environment consisting of people subjected to a life within a tower. This tower has a predefined set of laws that closely control the people's lives, and it is ultimately up to these individuals to act however they suppose is correct.

The tower is made up of multiple floors with one agent per floor, starting with Floor 1 at the top and increasing downwards. Above Floor 1, there is a kitchen which produces a fixed amount of food per day. The floor on which people are based is random, and periodically reshuffles with a fixed frequency. At the beginning of each day a platform begins moving down the tower, starting from the top floor where it is loaded with food and stopping for a constant time at each floor. Over a period of time, there is enough food on the platform to minimally satisfice the agents, but not maximally satisfy them all.

The people exist with limited knowledge of tower; they must eat enough within a period to survive and can communicate with other individuals on local floors. They know which floor they are on, but do not know how many floors there are in the tower nor how much food there is at the beginning of each day. Additionally, the people in the tower do not explicitly know how often the reshuffle period is. Finally, when a person dies, they are immediately replaced.

The scenario presented above is heavily influenced by *'El Hoyo'*[8] (translating directly to 'The Hole') a Spanish film released in 2019.

2.1.2 Assumptions

There are several assumptions from the given context and, for the implementation in this project, are described in this section.

1. Pre-existing knowledge – people exist in the tower with a level of knowledge of their environment; this reduces the amount of learning required before action can take place. Pre-existing ideologies are included in this: people have their ways of interacting with others, views on socialising and the like, perhaps acquired from a life before the tower. Basic concepts such as consuming food to increase health are known.

2. Taking food – people are allowed to take as much food from the platform as they are able to eat. They cannot preserve food to eat on separate days. Additionally, the utility of the food taken decreases exponentially (each unit of food has a decreasing improvement to HP) – representing the law of diminishing returns.

3. An honest environment – people are assumed to be truthful: direct deceit does not exist. There are very specific conditions in which deceit is permitted, these will be discussed alongside the introduction of the relevant features, but is related to breaking agreements in times of extreme need.

2.1.3 Observations & Understanding

An initial observation of the context immediately presents the concept of a 'world', i.e., the tower, and 'agents', i.e., the people. These agents interact with one another and the tower, within constraints, and must be able to react and evolve their behaviour to maximise their utility and survival.

Within the tower there exists a food allocation problem, which is directly related to the survival of agents and forms the core incentive for self-organisation. With a lack of organisation, the tower society would fall victim to the tragedy of the commons – the common pool resource would deplete, and the overall well-being of society would suffer for the gain of a few individuals. Hence, the crux of this project is a collective action resource management problem, and agents must be equipped with sufficient capabilities and information.

The aforementioned capabilities can be split into two sections: those related to the tower and those related to the agent interactions.

Within the tower, agent abilities are listed below:

1. View Food on Platform – the agent cannot view the amount of food on the platform unless it is on the agent's floor or the floor immediately below the agent.

2. Food Consumption – agents cannot preserve food. Agents can only eat food from the platform when it is on the agent's floor.

3. Time Details – agents do not have explicit access to timed parameters (reshuffle period, day length, time of platform on each floor etc.), however, can use memory to calculate these.

4. Food Details – agents do not know the food on the platform at the beginning of each day.

5. Health Details – agents do know their current health level and the utility they are receiving from food, as well as how long they can go without food before dying.

Agents are also able to communicate with each other:

1. Agents should be able to converse with one another and share information.

2. Agents should be able to form agreements which must be followed.

3. Agents should be able to differentiate between any new agents that appear locally.

4. The communication must be limited to single agent interactions and single messages cannot travel past a defined, local distance.

2.2 System Design

Based on our initial observations and constraints, possible implementations were explored early on in the design process. Our design process evolved from designing for the simple use case of allowing simple agents to eat food and communicate with primitive messages, to scaling this up to accommodate multiple agents on multiple floors.

2.2.1 Ticks and Timing Details

Discrete time in the simulation is represented by ticks. During each tick, all agents can perform an action such as 'eat' or 'send a message'. This means that the number of ticks per day restricts the communication between the agents. Moreover, given that messages can only be passed from one floor to one below or above on each tick, we can conclude that certain messages will never get to their receiver. The number of ticks per day is set in the initialisation of the simulation and is the product of the ticks per floor and the number of floors.

2.2.2 Agent Configuration

Agent structures are separated into two different aspects: a base agent and a custom agent, both of which are required to represent a single agent. The base agent component contains the core information of an agent, including its Health Points, Floor and UUID[1], as well as a pointer to the tower structure. The custom agent contains a pointer to its base agent and any parameters relevant to the specific agent behaviour; this varies between agent types and was implemented by agent teams.

The core parameters of an agent were abstracted out of the custom agent for two reasons:

1. Custom agents cannot manipulate their own data or information, rather they can only request access through 'getter' functions, which returns either information or an error.
2. The tower struct has access to the base agents and can therefore influence the agent parameters as determined by the laws existing within the tower. Examples of this include floor reshuffling and health decay.

2.2.3 Death and Replacement

Deaths occur in the tower when an agent has not eaten enough food to stay alive over a certain duration. The nature of replacement in the tower could be of several types:

1. Replacing agents with random agent type
2. Replacing agents with the same agent type
3. Replacing agents with the dominant agent type

Our design chose to replace agents with the same agent type. While replacing agents with a random agent type would introduce an element of uncertainty, this would have made it difficult to analyse individual agent behaviours. It would have been interesting to experiment with

[1]UUID – Universally Unique Identifier

replacing agents with the dominant agent type, however, this posed the question of defining what the most "dominant" agent would be.

2.2.4 Infrastructure

Our implementation began with considering the structure of the system given the scenario we had identified. The key required components include a tower that contains agents, with a discrete-time system. The simulation package holds the state of the tower, base agents, and their corresponding custom agents. The simulation package that creates the world and constructs agents inside a simulation environment. It is the simulation that calls the tower and agent functions to progress the simulation. The tower is the overall 'world' which stores information in its state such as the list of agents, platform level, food available, and tick count. The tower is responsible for updating the platform, reshuffling agents, and replacing dead agents. The platform is a parameter of the tower and stores the amount of food on the platform and its current floor. It goes down one floor at a predefined tick rate (e.g., every 10 ticks) and it is reset (fill the platform with food and set its floor to 1) at the end of each day.

Base agent contains the core information of an agent, such as an agent's HP and floor. Base agent also contains the `hpDecay()` function as well as a function to calculate an individual agent's utility. Custom agents are designed by each of the agent teams, where each one of them has a different strategy.

2.3 Message Passing

2.3.1 Primitive Messaging

For agents to talk amongst each other in the simulation, the first step was to design a system which would allow an agent to pass along a minimum viable message to their neighbours. These were the goals of communication in the MVP:

- Agents may only talk to their immediate neighbors (± 1 floor).
- Agents should be given the ability to ignore communication.
- Multiple agents could be "speaking" in a given tick.
- One agent can send one message per tick.
- **It is entirely at an agent's discretion if/when/how they wish to react to a message as long as their behavior is not deceptive**

The infrastructure should not force any behaviour onto the agent as long as the behaviour is honest. The basic philosophy of the communication infrastructure was to dictate as little as possible about the agent's behavior while keeping the API readable and user-friendly. This meant providing basic override-able behaviours that were as unobtrusive as possible to team strategies.

- `MessageType`: Indicates one of 13 enumerated message types.
- `SenderFloor`: Returns the floor this message was sent from.

- `TargetFloor`: Returns the floor the message is addressed to.
- `ID`: Each message sent is given a unique ID.

For the MVP, Team 1 constructed a system in which, for example, when Alice (currently at floor 30) wishes to send a message upward:

1. At Tick 0, Alice would construct the message for Bob containing the following information:
    - **Alice's ID**
    - **Alice's current floor**: 30
    - **Message's target floor**: 29 *Since Alice wants to message the floor above her*
    - **Message Type** (to be elaborated on the Common Languages Section)
2. Still at Tick 0, Alice calls `SendMessage` which takes her message and passes it to the tower.
3. The tower acts as the communication authority, finding the agent who is on floor 29 at Tick 0 and inserting the message into the recipient agent's (Bob's) inbox.
4. At Tick 1, Bob may choose to call `ReceiveMessage` which would extract Alice's message (if Alice was the first or only agent to send him a message at Tick 0) and respond.
5. If Bob has called `ReceiveMessage`, Alice's message is removed from Bob's inbox.

With this implementation, each agent is responsible for calling `ReceiveMessage` once per tick to receive any messages. The inbox is a FIFO queue – if multiple messages are received or if there are outstanding messages from previous ticks, only the earliest one can be retrieved in a given tick. Additionally, "receiving" does not necessarily imply "reacting". An agent is capable of receiving a message and not responding to it at all without the sender's knowledge. Decoupling "receiving" and "reacting" gave agent teams the freedom to ignore messages entirely (metaphorically covering their ears in the tower and refusing to listen to anybody), to listen to messages and do nothing about them, or to actively listen and react.
Beyond the MVP, however, messages needed to have meaning. Because the process would be automated it was meaningless to send unstructured strings, how was an agent to understand "Please only eat what you need to survive."?
A common language was necessary.

2.3.2 Common Language

To design the common language, Team 1 surveyed the agent teams asking for what they would like to talk to other agents about. We collected suggestions and found that communication indicated one of four possibilities:

1. An agent could be *asking* another agent for information. "How much food is on the platform when it gets to you?" or "How much food did you take?"
2. An agent could be *stating* something about its state or environment. "I am in critical (health) condition" or "There is no more food left on the platform"

3. An agent could be *request* something from another agent. "Please leave 10 food on the platform for me."

4. An agent could be *responding* to another agent's request. "Yes." or "No."

These categories solidified into four basic messages - `Ask`, `State`, `Request`, and `Response`.
In addition, `Ask` and `Request` are messages that expect a response. A `State` message could be a response to an `Ask` but also could be an unprompted announcement while `Response` must be responding to some pre-existing `Request`.
While Team 1 did not want to force agent teams into any behaviors, we wanted to make the API compatible with the expected etiquette of communication. This lead to the message categorization and pair-wise reply functionality summarized below.

Message Category	**Reply Category**	**Body Functions**	**Description**
`AskMessage`	`StateMessage`	`Reply`: Returns the appropriate statement	Inquires something about a neighboring agent's state.
`StateMessage`	N/A	`Statement`: Returns the value of statement. *e.g. a StateHP message of 5HP would return 5.*	Announces something about an agent's state or environment.
`RequestMessage`	`ResponseMessage`	`Reply`: Returns `BoolResponse` `Request`: Returns the value of a request *e.g. a RequestLeaveFood message that requests an agent to leave 10 food would return 10 on* `Request`	Asks how much food is on the platform when it arrives at the (recipient) agent
`ResponseMessage`	N/A	N/A	Asks how much food an agent is planning to take

From these basic categories 11 specific messages were developed that specified what was being asked, stated, requested or responded to.

Message Type	**Category**	**Description**	**Reply Type**
`AskFoodTaken`	`AskMessage`	Asks how much food an agent has (already) taken	`StateFoodTaken`
`AskHP`	`AskMessage`	Asks how much HP an agent has	`StateHP`
`AskFoodOnPlatform`	`AskMessage`	Asks how much food is on the platform when it arrives at the (recipient) agent	`StateFoodOnPlatform`

`AskIntendedFoodIntake`	`AskMessage`	Asks how much food an agent is planning to take	`StateIntendedFoodIntake`
`StateFoodTaken`	`StateMessage`	States how much food an agent has taken	
`StateHP`	`StateMessage`	States how much HP an agent has	
`StateFoodOnPlatform`	`StateMessage`	States how much food is on the platform when it arrives to the agent	
`StateIntendedFoodIntake`	`StateMessage`	States how much food the agent is planning to take	
`RequestLeaveFood`	`RequestMessage`	Requests that an agent (presumably above you) leaves a certain amount of food on the platform	`BoolResponse`
`RequestTakeFood`	`RequestMessage`	Requests that an agent (presumably above you) takes a certain amount of food on the platform	`BoolResponse`
`BoolResponse`	`ResponseMessage`	Message affirming or rejecting a request	

The distinction between "receiving" and "reacting" becomes important. We wanted to ensure that *receiving* any of these message types did not mean that the recipient had to *react*. For that reason, the "reaction" was separated into an external function which the agent could optionally call after they extracted their message from the inbox.
The design works such that if Alice on Floor 10 wanted to ask Floor 10 to leave 10 food on the platform for her:

1. At Tick 0, Alice constructs a `RequestLeaveFoodMessage` with `Request` set to 10 addressed to Floor 12.
2. At Tick 0, Alice sends the message.
3. At Tick 1, Bob (whose inbox was previously empty and who was listening for messages) receives Alice's message. If he chooses to react, he decides whether or not he wants to cooperate with Alice and generates his response with the `Reply` function of the message (no need to construct the *BoolResponseMessage* himself).
4. At Tick 1, he sends his reply back, addressed to the message's sender floor.
5. At Tick 2, Alice (whose inbox was also previously empty) receives Bob's response and may choose to react to it.

The system so far provided a solid mechanism for agents to communicate to their neighbours, to gather information and to request help in times of crisis (potentially building trust or temporary

relationships). However, it was contingent on communicating agents being on consecutive floors, any relationships built through this kind of communication could only last for a reshuffle period unless the agent implemented a specific kind of internal memory structure. Additionally, the communication was still relatively rudimentary. Alice's request for 10 food could be rejected by Bob because his HP is critical and he will starve immediately without food, but the current system only allows Bob to either accept or refuse Alice's request without communicating anything about his own state. This could potentially worsen his relationship with Alice, had Alice known that Bob was in a critical state perhaps her trust to him would not degrade because of this rejection.
For more nuanced, long-lasting communication, treaties and message propagation were needed.

2.3.3 Treaties

Treaties are formalized agreements between agents who agreed to a set of conditions and corresponding requests. An example treaty might be *"Please leave 40 food on the platform if you are not in critical condition"*. In an honest run of the experiment deception was disallowed and it was ensured that if an agent agreed to a treaty, they would be be bound to uphold it.
This required designing an internal mini-language for how treaties should be written and understood. It was decided that a treaty consisted of 6 language components:

Field	**Description**
`ConditionType`	Dictates the type of condition the treaty is active with possibilities of: • `HP`: The treaty's activeness is dependent on the signing agent's HP. • `Floor`: The treaty's activeness is dependent on the signing agent's current floor. • `AvailableFood`: The treaty's activeness is dependent on the signing agent's available food (food on platform).
`ConditionOp`	Includes all the mathematical operators $\geq, >, =, <, \leq$
`ConditionValue`	Value that the `ConditionType` has to meet the `ConditionOp` for.
`RequestType`	The request agreement for the treaty, possible requests are: • `LeaveAmountFood`: Requests a signed agent to leave a fixed amount of food. • `LeavePercentFood`: Requests a signed agent to leave a percentage of food on the platform. • `Inform`: Requests a signed agent to alert their neighbor if treaty conditions are met.
`RequestOp`	Includes all the mathematical operators $\geq, >, =, <, \leq$
`RequestValue`	Value that the `RequestType` has to meet the `RequestOp` for.

The aforementioned treaty *"Please leave 40 food on the platform if you are not in critical condition"* would be translated into

```
ConditionType = HP
ConditionOp = GT
ConditionValue = 20 //(where 20 is the critical threshold)
RequestType = LeaveAmountFood
RequestOp = EQ
RequestValue = 40
```

In addition to the language components the treaties included a `SignatureCount` which tracked an estimate of how many agents had signed the treaty thus far. This number is not an accurate estimation, it only vaguely indicates whether or not a treaty is popular. The introduction of treaties also necessitated the creation of one more message category as well as a message sub-type.

Message Category	**Reply Category**	**Body Functions**	**Description**
`ProposalMessage`	`ResponseMessage`	`Reply`: Returns `TreatyResponseMessage` `Treaty`: Returns the treaty that the proposal is carrying	Carries a treaty from proposer/propagator to recipient.

`TreatyResponse` sub-type of `ResponseMessage` which were response messages that contained the ID of the treaty it was responding to.

The treaties-relay-design and the inaccuracy of the signature count is illustrated if Alice, still on Floor 10, wanted to propose the treaty *"Please leave 40 food on the platform if you are not in critical condition"* to her neighbours downstairs and upstairs it would work as follows:

1. Tick 0: Alice constructs the treaty, embeds it within a `ProposeTreatyMessage` and sends it upward (as she can only send one message per tick). Alice's treaty begins with a `SignatureCount` of 1 (since she implicitly signed it by proposing).

2. Tick 1: Treaty arrives on the 9th Floor in Bob's (previously empty) inbox and he chooses to respond positively and signs. On Bob's local copy of the treaty, the `SignatureCount` has incremented to 2 but not on Alice's. Bob replies through `Reply` and sends the `TreatyResponse` downstairs.

3. Tick 1: Meanwhile, Alice also sends the treaty downward to Floor 11.

4. Tick 2: Treaty arrives on the 11th Floor in Carol's (previously empty) inbox and she also chooses to respond positively.
   Here, Carol's local copy of the treaty has a `SignatureCount` of 2 not 3 despite the fact she is the third person to sign.

5. Tick 2: Meanwhile, Alice has received Bob's affirmation. Knowing that he signed it, she increments her copy of the treaty.

6. Tick 3: Alice also receives Carol's affirmation, she increments her local `SignatureCount` to 4, making her the only person with access to the correct signature numbers.

7. Tick `n`: In some future tick post-reshuffle where Alice, Bob and Carol are no longer neighbours, the three of them and whoever else signed copies of the treaty (if Carol or Bob

further propogated the treaty) are bound to the agreement that if they are not in critical condition, they would leave 40 food)

Although we discussed several approaches for keeping an accurate state of the `SignatureCount`, they required too much communication overhead which could be detrimental to strategies since message sending is capped at one per tick. Agent teams opposed the idea of removing the field altogether so an inaccurate but potentially useful `SignatureCount` was kept.
Because the honest run of the experiment requires that no agent breaks their treaties, the base agent rejects treaties by default. This default was expected to be overridden by any agent team who wanted to communicate with other agents, so this introduced difficulties in how to ensure that treaties that had been signed were followed. There were suggestions for a technical solution however ultimately that was too restrictive for agent strategies as it would require Team 1 to somehow conditionally override agent behaviour depending on if they had an active treaty or not. Ultimately it was decided that the honesty would be maintained through a combination of readable agent code and thorough code reviews from Team 1 as treaty strategies were implemented.
With the advent of treaties and the possibility of building relationships that outlived a reshuffle, it was time to extend communications beyond immediate neighbors to possibly tower-wide.

2.3.4 Message Propagation

Multi-floor communication was a long-awaited feature among agent teams so there had already been many conversations about how best to achieve this. There were two main approaches:

1. **Shouting**: An agent could shout up or down a specified number of floors and every agent on floors in between would also hear the message.
2. **Propagating**: An agent could only speak to its immediate neighbour but they could instruct their neighbour to pass the message on to the target floor.

Option 1 required some limit on the number of floors you could yell across. Agent teams also suggested that shouting should not be a free action and should "cost" HP when engaged in often. Although interesting, it introduced unnecessary complexity and was discarded.
Option 2 also had some interesting implications. There were multiple possibilities in how the in-between agents would behave:

- An in-between agent could break the propagation chain and not inform the original sender.
- An in-between agent could tamper with the message before passing it on.
- An in-between agent could eavesdrop on the message being passed.

After some discussion, it was decided that for honest experiments breaking the chain without informing and message-tampering were disallowed while eavesdropping remained legal.
We set the default behavior of propagation to be "dutifully pass the message if it was not addressed to your floor" on the infrastructure level. This was so agents who did not wish to eavesdrop or did not engage in intra-agent communication at all would not need to implement separate handlers for propagation.
With the completion of message propagation, communication mechanisms within the tower were finished.

2.4 Health Modeling

2.4.1 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 `WeakLevel` (Fig. 2.1), and their HP takes the value of `WeakLevel`. The amount that an agent's HP increases from eating is determined by the function `updateHP()`.

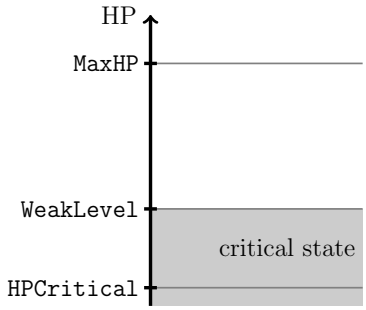


Figure 2.1: The health of the agents is represented by a HP value between `HPCritical` and `MaxHP`. All HP values which are below `WeakLevel` are classed as critical. The diagram is not drawn to scale.

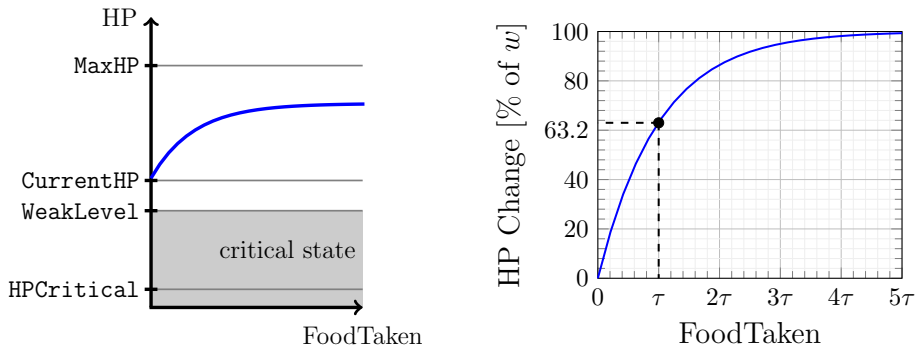
2.4.2 Food and Health: `updateHP`

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 (2.1):

$$\texttt{newHP} = \texttt{currentHP} + \underbrace{w(1 - e^{\frac{-\texttt{foodTaken}}{\tau}})}_{\texttt{HPChange}} \tag{2.1}$$

The two parameters w and τ are defined at the beginning of the simulation. The shape of this curve is given in Fig. 2.2 together with some important parameters.

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



(a) Overview (b) Detailed representation

Figure 2.2: `updateHP` as a function of the amount of food eaten ("FoodTaken").

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 (2.2):

$$\texttt{newHP} = \min\left\{\texttt{HPCritical} + \texttt{HPReqCToW}, \texttt{currentHP} + w(1 - e^{\frac{-\texttt{foodTaken}}{\tau}})\right\} \tag{2.2}$$

2.4.3 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 (2.3):

$$\texttt{newHP} = \texttt{currentHP} - [b + s(\texttt{currentHP} - \texttt{WeakLevel})] \tag{2.3}$$

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.

To ensure that the HP value at the end of the day is bounded by `MaxHP`, we slightly modify (2.3) to produce (2.4):

$$\texttt{newHP} = \max\left\{\texttt{MaxHP}, \texttt{currentHP} - [b + s(\texttt{currentHP} - \texttt{WeakLevel})]\right\} \tag{2.4}$$

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

$$\texttt{currentHP} \geq \texttt{HPCritical} + \texttt{HPReqCToW}, \tag{2.5}$$

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

$$\texttt{newHP} = \texttt{HPCritical} \tag{2.6}$$

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.

2.5 Utility and Social Welfare

To assess the performance of the agents in the tower as a group, we first need to define a metric. A common choice is the so-called *social welfare*, based on each agent individual utility. For this project, we implement the notion of utility as introduced in [9].

Our system is composed of N agents that can perform specific actions in relation with the common pool resources. In a general context, each agent $i \in \{1, \dots, N\}$ takes the following actions at each iteration $t \in \{1, \dots, \infty\}$:

1. Determines the resources it has available, $g_i \in [0, 1]$.
2. Determines its need for resources, $q_i \in [0, 1]$.
3. Makes a provision of resources, $p_i \in [0, 1]$.
4. Makes a demand for resources, $d_i \in [0, 1]$.
5. Receives an allocation of resources, $r_i \in [0, 1]$.
6. Makes an appropriation of resources, $r'_i \in [0, 1]$.

In the current setup, the available resources q_i corresponds to the current amount of food on the platform. The need for resources q_i is defined in relation with the health of the agents. We set the following values to q_i:

$$q_i = \begin{cases} \frac{\texttt{numberDaysInCriticalState}}{\texttt{maxDaysInCriticalState}} & \text{if } \texttt{currentHP} \leq \texttt{weakLevel} \\ 0 & \text{else.} \end{cases} \tag{2.7}$$

This way, we ensure that q_i is bounded by 1 and is proportional to the days spent in the critical health zone below `weakLevel`.

Moreover, the agents do not make any provision p_i to the common pool, as they cannot give food to the platform ($p_i = 0$). Their demands for resources is the food they ask for when the platform is at their level. In addition, the agents appropriate all resources they are allocated, so that $r'_i = r_i$.[2]

The total resources accrued at the end of an iteration, R_i, is hence defined as:

$$R_i = r'_i + (g_i - p_i) \tag{2.8}$$

[2]We ensure that all these parameters are constrained to the range $[0, 1]$ by dividing the mentioned quantities by their maximum values.

where each agent will 'generate' resources equal to: its appropriation, plus the amount available on the platform, minus the provision made back to the common pool.

Using the above parameters, it is possible to compute the following utility per agent:

$$u_i = \begin{cases} \alpha_i q_i + \beta_i (R_i - q_i) & \text{if } R_i \geq q_i \\ \alpha_i R_i - \gamma_i (q_i - R_i) & \text{else} \end{cases} \tag{2.9}$$

where α_i, β_i and γ_i are tuning parameters that follow the rule $\alpha_i > \gamma_i > \beta_i$. In our work, we use the values $\alpha_i = \alpha = 0.2$, $\beta_i = \beta = 0.1$, and $\gamma_i = \gamma = 0.18$.

Finally, we use (2.9) to compute an average global utility, which corresponds to the social welfare SW divided by the number of agents:

$$U = \frac{\sum_i^N u_i}{N} = \frac{SW}{N} \tag{2.10}$$

2.6 Technology Stack

Our technology stack was chosen in the initial meetings of the project, where the individuals were able to propose languages for the frontend and backend, with the backend language used to implement infrastructure and agents. The proposed languages for the backend were:

1. Python: A widely used, general purpose programming language with an emphasis on code readability. The language was familiar to a large percentage of the class as a result of having used it in previous projects.

2. Go: A less well known language, Go was attractive to the class due to its easy-to-learn nature and easy implementation of concurrency which would be useful in running multiple agents. However, it was unfamiliar to the majority of the class.

3. C++: A general purpose OOP language, C++ was familiar to a portion of the cohort having studied it in their first year. However, it is not as easy to learn as Go or Python, so this was considered a disadvantage.

After an in-depth review, a vote was cast and Go was the clear preference. Overall, Go was chosen due to its advantages in implementing concurrency and its easy-to-learn nature. In addition to this, the packages in Go would make the code more scalable and readable.

React and Typescript were chosen as the frontend language. They were chosen primarily by the infrastructure team, as this decision would not affect those teams working on agent development.

Chapter 3

Data Logging and Dashboard

3.1 Need for Data

Data logging allows for the analysis of how internal variables change with time during a simulation. One might be interested in different variables from the system in question, such as messaging frequency, food variation over time or the agents' death rate, with analysis also requiring the linking of different simulation parameters with their logged data (e.g., the number of agents or the amount of food available).

It is therefore important to setup a logging framework that structures data effectively and scales efficiently for large simulations. The below subsections will introduce the simulation parameterisation, followed by the different logging strategies (dumps and structured), and concluding with the logging interface and the analysis dashboard.

3.2 Parameterisation

First, it was important to enable an effective parameterisation of the simulation. A configuration interface was therefore set to abstract variables from the simulation environment and a configuration object is passed to the simulation environment instance to define the following simulation parameters:

1. Agent count for each team
2. Agent count for random agents
3. Agent count for selfish agents
4. Agent HP
5. Agents per floor
6. Amount of food on the platform
7. Food per agent ratio (optional)
8. Ticks per floor
9. Total days
10. Reshuffle period

The configuration parameters are then written in a JSON file which can be parsed into a configuration object.

3.3 General Log Dumps

Following the configuration work, a general logger was set up. The general logger is effectively a single shared logger between the different instances in the system (simulation, tower, agents, etc...). Each instance appends by default an identifier to a log line in the general logger log file (named `main.json`). Here is an example below:


```
{"agent_id":"dece1252","agent_type":"Team5","floor":3,"health":100,
"level":"info","msg":"Reporting agent state of team 5 agent","reporter":"agent"}

{"Floor:":7,"HP:":100,"Social motive:":"Altruist","agent_id":"2bfd3cfb",
"agent_type":"Team6","level":"info","msg":"Reporting agent state:","reporter":"agent"}

{"agent_id":"7873bb50","agent_type":"Team1","floor":5,"health":100,
"level":"info","msg":"Reporting agent state","reporter":"agent"}
```


The general log dumps are not structured, yet they enable agent teams to directly log custom states from their agents and use them, for example, in evolutionary algorithms or machine learning training.

3.4 Structured Logs

The general log generates huge amounts of data and therefore was ineffective for data analysis and drawing conclusions easily from different simulations. Therefore, structured logs had to be introduced, where particular data are captured optimally to generate lightweight, understandable logs. After discussions with agent teams, the following structured logs where included:

1. **Food**: food available on the platform and food taken by agents.
2. **Death**: agent death (including agent state at death time)
3. **Messages**: receiver, sender, message type and content
4. **Utility**: agent utility (including agent state)
5. **Story**: a combination of the above logs in a sequential manner

The agent state encapsulates by default the agent's age, floor, HP, utility, and custom state (e.g., emotions or memory).

In the general log, custom agents could pass all sorts of information and therefore the log schema could not determined at compile time (which is required for a visualisation dashboard). Structured logs have a well-defined schema which does not depend on custom agents. For example, a food log will always take the following form:


```
{"day":2,"food":19,"level":"info","tick":222,"time":"2022-01-17T02:13:37Z"}
{"day":2,"food":0,"level":"info","tick":262,"time":"2022-01-17T02:13:37Z"}
{"day":3,"food":100,"level":"info","tick":321,"time":"2022-01-17T02:13:37Z"}
{"day":3,"food":90,"level":"info","tick":322,"time":"2022-01-17T02:13:37Z"}
```

With structured logs set up, it is now possible to bundle logging into an interface and connect it to a visualisation tool.

3.5 Logging Interface

It was important to set up logging such that structured loggers could be easily extended with new loggers, while preserving the ability to generate general logs. The following design was adopted to encapsulate the simulation configuration and the loggers.

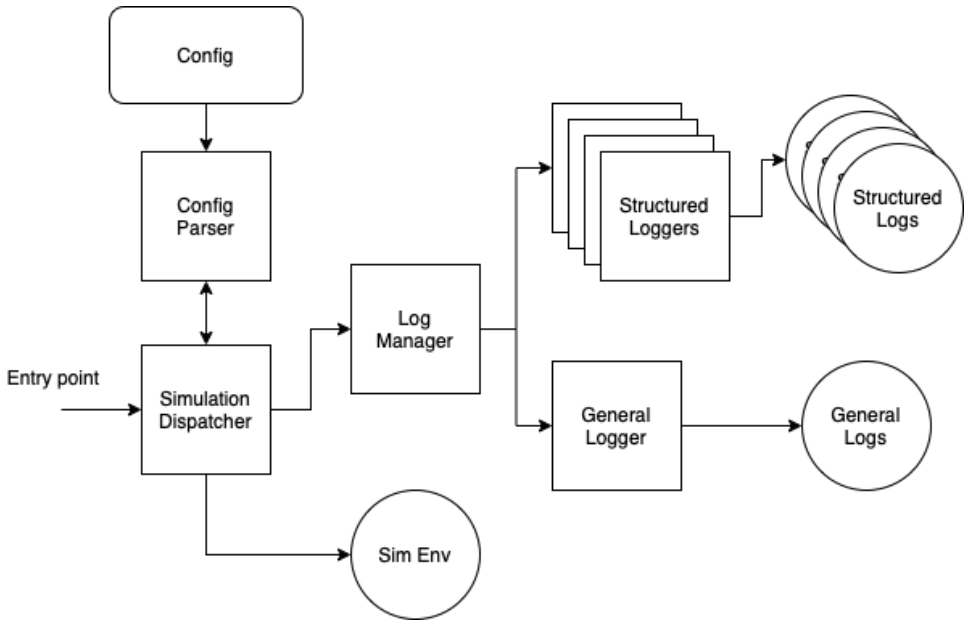


Figure 3.1: Diagram showing how logging fits in the system

The entry point to the simulation dispatcher is linked to both the command line interface as well as a server (with HTTP listeners) to support a browser-based dashboard.

3.6 Dashboard

With the logging interface, it was now possible to connect to the simulation entry point via a frontend and create simulations or visualise results. By design, the frontend had to support the following features:

1. **Manage simulations**: easily navigate between different simulations
2. **Create a simulation**: configure and run a new simulation
3. **Numerical results**: display numerical outcomes such as total deaths or average utility
4. **Graphical results**: display outcomes such as message types per agent type or food available per floor

To encapsulate all these features, the dashboard was designed with the following structure:

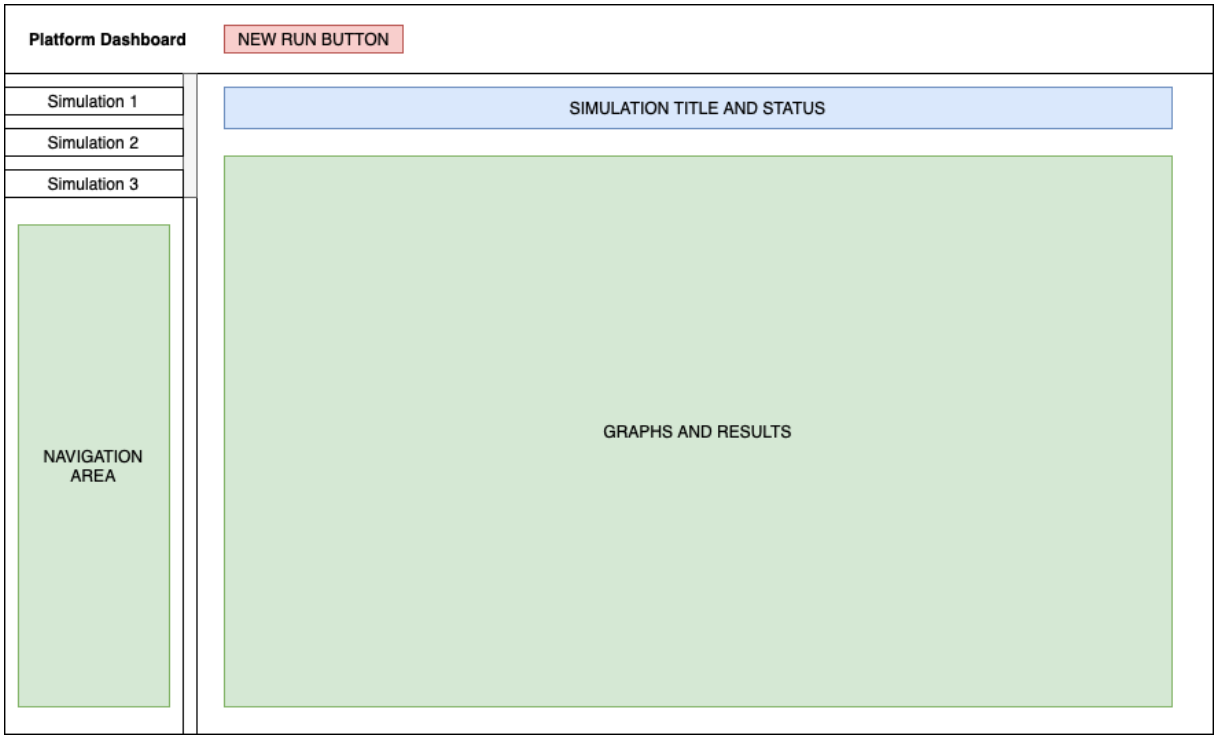


Figure 3.2: Diagram showing the frontend design

The "new simulation" button pops a configuration form where the user is able to fill in the different simulation parameters. The navigation area allows the user to switch between existing simulation results. Finally, numerical cards and different graph types have been included to visualise the results. The final frontend implementation can be tested online on https://somas-2021-568r2.ondigitalocean.app or locally by following the guidelines on the GitHub repository.

Chapter 4

Team 2 Agent Design

4.1 Core Idea and Overall Strategy

The objective of this agent design was to implement a form of reinforcement learning to allow for an increase in performance and thus utility over time. This must be met without compromising on the overall goal of the simulations, whereby the agent must attempt to strike a balance between prioritising the micro-level goal of maximal individual utility or prioritising the macro-level goal of maximising collective utility through sustainable resource management, the latter involving a level of cooperation with the other agent types. Certain parallels can be drawn to evolutionary ecology, as the tower with the resident agents can represent a closed ecosystem, with the platform providing a limited food source.

4.1.1 Teleo-Reactivity

Residents of the tower are simulated as autonomous agents, which are functioning within an uncertain environment. Each agent has the individual goal of survival, and each agent is able to sense the amount of food on the platform, upon its arrival at their floor. Thus, the immediate actions of the agents are to achieve the goal of survival. This can be regarded as a Teleo-Reactive program [10], which is defined as "one whose actions are all directed towards achieving goals in a way appropriate to the current perceived state of the environment". This sets one of the foundational design principles required for a successful agent – that is, to continuously monitor relevant information as the circumstances of the agent change. Ideally, the agent will be able to quickly adapt and be able to conduct drastic behavioural changes when required, but this can only be achieved after a certain period of learning from the environment.

4.1.2 Competitive Exclusion Principle

Competing and cooperating with the other agents may seem to contradict the Competitive Exclusion Principle. The Competitive Exclusion Principle states that two or more species competing for the same limited resource will not be able to exist at constant populations and will inevitably lead to either the weaker competitor's extinction, or a distinct change in its behaviour (i.e. "complete competitors cannot coexist" [11]). The superior competitor generally will have an aggressive competitive advantage which leads to this. This principle aligns with our context, as our agents must satisfy their hunger with limited food being delivered on the platform, while

other team agents will do the same. Conversely, we must not cause the deaths of other agents by taking too much food for ourselves since all agents must also strive to achieve the highest collective utility possible. In order to balance these objectives, we have implemented functions unique to our agents to ensure the survival of our agent without leading to the demise of others.

4.1.3 Dynamic Adaptation

Our goal is to create a unique advantage by allowing for the agent to experiment with different actions to different situations, until the most appropriate actions can be dynamically chosen for the future. A reward system is utilised, whereby a suitable action will be rewarded, and a detrimental action will be penalised. Given enough iterations, the agent should be sufficiently trained to consistently choose the most correct actions. This is achievable as most other agents have fixed behaviours which can be predicted by our agent; however, we also experiment with random agents to understand and quantify the change in performance and utility. Relying on a stochastic method to execute learning can result in vastly different outcomes, but it is expected that in general the agents will perform better with time.

4.1.4 In the Pursuit of Organisation

Under ideal circumstances, the agents will unite into an organisation where they can collectively decide the resource division strategy for that iteration. This would also allow for a pseudo-democratic justice system, whereby the agents can vote upon the reward or punishment of each other, if one respects or violates the values of the organisation. More generally, organisations can be defined as "collectivities oriented to the pursuit of relatively specific goals and exhibiting relatively highly formalized social structures" [12]. At first, there may seem to be a straightforward solution in forming an organisation, given the benefits mentioned, but formation of such a society requires great risk on the part of the agent. There are significant constraints preventing this strategy from being conveniently utilised; trust is risky and difficult to build amongst agents given the strong random nature of the environment, the food resources are too limiting to allow for the privilege of altruistic risk-taking, and the number of actions able to be taken per day are also highly limiting. These factors will result in agents somewhat selfishly prioritising survival over anything else, thus complete self-organisation will be a difficult, if not impossible to achieve within this environment. A mechanic available to agents is the "treaty", which are formal agreements upheld by the participating agents to dictate behaviours for a set period of time. For a non-dynamic agent, this is beneficial, as it can use the knowledge of other agents to switch to a superior set of behaviours. For a dynamic agent such as ours, treaties will override any and all learning developed up to that point in the simulation and will prevent any significant learning for the duration of the treaty. Thus, it is in the best interests of a learning agent such as ours to reject all treaties, which unfortunately comes at a cost of preventing any progress to Tower-wide agent organisation.

4.1.5 Utility Considerations and Pareto Optimality

It is not possible to maximise both individual and collective utility simultaneously. Maximising only individual utility would result in taking more resources than required to satisfy an agent's needs to ensure that it can survive on floors with less food. This action is contradictory to maximising collective utility as it would unnecessarily reduce the resource pool for the other agents. As a result, collective utility is prioritised to enhance the survival rates of all the agents

in the tower. In principle, the consistent maximisation of the collective utility will lead to a Pareto optimal outcome, whereby there will be "no other outcome that makes one player better off without making another player worse off". In actuality, due to the Competitive Exclusion Principle, the agents can only achieve this if they all converge their behaviours to form a single new agent type. This is the only method to ensure the collective survival of all the agents whilst also removing the competition which prevents them from maximising the collective utility (and thus achieving the Pareto optimal scenario).

4.1.6 Education and Reincarnation

The knowledge gained over time is stored in the agent's memory. In order to provide a more realistic simulation of the tower scenario, each individual agent's memory will be erased upon death. This presents a limit to the effectiveness of the learning method, as the agent's learning rate cannot be so high that it only takes higher risk choices to learn faster, it must consider its survival. However, a separate setting was made where "reincarnation" was implemented. In this setting, upon the agent's death, it is respawned with all the knowledge it had gained in previous lives. We use this as a comparison metric to qualitatively assess any difference in overall utility and test the model with maximum iterations to learn as opposed to just its lifespan.

4.2 Strategies and Algorithms

4.2.1 Q-Learning

Instead of using a deterministic method, we apply Q-Learning to dynamically learn a strategy for our agent based on the environment given by the game settings. The Q-Learning method we use is an extension of the classic Temporal-Difference (TD) learning.

One-Step Temporal-Difference Learning

Temporal-Difference (TD) Learning combines the attributes of Monte Carlo methods and Dynamic Programming. Like Monte Carlo, TD has the ability of online learning from the specific feedback that the agent receives without the need to construct a model of the environment. In addition, TD is equipped with the advantage of Dynamic Programming. It is able to update the estimates of the current state based on other estimations of future states, which means that it is not necessary for TD learning to wait for the final outcome when updating the strategies [13].

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{4.1}$$

where V is the value of states and R is the reward for transferring to some state.

The $R_{t+1} + \gamma V(S_{t+1})$ term is the reward for transferring to the next state plus the time-decayed value of the next states. This is the approximation of the Markov Reward Process if we keep iterating this step until reaching the final state. Thus, $R_{t+1} + \gamma V(S_{t+1})$ can be considered as a better estimation of the value for the current state. In this case, it can be easily observed that $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is the error between our estimated current state value and its better approximation. Finally, the value of the current state is re-estimated with this error plus its previous estimated value.

The TD learning method has been proven to be able to converge. It is not difficult to imagine that if we maintain this updating process for the values of all states, we will be able to reach a good approximation to the value defined in the Markov Reward Process.

Q-Learning Algorithm

Q-Learning is an off-policy version of the Temporal-Difference method.

Instead of only considering the transition between states, policies and actions are introduced in Q-Learning. An action is a behaviour that the agent can perform under some situation. A policy is the behaviour for an agent at a given time, which can be interpreted as the mapping from a given state to a given action corresponding to that state. For an off-policy algorithm, the policy that is evaluated and improved can differ from the policy that the agent will take to introduce the action. In other words, we may update our strategy under the assumption that the agent will take one action in the next state, but the agent may eventually take another action.

In Q-Learning, the transition between state-action pairs is involved. The state-action pairs also belong to the Markov Reward Process as in the TD method. Q value measures the value of a state-action pair which means that Q value tells the quality of taking one action under given state. Moreover, a Q table is a combination of all the Q values which includes all the possible states and corresponding actions along with their Q values [13].

In terms of the actual policy of an agent in Q-Learning, we combine the exploration policy and greedy policy. In greedy methods, the agent selects an action which has the highest quality under the current state. In greedy methods, the agent selects an action which has the highest quality under the current state. In other words, the agent will select the action with the highest Q value in that state. The exploration policy introduces a probability δ, which is the probability that the agent will ignore past experience (i.e., the Q table) and explore potentially better results. In summary, the agent will have a probability of δ to take a random action and a probability of $1 - \delta$ to follow the greedy policy.

The Q table is updated with the following function

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{\alpha} Q(S_{t+1}, a) - Q(S_t, A_t) \right] \tag{4.2}$$

The updating function is nearly the same as that in TD learning except that we now measure the value of a state-action pair. The error between the estimated Q value and the better estimated Q value of the current state is also utilised in the updating process. Although the agent has a probability to take the random policy, we assume that the agent acts according to the greedy policy for updating which is expressed in the term $\max_{\alpha} Q(S_{t+1}, a)$. Such setting ensures a stable but exploratory learning process and that is why Q-Learning belongs to the off-policy category.

4.2.2 Hill Climbing

To improve the process of changing strategies we implement the Policy Hill-Climbing (PHC) algorithm, which is a simple extension of Q-learning. PHC is a simple adaptation that performs the hill-climbing algorithm in the space of mixed strategies. Here, the starting point of the policy space is initialised so that all policies have a uniform probability, unlike the more common random initialisation of probabilities. The PHC algorithm is formulated in Algorithm 1.

Algorithm 1: PHC Algorithm

$\alpha \in (0, 1]$; ▷ `Learning rate`

$\beta \in (0, 1]$; ▷ `Learning rate`

$Q(s, a) \leftarrow 0$;

$\pi(s, a) \leftarrow \frac{1}{|A_i|}$;

begin ▷ `Perform each iteration`

  $a \leftarrow \pi(s)$;

  $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left(r + \gamma \max_{\alpha} Q\left(s', a'\right)\right)$;

  $\pi(s, a) \leftarrow \pi(s, a) + \begin{cases} \beta & \text{if } a = \text{argmax}_{a'} Q(s, a) \\ \frac{-\beta}{|A_i|-1} & \text{otherwise} \end{cases}$;

PHC keeps the Q-values as normal Q-learning would, but additionally also retains the present mixed policy. The algorithm performs non-gradient based rational optimisation by increasing the probability of selecting the action that gave the highest Q-value according to some given learning rate α. The algorithm improves the policy by increasing the probability for selecting action with the highest Q-value according to the learning rate $\beta \in (0, 1]$. It should be noted that for a learning rate $\beta = 1$ the algorithm performs homogeneously to Q-learning as with each iteration, or step, the algorithm executes the highest Q-valued action which is precisely the greedy policy as explained earlier in Q-learning.

The PHC algorithm is only able to locate an optimal policy against stationary strategy opponents and has never been proven to converge if pitted against non-stationary strategy opponents [14]. As according to Q, that converges at some Q', the mixed strategy π is greedy and will therefore converge to the best possible policy in response to the environment the agent is placed in.

4.3 Agent Design and Implementation

4.3.1 Design of States and Actions

As the learning involves interaction with the environment, it is essential to define the state space and the action space. A state space is a set of states, represented by vectors of observations, that an agent could possibly experience throughout its life. An action space is a set of actions that an agent could possibly perform in the game. The design of the state space and action space is tailored to the task assigned, namely maximising both individual and collective utilities.

Design of State Space

Firstly, we identified what observations are needed to determine the condition of the agent and the environment which is represented by the Tower and other agents. Considering the given task, we chose to observe four variables:

1. Our agent's HP
2. Amount of food currently on the platform
3. The number of days in critical condition

4. The HP of the neighbour that is on the floor below our agent

The first three observations served to fulfil the first aim of the task, as they could effectively reflect the health condition of our agent. The last observation was used to assess the impact of the action of our agent on the neighbouring agent, which is useful for the second aim of the task. Subsequently, we found the resolution of each observation by trial and error. Having a high resolution, the agent would be able to perceive tiny changes within the environment at the cost of increased number of states, and thus slowing the learning process. Having a low resolution, the agent would not be able to adapt to changes in the environment effectively and promptly. Finally, the state space of our agent is defined by ten state intervals for agent's HP (0 – 9, 10 – 19, . . . , 80 – 89, 90 – Max HP), ten state intervals for the food on platform (0 – 9, 10 – 19, . . . , 80 – 89, 90 – Max food), four states for Days at critical (0, 1, 2, 3), and eleven state intervals for neighbouring agent's HP (-1, 0 – 9, . . . , 80 – 89, 90 – Max HP), with an additional unknown state represented by –1, as the neighbouring agent might refuse to share such information. Table below illustrates the state space definition.

State Number	HP	Food on platform	Days at critical	Neighbour's HP
0	0 – 9	0 – 9	0	-1
1	0 – 9	0 – 9	0	0 – 9
2	0 – 9	0 – 9	0	10 – 19
...				
11	0 – 9	0 – 9	1	-1
12	0 – 9	0 – 9	1	0 – 9
...				
44	0 – 9	10 – 19	0	-1
45	0 – 9	10 – 19	0	0 – 9
...				
440	10 – 19	0 – 9	0	-1
441	10 – 19	0 – 9	0	0 – 9
...				
4399	90 – Max HP	90 – Max Food	3	90 – Max HP

Table 4.1: State space definition.

Implementation of State Space

The state space is implemented as a 4D slice, with each dimension corresponding to columns 2-4 in Table 1. The 4D slice is then iterated through with each state being assigned an incrementing number. The state number of the agent can then be checked by accessing the 4D slice with the corresponding indices.

```
func (a *CustomAgent2) CheckState() int {
    hp := CategoriseObs(a.HP())
    currFood := CategoriseObs(int(a.CurrPlatFood())
    critDays := a.DaysAtCritical()
    nHP := CategoriseObs(a.neighbourHP)
    return a.stateSpace[hp][currFood][critDays][nHP]
}
```

Action number	Intended food intake
0	5
1	10
2	15
3	20
4	25
5	30

Table 4.2: Action space definition.

Design of Action Space

In the game, taking food is the main interaction with the Tower, which provides feedback according to the amount of food taken. Thus, we divided the action of taking food into six distinct actions with different number of intended food intake, as shown in Table 4.2.

We excluded the possibility of taking more than 30 food units, as eating above this level will only bring punishment to our agent according to our reward design. Note that the intended food intakes are multiple of 5, as we would like the agent to have less actions to learn, resulting in faster convergence.

Implementation of Action Space

The action space is implemented as a 2D slice. It is iterated through and assigned the appropriate food intake according to Table 4.2.

```
for i := 1; i < actionDim; i++ {
    actionSpace[i] = actionSpace[i-1] + 5
}
```

Design of Reward Function

The reward function provides a way for us to set the desired behaviour of our agent. The reward function will assess the quality of a particular action under a specific state and provide feedback, either reward or punishment, to the agent. Initially, we have designed four desired behaviours. We encourage our agent to survive, not to overeat, and to save neighbour: if the neighbouring agent is in critical state, the agent will be punished. However, the last desired behaviour was only enforced by the simple communication of "Ask HP" with the agent that is one floor below. Indeed, as shown in Table 4.3, implementing a simple communication did not improve the agent's performance on saving other agent, i.e., the overall death count remained the same after adding the simple communication and tuned reward function accordingly. A potential reason might be that the agent cannot learn to save with such limited information. As such, in our subsequent experiments, the simple communication was not considered, and the primary focus was placed on the first two desired behaviours.

		Without communication	With simple communication
Cumulative death in 500 days	Homogeneous tower (14 team 2 agent)	50	78
	Heterogeneous tower (2 agents per team)	133	139

Table 4.3: Effect of simple communication on cumulative death count in 500 days.

Implementation of Reward Function

The final reward is the sum of survive bonus, eating bonus, wasting bonus and saving bonus. The individual bonuses will be explained now.

Survive bonus The survive bonus is implemented with two conditions.

Algorithm 2: Survive bonus algorithm

```
if Agent is in critical state then
    surviveBonus ← surviveBonus +1;
else
    surviveBonus ← surviveBonus −(3.0 × daysAtCriticalHealth);
if Agent was in critical state before action then
    surviveBonus ← surviveBonus +(5.0 × daysAtCriticalHealth);
```

The first condition encourages actions that keep the agent in a non-critical HP state. The second condition encourages actions that take the agent out of a critical HP state (if it was in one).

Eating bonus The eating bonus is implemented by checking if the agent is in a critical condition and encourage it to eat more if it is.

Algorithm 3: Eating bonus algorithm

```
if Agent is in critical state then
    eatingBonus ← eatingBonus +(0.01 × amountOfFoodTaken);
```

Wasting bonus The wasting bonus encourages the agent to only take the amount of food that is necessary to reach maximum HP. We penalise the agent if it wants to waste food and we also penalise the agent if it does waste food. The wasting bonus is implemented with two statements.

Algorithm 4: Wasting bonus algorithm

```
wastingBonus ← 0.2 × (expectedHPIncreaseWithIntendedFood − actualHPIncrease)
```

Saving bonus The saving bonus encourages the agent to perform actions that result in the agent in the floor below being in a nominal state. This is implemented with the condition which checks the agent below's state.

Days	Overeating threshold	Overeating	Critical	Total Death:	Team2 Death per agent	Mean Age
5000	80	0.2	10	786	8	88.14
5000	80	0.2	5	863	7	72.94
5000	80	0.2	1.5	876	10	63.44
5000	80	0.2	3	844	7	73.11
5000	80	0.2	15	913	6	71.63
5000	70	0.2	10	797	12	71.61
5000	60	0.2	10	830	13	82.71
5000	80	0.5	10	859	9	80.48
5000	80	1	1	865	6	69.33

Table 4.4: Simulation results used for hyperparameter tuning.

Algorithm 5: Saving bonus algorithm

```
if Agent on floor below is in critical state then
    savingBonus ← savingBonus −3;
else
    savingBonus ← savingBonus +1;
```

4.3.2 Simple Reward Function

Based on the health decay model defined in 2.4, the agent reward was designed to encourage the agent to reward a health above critical and punish dropping to critical health as well as punish overeating using a reward point value for each day. The reward function with its tuned hyperparameters is as such:

- Set reward to zero for each new day
- If HP is 80-100 (Punish)
  - Reward == $0.2 \times$ food taken that day
- If HP is 4-79 (Reward)
  - Reward = 3
- If hp is critical (Punish)
  - Reward=$10 \times$ Consecutive days agent has stayed at critical

When tuning the hyper parameters in the reward function appropriately, the main factors for performance tuning were cumulative total tower death count, average agent life span, and number of Team 2 agent deaths.

1. Tuning the overeat threshold down meant fewer overall deaths, but more team2 agent deaths, a rough optimal value was found at 80 hp. Setting a higher threshold would have the opposite effect
2. Tuning the overeat punishment parameter seemed to have little impact on agent behaviour as the agent would understand punishment regardless of severity and try to stay below the overeat threshold. An optimal value was found at 0.2*(food taken that day)

3. For tuning the critical health status punishment parameter, if agent was punished less for having consecutively critical hp meant fewer overall deaths, but more team2 deaths. Setting a stronger punishment for this gave the opposite effect. The value for this parameter giving best performance on average was at 10*(days agent has stayed at critical consecutively)

The values found for tuning are only estimates of what gives best agent performance in synergy with the tower. This is because the tower introduces a very strong randomness effect as it reshuffles the agent floor locations after only one day. This means almost no run will ever be the same and highly unlucky events, with for example no food for several days, are very likely to occur.

4.4 Results and Analysis

4.4.1 Aim of the Simulations

The aim of these demonstrations, as outlined throughout the report, was to design and produce an agent that could display the survivability of a reinforcement learning agent. The agent was intended to attempt to strike a balance between prioritising individual utility through survival, and collective utility through sustainable behaviour with resources. This was to be done with the principle of Teleo-Reactivity in mind, whereby the agent must take actions to pursue the goal of survival by dynamically adapting to the environment using perceived information. As such, for the demonstration to be considered successful, the final design of the agent should display how it learns through reinforcement learning, with the core strategies combined with Q-learning and Hill Climbing and the environmental response, to survive in the Tower along with other agents. This includes utilising the design of states for q-table and reward formula, as well as demonstrating taking a variety of actions before died. This in turn would culminate into a converge reinforcement learning agent. This section aims to highlight how the agent achieves each of the environments by going through the survivability of the reinforcement learning agent in the environment with different hyperparameters.

The performance of our agent will be justified by metric relating to both individual and collective utility: reward by day, cumulative reward by day, HP vs day, and food taken per day (HP), and age lived per generation.

4.4.2 The Performance of our Agent

As mentioned, the performance of our agent will be measured with metrics involving reward, HP, food taken, and age. For a reinforcement learning agent, cumulative reward is often used to judge a reinforcement learning algorithm. The reward that the agent receives while acting and learning tells how well the algorithm performed while being deployed. On the other hand, the plot for HP vs days describes the survivability in the environment (Tower in this situation).

Convergence

Figure 4.1 is the result of a simulation for 5000 days without random agents. Reincarnation is allowed in this simulation.

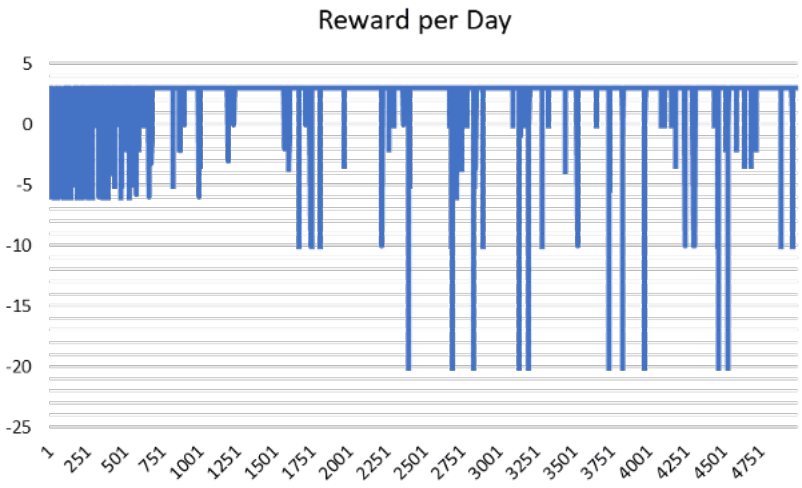


Figure 4.1: Reward per day.

Figure 4.2: Cumulative reward against days.

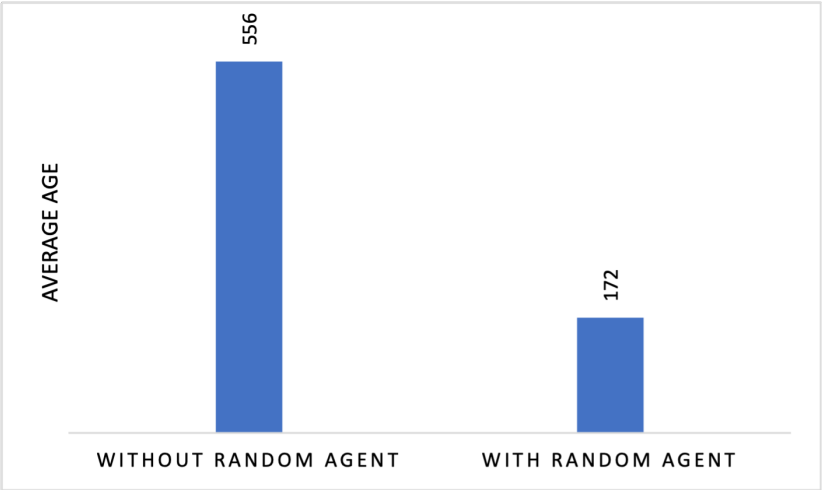


Figure 4.3: The average age of Team 2 Agent.

It can be observed that during approximately the first 500 days, the agent consecutively gets punished and the reward fluctuates frequently. Then, it starts to find the way to cater to the design of reward function, and despite the instability caused by randomness in the tower, the reward function tends to be smoother, as the changes in reward function occur less frequently.

In addition, in Figure 4.2, we can find that the cumulative reward increases at a nearly steady speed after fluctuation. The turning point is approximately 500 days as well.

Thus, it can be concluded that under the configuration of this experiment, the policy of the agent is able to converge at around 500 simulation days. This means that the agent manages to find the best policy to maximise the reward he can get and persist this policy to consecutively achieve better rewards. Within its context and ability, it has achieved the Pareto optimal level for individual utility.

The verification of convergence is fundamental to the presentation of our results. All the results introduced next are based on our successful implementation of the learning algorithm.

4.4.3 Random Agent

The effect of having random agents in the tower to our agent is discussed in the following. The random agent takes food randomly with the range between 1 till max food available on the platform. This strategy can result in the random agent overeating, which will reduce collective utility. This is due to total food allocation equal to (FoodPerAgentRatio) $*$ (number of agent) while the random agent will intend to eat between 1 and total food allocation.

With the inclusion of the random agent(s) in the tower, the average age of our agent is observably decreased from 556 days to 172 days in Figure 4.3. This demonstrates the direct implication of collective utility on an individual agent; regardless of individual strategies, a rogue agent which does not keep the macro-level goal of collective utility will hamper the individual utilities of the other agents.

The results in Figure 4.4 and Figure 4.5 are from a reincarnation-allowed simulation with only one agent of our team. In Figure 4.4, our agent with 2 random agents performs better than without random agent before 2500 days. However, when comparing with the plot of reward by days (Figure 4.5), our agent with 2 random agents has a greater total reward penalty. This

Figure 4.4: Cumulative reward by days.

Figure 4.5: Reward by days.

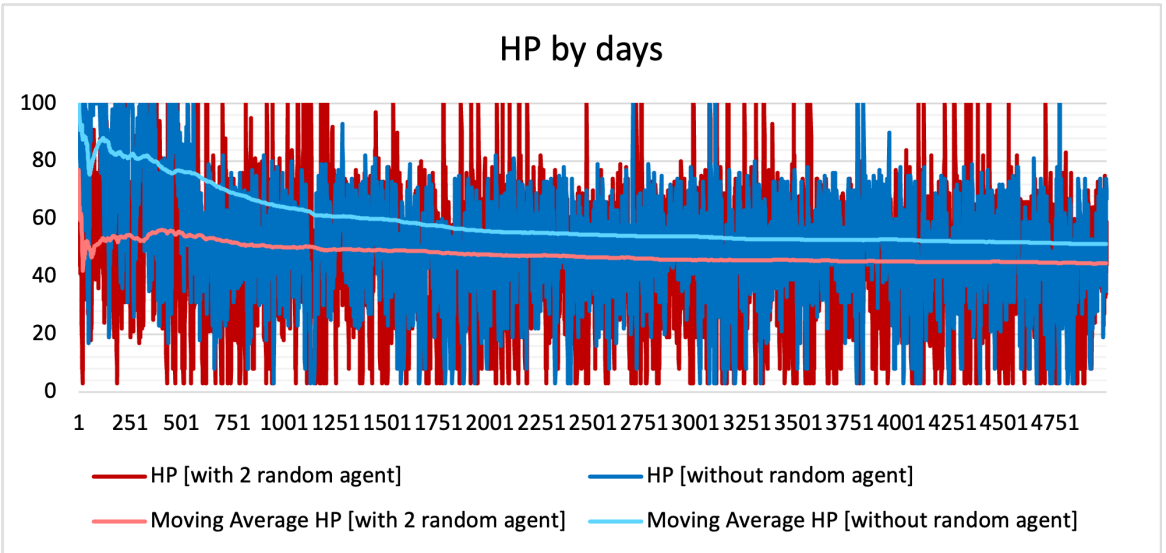


Figure 4.6: The change of agent HP by days.

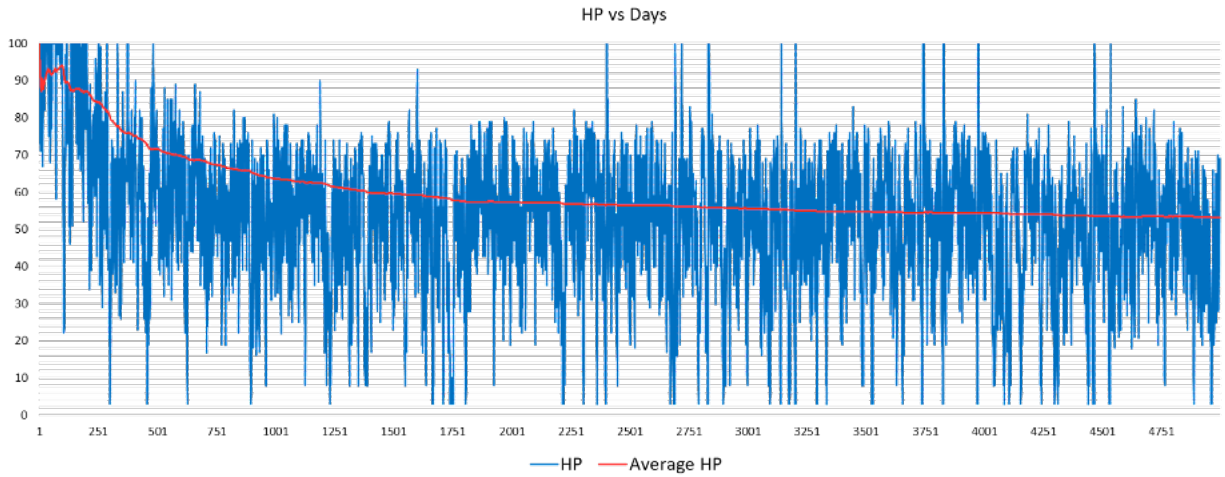


Figure 4.7: Average HP against days.

is due to the randomness introduced by the random agent, causing the learning process of our agent is insufficient. On the other hand, without implementing the random agent in the tower, our agent able to learn after being deployed.

When there are random agents in the tower, our agent tends to overeat more frequent to confront the effect from the random agent in Table 4.6. However, the moving average HP of our agent is lower compared to without random agent.

To conclude, the random strategy of the random agent detriments the learning process of our reinforcement learning agent. The main goal for our agent is to change its strategy according to the environment to survive, the randomness effect from random agent cause confusion for our agent as the reward may be different for the same action our agent has taken at the same state. This leads to a poor learning environment for our agent and removing random agent during the learning process of our reinforcement agent is suggested.

4.4.4 Average HP

The results in Figure 4.7 are from an experiment without random agents. Only one agent of our team is included, and reincarnation is allowed.

Following the design in the reward function, the HP of our agent is learnt to retain under 80. It does not choose to stay as close as possible to the upper HP limit 80. Instead, the agent decides to maintain the HP around 55 on average.

This may indicate that staying at a high HP level may not be a good surviving strategy in the tower.

4.4.5 Reincarnation of Agent

The results of this part are from experiments without random agents and only one agent for our team is involved.

With reincarnation, the agent will not be able to lose its memory or experience after each death, which is a basic factor for a learning algorithm. In our hill-climbing Q-learning, the reincarnation

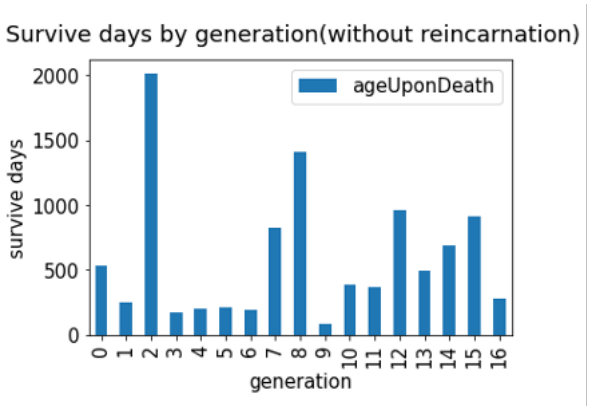


Figure 4.8: Surviving days of each live without reincarnation.

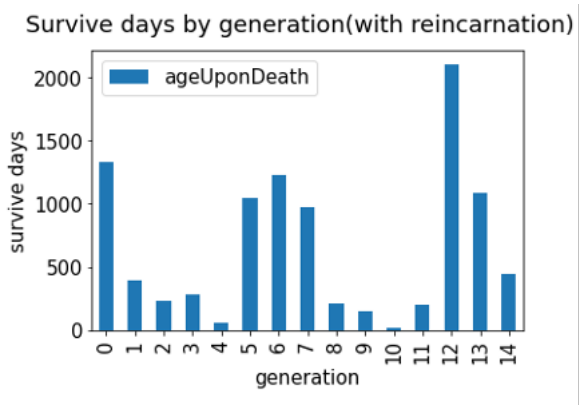


Figure 4.9: Surviving days of each live with reincarnation.

allows for the retention of the Q table and policy table when the agent dies.

From the comparison of Figure 4.1 and Figure 4.2, it seems that there is no difference between the agent with and without reincarnation in terms of the total death and surviving days. The agent without reincarnation can live as long as the agent with reincarnation although it cannot inherit. This is partly because that, due to our effort in design to minimise the model complexity and maximise the convergence speed, our agent learns fast and normally it will converge within 200 days. Thus, if the agent can avoid death in the first 200 days, which is of high probability, it can live for a long time with the learnt strategy. Its policy will be self-optimised consistently as long as he is alive.

This can be further explained by examining the Figure 4.3 graph that specifically shows the 13th life of the non-reincarnating Team 2 agent from Figure 4.1 Here the agent although not reincarnated from a previous life still shows a cumulative reward curve that eventually shows a linear increase, meaning it has found a near optimal policy through dynamically adapting fast enough.

In addition, the loss of reincarnation still casts some instability. From Figure 4.4, we witness that the HP of the agent fluctuates more heavily if it cannot reincarnate. This may result from the fact that the most unstable duration for the Q-learning algorithm is the learning period. If the agent cannot reincarnate, it has to experience the learning period again every time it dies which will definitely introduce more unstable factors.

Enabling reincarnation can let the agent pass its learnt knowledge to its child, while disabling

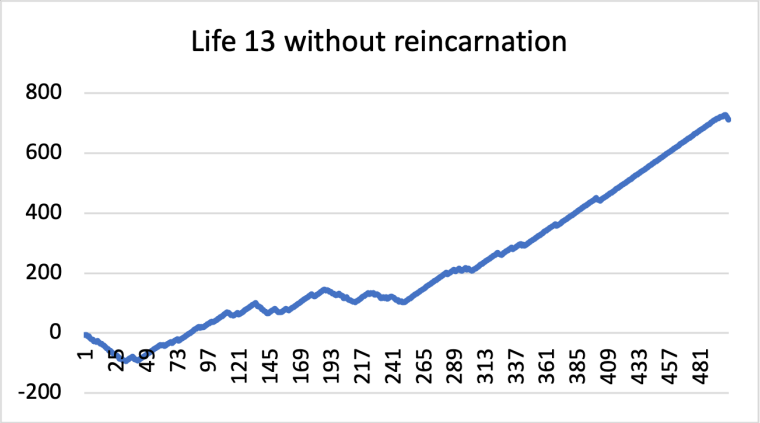


Figure 4.10: Cumulative reward of 13th Team 2 agent life without reincarnation from the previous figure.

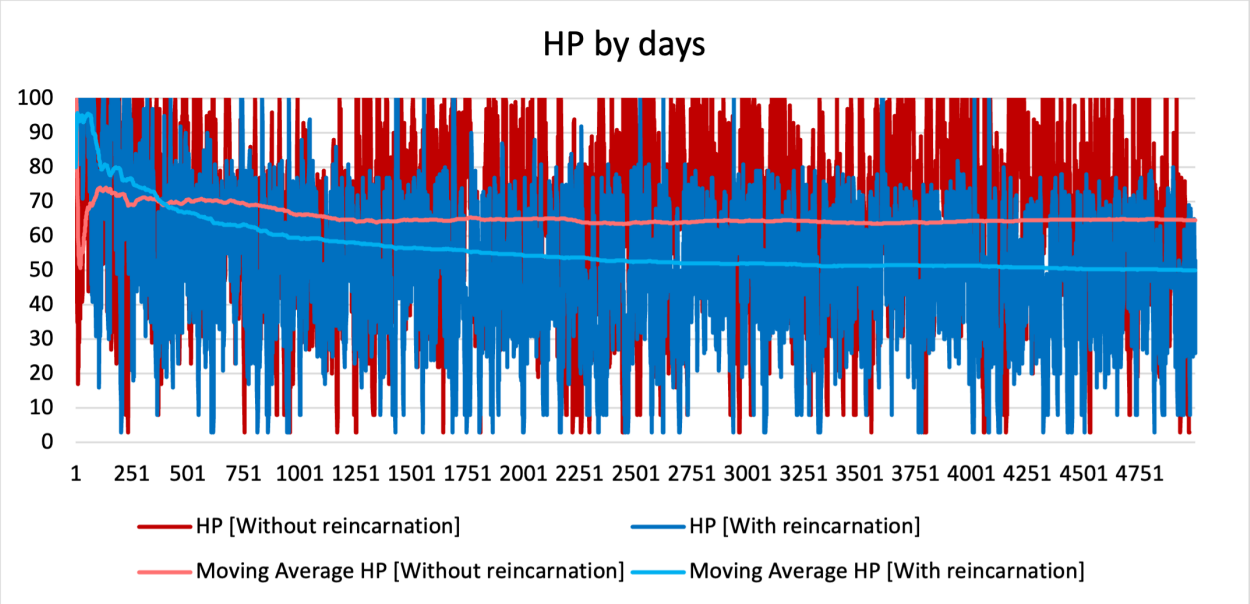


Figure 4.11: The change of HP by days with and without reincarnation.

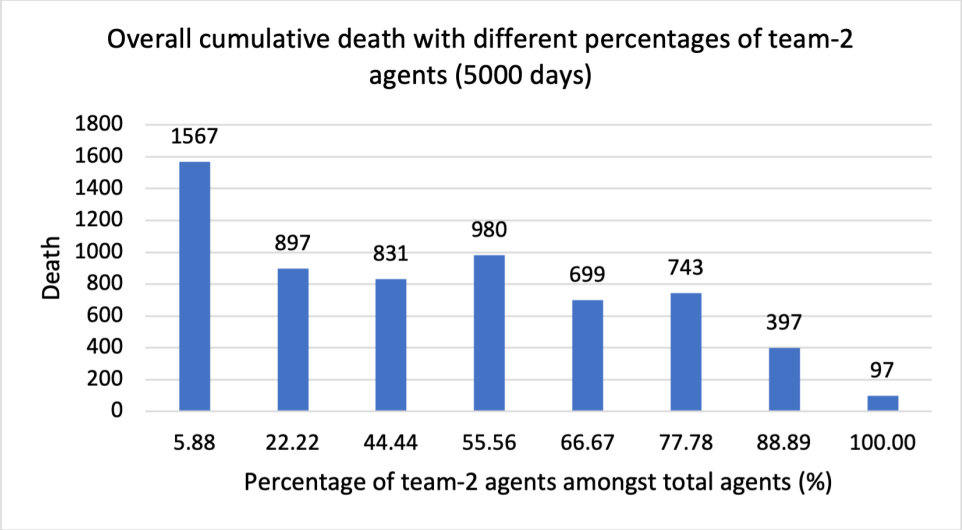


Figure 4.12: 1 Shows the cumulative deaths of a towers with varying percentage of total agents being Team 2 agents. Total agent count is 18 agents.

the reincarnation will lead to a tumultuous relearning process for every generation. Enabling the reincarnation of our agent is important as it let our agent learn from the reward and inherit the experience from its predecessor.

As a conclusion to reincarnation, the most important impact of reincarnating the Team 2 agent is not an increase in its own performance as it learning rate is fast enough for the average life duration. The real trade off with reincarnation is the effect it has on the tower's other agents. The average HP is lower with reincarnation as can be seen in Figure 4.11, and so the other agents benefit from this much more. As previously discussed, maintaining the Pareto optimal individual utility will allow for an increase in each other agent's individual utilities, and thus the collective utility.

4.4.6 Self-Organising Ability

A tower that can self-organise has reached an equilibrium where agents no longer die and instead are able to survive all together based on their adaption to each other and the environment. As discussed in Section 4.1.4 this generally would occur when all the agent behaviours have assimilated and converged to become one and the same. Thus, it is in our interests to test a scenario with a tower full of the same agent type. To test Team 2 agents' own self-organising ability, towers with different number of Team 2 agents present where tested.

The results in Figure 4.12 shows that the Team 2 agent does indeed have a huge impact on the towers performance. The more Team 2 agents present in the tower the fewer deaths the tower would have. But with this reading alone its not possible to determine if the Team 2 agents actually has a self-organising ability as there is no way of knowing when the last agent died in each run. Figure 4.13 on the other hand shows how a tower consisting only of Team 2 agents eventually reached an equilibrium point where all agents stayed alive. As such the design of the Team 2 agent showed self-organising ability with its Q-learning PHC reinforcement learning strategy.

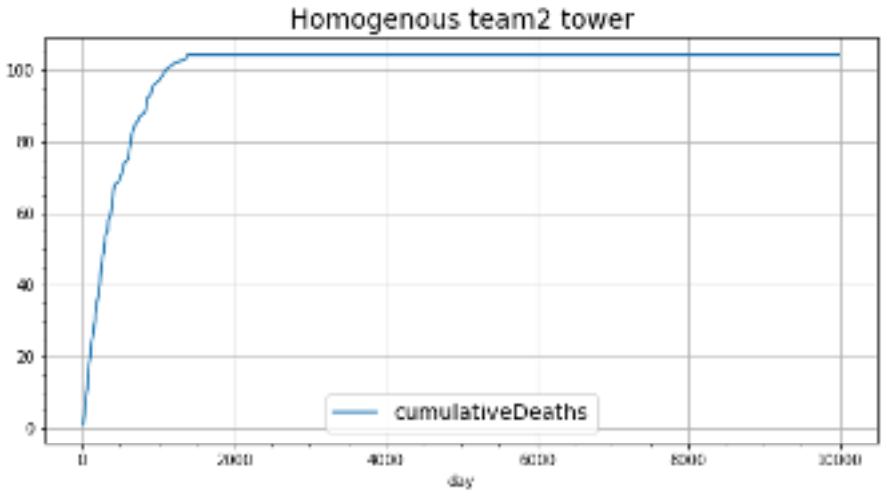


Figure 4.13: tower of only Team 2 agents reaches a point where all agents stays alive.

HP	Average Food decided to take
0-10	27.3

Table 4.5: Policy of agent when HP is 0-10.

4.4.7 Learnt Policies

Table 4.5 shows that when the HP of agent is low, the agent tends to take in as much food as he can to survive, as it must prioritise its individual survival and cannot consider other lofty altruistic goals.

From Table 4.6, it is prominent that the agent is still in a dangerous HP range. When there is not much food on the platform, the agent decides to eat more to survive. When the food on the platform is 20-30 which is still not safely sufficient, even though its HP is not high enough, the agent seems want to leave food for others. The agent probably considers that in such case, leaving food for others can be a better choice because if some agents fall into the critical level, they may need more food to reach the weak level than what they need now. Perhaps it has developed some form of empathy, where it perceives that other agents may be in similar if not more dire situations.

When the food is abundant on the platform at this HP level, the agent takes in more food. This might be a strategy to prevent being allocated to lower floors on the next day since its HP is still near critical level.

Table 4.7 shows the decisions of the agent when HP is in 20-30 level. It is observed that its policy at this level is quite similar to that at HP 10-20. It still chooses to ensure the survival of

HP	Food on the platform	Food decided to take
10-20	0-20	13.2
10-20	20-30	7.1
10-20	30-100	23.9

Table 4.6: Policy of agent when HP is 10-20.

HP	Food on the platform	Food decided to take
20-30	0-20	16.1
20-30	20-40	6.1
20-30	30-50	21.0
20-30	50-100	5.0

Table 4.7: Policy of agent when HP is 20-30.

HP	Food decided to take
30-100	5.0

Table 4.8: Policy of agent when HP is 30-100.

itself when the food is not enough. When the food is above the level of scarce, it still decides to eat less and leave the food for others. In addition, when there is 30-50 food on the platform, it decides to eat more again, but when the food on the platform is more than this it chooses to barely eat. It seems that from the experience of agent in HP 20-30, 30-50 food on the platform can be a dangerous indication that it may starve or lose HP in the next few days, so he tends to supply himself for the future. And the agent considers it unnecessary to eat much when there is abundant food on the platform, due to the reward function penalties.

Table 4.8 shows that when the HP of agent is larger than 30, it tends to eat the minimum amount of food and leave the food to others.

In summary, from the observations gathered, the learnt policy of the agent follows certain patterns:

- The priority is the survival of itself; the tower scenario is often too unpredictable to switch to more altruistic objectives.
- Leaving food for others when the food is not sufficient tends to bring more collective benefit in the long term.
- Maintaining HP over 30 is safer for the survival of agent and is more likely to result in a Pareto optimal value for individual utility.

Chapter 5

Team 3 Agent Design

5.1 The Agent

Team 3's agent design is based on the Theory of Reasoned Action (TRA) and its extension, the Theory of Planned Behaviour (TPB). The agents are designed to have a shiftable personality dependent on the three variables in Figure 5.1 that TPB [15] links to the behaviours of an agent. These three variables, together with a social network, define its personality as a whole, [16] and influence the actions it will take.

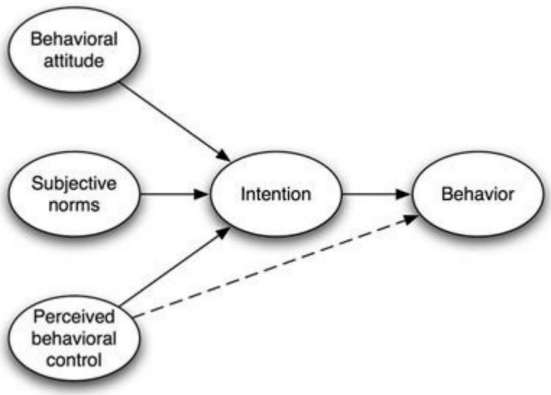


Figure 5.1: TRA explanation.

The Theory of Reasoned Action was developed by Martin Fishbein in 1967 and was later improved by Icek Ajzen and Martin in 1980, with the Theory of Planned Behaviour. The TPB states that the behavioural intentions of individuals can be attributed to three interdependent factors: perceived behaviour control, the subjective norm, and the attitude of the individual, as shown in Figure 5.1. It must be noted that the TPB is unable to accurately and consistently predict non-volitional behaviours of the individual. Therefore, TPB can only be applied to those actions for which a reason can be found. These are actions that occur as a reaction to others and that the individual has control over [17]. This theory is applicable to agent decisions in the tower since the agent must make decisions in a reactionary manner, based on interactions with other agents.

In order to apply the TPB to the agents, we define three agent variables, each corresponding to one of the three components of the theory. Using this approach, we have defined the agent

variables to be:

1. **Stubbornness**: it addresses perceived behaviour control, which is the influence that external agents have on the decisions that are to be taken.
2. **Morality**: it addresses the subjective norm, which is one's idea of what is socially perceived to be right or wrong.
3. **Mood**: it defines the attitude towards the action to take.

These variables are modified as an agent interacts, which leads to our agent having mutable behavioural intentions, and making decisions with different "intended outcomes" as the variables change.

In literature surrounding the substantive uniqueness of human behaviour [17] Martin Fishbein argues that no two agents will react identically to the same circumstances. In practice, reactions tend to be normally distributed, with a random aspect to them. To account for this, all possible changes to internal agent variables are implemented by specifying the range within which the variable can change. The final change to the variable is randomly selected from within this range. This characteristic of randomness in the agents' behaviour compensates for the fact that the TPB does not take into account non-volitional behaviours, so including this randomness makes the agent act in a more human-like manner.

The next step after identifying the agent strategy was to think about the scenario and how the problem should be solved. Our first approach used classical game theory, which states that under normal circumstances, an unbiased agent would always prioritise themselves and their needs over anything else. Hence, this dictates that each agent should not act in a moral way at all. In our scenario, this leads to agents taking very large amounts of food, causing agents below them to have access to less food, and have a greater likelihood of death. However, we found research demonstrating how humans in a social environment tend to make decisions benefiting the collective [18]. This contradicted traditional game theory views and made this approach unreasonable when trying to recreate a human type behaviour, following the two most common explanations of why humans violate classic game theory: enlightened self-interest and group identity.

Enlightened self-interest includes a human's awareness of long-term consequences, along with their desire to avoid social punishment and self-punishment. Assuming a group identity is the act of regarding the goals of the collective as one's own goals. In the tower, agents with a group identity would see their own primary purpose as ensuring that all agents in the tower survive, and not that it has survived as an individual. As such, it feels like its social network is an extension of itself and care for it greatly. According to research, the above factors are sufficient to make humans behave in a different manner within a collective than classic game theory would dictate.

With this in mind, we further extended our definition of the agent variables and changed the way our agent would use them. In addition, we implemented a social network by defining an additional "friendship" variable, to aid the agent in assuming the aforementioned group identity.

1. **Stubbornness**: Defines the willingness of the agent to read the messages it receives, and in doing so, form relationships. Due to this, it is also more or less willing to succumb to peer pressure. In general, stubbornness can be summarised as the willingness to change views based on possible long-term consequences. This is reflected in how the agent reacts to

different messages, on the basis of enlightened self-interest and not wanting to be punished for their behaviour.

2. **Mood**: An agent's willingness to consider the group identity at a given time. The agent's decisions will benefit the group more the better the mood they are in. In the tower this is translated as eating less food when possible, aiming for the preservation of the group.

3. **Friendship**: The degree to which another agent believes an agent has acted in its own self-interest. This defines the degree of social punishment or praise that an agent receives. This will naturally divide an agent's perspective of other agents into those who act in their own self-interest and those who do not. The agent interacts with these two perceived categories of agent differently.

4. **Morality**: A measure of susceptibleness to guilt, which is a form of self-punishment. On top of this, morality also represents the degree to which agents are conscious of the group identity. Knowing that other agents are suffering or dying, especially ones that belong to the agent's social network, will make the agent feel guilty and more willing to help others if their morality is high.

The agent will interact and make decisions based on a combination of these variables, as defined above. More information is given in the sections below, which explains different aspects of the agent, such as their daily actions, memory, decision taking and communication.

5.1.1 Action Process

The Agent takes the following steps during their day:

1. The agent starts by updating its defining variables at the beginning of each day based on their HP. This acts as the agents checking in on itself and changing its attitude(mood) and morality depending on its condition(HP).

2. If the agent was just 'born' it will estimate an initial value for the average food consumed based on the amount of food required to maintain, 50 health.

3. When the tower reshuffles the agents (which each agent knows by checking what floor it is on) mood changes depending on the relative position of the new floor to the prior floor.

4. The agent attempts to eat every day. The quantity depends on treaties, the messages it has received and whether other agents have requested it to leave or eat a certain amount of food or a calculation based on the agent's current values. More information on this can be found in Section 5.3.

5. Finally, the agent will attempt to communicate with other agents and create a social network via message sending and treaty proposition. More information on this is can be found in Section 5.2.

5.1.2 Agent Knowledge

Agent 3 will be able to make decisions when receiving messages or after signing treaties and remember these until it eats. This information is used to show that a predetermined decision has been taken instead of "impulsively" eating food depending solely on the agent's hunger and state of mind. The decisions that the agent takes include:

1. The floors the agent has been in before. This information will affect the agent's mood after being reshuffled, as it would remember where they have been and know if the new floor is a good or bad one.
2. The last recorded HP value. Used to see changes to the agent's health.
3. The agent's friendship network. It is aware of the people it has met during their time in the tower. The agent's friends are identified and are considered more or less close to the agent with an affection parameter that ranges from 0 to 1, with 0 representing a complete dislike for the other agent.
4. The amount of food last eaten.
5. The amount of food last seen on the platform.
6. A moving average of food consumed.
7. The age of the agent.
8. The treaty we have proposed until it is accepted or rejected.
9. The estimation of the reshuffle period.
10. The HP our neighbour above last told us it had.
11. The HP our neighbour below last told us it had.

5.1.3 Agent Generation

The defining variables of the agents are randomised when it is generated in order to avoid preconditioned agents and mimic different personalities. All three variables are defined to be in the range 0 to 100. Morality and mood are randomly allocated to a value in the range, but stubbornness is limited to a number between 0 and 75. As explained in our agent parameters, stubbornness represents the effect of external influences on an agent. A stubbornness of 100 would mean this agent is completely unwilling to listen and would ignore all incoming messages, effectively making them an agent that disrupts the tower. In order for this to not be the case, we added the artificial upper bound to the value of stubbornness.

As Shao et al. have shown in "Beyond Moral Reasoning: A Review of Moral Identity Research and Its Implications for Business Ethics" [19], one's moral action is most heavily influenced by their moral personality, as opposed to their moral reasoning, which empirically seems to show only a minor influence on the moral behaviour of an agent [20]. Hence, we used the variance and unpredictability to initialise our agents randomly on the moral scale. This insight also allowed us to witness a purely emergent strategy and nature as opposed to a pre-programmed inherent morality.

While the implementation of initial randomness alone allowed for interesting strategies to occur, we also wanted the agents to adapt their morality on the basis of their community and associations, as argued by Shao. This was made possible by having all actions an agent does affect its emotion variables.

5.2 Agent Communication

Communication between agents is fundamental for the possibility of collaboration and self-organisation between agents. Crowd behaviour is considered to be generated by individuals, context-dependent and dynamic, as defended by the modern foundation of crowd behaviour [21]. In this project, communication between agents is based on two concepts, treaties and direct messages, which are equally exploited by the Team 3 agent and combined with the conceptualisation of friendship levels.

5.2.1 The Agent's Social Network

The agent forms relationships with others surrounding it as they communicate with each other. The agent remembers everyone it has met and assigns a level of friendship to them depending on their interactions. By doing this, the agent forms a social network that will affect its decisions, something that the Theory of Planned Behaviour explores in the "subjective norm" aspect of the theory. In the TPB, all three main variables are interdependent and for this reason, the agent will have a better or worse opinion of someone it has just met depending on their morality at that point.

The large importance which the agent places on friendship is not only a logical interpretation of the TPB. In fact, it is a notion which philosophers and legislators have repeatedly confirmed throughout the ages. For example, the Aristotelian concept of friendship defends it as "the finest way we exercise practical reasoning". Quoting Aristotle: "Friendship seems to hold cities together, and lawgivers seem to care for it more than for justice ... and when men are friends, they have no need for justice", friendship is considered the best form of human moral action by The Nicomachean Ethics [22]. Thus, it is no surprise that friendship is fundamental in the decision-making process of the agent, and also in its responses to communication initiated by other agents.

To provide an alternative perspective on friendship, McCullough et al. investigate the nature of prosocial action and gratitude's place in transactions [23]. The friendship function allows the agent to assign a level of gratitude in response to the prosocial action of the other agents. In this way, the agent could more effectively alert themselves to the prosocial nature of another's actions. By utilising the friendship function to, in turn, behave more prosocially to those who have done so to us, we embedded a system of gratitude in the agents' actions. This allows the agents to be more capable of self-organising.

We further developed the agent's response to actions taken by foreign agents by evaluating the selflessness of the action taken. That is to say, the more severe the sacrifice another agent makes, the more grateful the agent becomes. This is based on McCullough's definition that one experiences gratitude when "they have benefited from someone's costly, intentional, voluntary effort on their behalf".

Emotions allow us to respond to reality in a self-protective and self-enhancing way [24], it was this response that we wanted the agent to embody so that emotions could govern the evolution of their actions, by either leading them on a self-protective path, if circumstances caused selfish agents to prevail, or a self-enhancing manner, as cooperation would beget further cooperation through the agents' system of gratitude.

This is the basis of our friendship metric and implies the agent will react more positively to a friend and will increase the friendship of those with whom it has positive interactions.

5.2.2 Message Sending

Message sending is determined by a probabilistic decision, with each message having a 20 percent chance of being sent. The direction the messages are sent in is determined by the moral state of the agent. When the agent's morality is low, they will tend towards self-preservation and send messages upwards requesting information or help. On the other hand, when morality is high, the agent will look to create a stable society as stated by their willingness to look ahead and communicate with those below them to offer help and information, aiming to fortify their social network.

The design of message passing has various noteworthy characteristics that act to aid the agent fulfil its intentions:

1. The agent prioritises itself when its HP is critical, opting for treaties and messages that involve other agents leaving food in the platform and utilising this mechanic as a failsafe when in danger.

2. The agents take into account the neighbours' health when sending messages proposing how much food to leave on the platform. This makes the message more likely to be accepted.

5.2.3 Message Reception

When messages are involved, stubbornness is used to decide to ignore or read a message, with its reaction or answer also being dependent on the stubbornness of the agent.

The agent will follow the next process when a message is received:

1. The agent will decide to ignore or read the message, with a probability of ignoring the message equal to the stubbornness level.

2. If the agent decides to read the message it is analysed and matched with the type of response or responses acceptable for the message.

3. The agent will note the agent that sent the message and consider the friendship level it shares with the sender. Friendship together with the morality level and the content of the message itself will decide the exact content of the response.

4. The content of the message and its sender will affect the agents' emotional variables and the friendship level with the sender. This change affects the future reaction to communication done by other agents and the decisions taken by the agent from that point onwards.

To ensure a minimum level of communication all messages read will be answered by the agent.

5.2.4 Reaction to Received Messages

The changes are dependent on four criteria to determine initial agent characteristic manipulation. These four variables are the variables present upon the receipt of the message and somewhat representative of the type 1 brain system discussed in [25], [26].

The initial emotional response is based on the variables of the immediate past, the hp moving average on the last five ticks, opposed to the overall hp average, or welfare average. This is to

represent the subconscious layer of emotional response, as seen in [27], whereby our response to the situation is more irrational and illogical than the conscious mind perceives.

Summing the net total of all the initial responses to zero, allows us to explore the oscillatory nature of emotion while avoiding the possible pitfall of any dominant emotions.

HP dec last tick [nonfriend]	Stubbornness	Morality	Mood	Friendship
ask Food Taken	0	-0.5	-0.5	0
ask HP	0	-0.5	-0.5	0
ask Intended Food Intake	0	0	0	0
Request Take Food	1	-0.5	-1	-1
Request Leave Food	1	-0.5	-1	0

HP inc last tick [nonfriend]	Stubbornness	Morality	Mood	Friendship
ask Food Taken	0	0	0	-1
ask HP	-1	1	1.5	1
ask Intended Food Intake	0	0	0	0
Request Take Food	0	0	0	0
Request Leave Food	-1	1	1.5	1

Sum total	Stubbornness	Morality	Mood	Friendship
Request Leave Food	0	0	0	0

Table 5.1: Message Reactions.

This zero-sum agent is how the agent is currently set, it represents the neutral agent. One who possesses no inherent emotional bias. This was done by balancing the positive and negative reactions to a friend, and those to a non-friend separately. We noticed the need for this when the agent would initially get stuck at the positive and negative limits of certain emotions. As future work, it would be exciting to see the emergent strategies which we hope to observe as we introduce biases. For instance, an agent whose net sum of all interactions leaves them with a stubbornness increase, coupled with an inherently moral agent, may create strategies yet thought of.

5.2.5 Proposing a Treaty

When the agent HP is critical, it will ask for help by either asking for the agent above to leave at least the amount of food necessary to survive or, if it doesn't have any active treaties, a treaty that states: if HP is higher than 20 leave 95% of the food in the platform for 3 days. At this time the agent's self-preservation instinct takes precedence and its objective becomes to eat enough food to get out of the critical state before the agent dies. When the agent is not in critical condition, it is able to propose other treaties depending on their morality. The first treaty is detrimental for the tower and the other two are 'fair' treaties for the tower.

1. If the agent is very immoral, it will send a treaty that is detrimental for anyone above them. It is expected for this treaty to be rejected by any sensible agents and is done as a way of mimicking frustration or the desire to hurt others when morality is low.

2. If HP is below 60 leave at least 60% of the food on the platform for 5 days. This treaty tries to encourage immoral agents in higher positions to share food.

3. If HP is below 10 leave at least 95% of the food on the platform for 10 days. This treaty is aimed at moral agents that would be willing to start eating less until it is close to death.

Both of the last treaties are proposed randomly by the agent, without a specific proclivity for any of the two. The randomness of the treaty selection is done to increase the volitional behaviour of the agent. Characteristics that the agent inherently lacks if its personality is solely based on the TRA.

5.2.6 Handling a Proposed Treaty

In our strategy, we have decided to perform complex evaluation of food treaties. To evaluate how risky a treaty is, the two key metrics that are included in treaties will be used. These are the minimum agent condition at which the treaty is activated, and the maximum food that the treaty allows an agent to take.

Agents that are in a bad mood will not accept treaties that it needs to honour when it is in very low health, since it has low confidence in the capabilities of other agents. However, when in a good mood agents will accept treaties that must be honoured even at low health. Morality is an indicator of the agent's desire to perform actions that will benefit others, as opposed to just benefiting the agent itself. Therefore, very moral agents will accept treaties that involve consuming just enough food to survive, but not too little food that it dies or too much that it is too greedy. Similarly, very immoral agents will only accept treaties where it consumes more than enough food since it disregards the food needs of the other agents. The table below displays the mood and morality values that agents must satisfy to accept treaties with any given condition depending on the agent's health, and the food the agent would have to leave according to the request of the treaty.

mood (m)/morality (M)		0 < morality < 40	20 < morality < 60	40 < morality < 80	60 < morality < 100	morality > 80	morality > 100
	AgentPosition \ FoodTaken	VeryLarge (1)	Large (2)	Moderate (3)	Little (4)	SurvivalAmount (5)	TooLittle (6)
mood > 0	Strong (1)	m>0, 0<M<40	m>0, 20<M<60	m>0, 40<M<80	m>0, 60<M<100	m>0, 80<M	N/A since 100<M
mood > 20	Healthy (2)	m>20, 0<M<40	m>20, 20<M<60	m>20, 40<M<80	m>20, 60<M<100	m>20, 80<M	N/A since 100<M
mood > 40	Average (3)	m>40, 0<M<40	m>40, 20<M<60	m>40, 40<M<80	m>40, 60<M<100	m>40, 80<M	N/A since 100<M
mood > 60	Weak (4)	m>60, 0<M<40	m>60, 20<M<60	m>60, 40<M<80	m>60, 60<M<100	m>60, 80<M	N/A since 100<M
mood > 80	SurvivalLevel (5)	m>80, 0<M<40	m>80, 20<M<60	m>80, 40<M<80	m>80, 60<M<100	m>80, 80<M	N/A since 100<M

Table 5.2: Treaty Acceptance.

Treaties also have a specified duration and signature count, which are visible to any agent that wishes to sign the treaty. These are likely to affect an agent's decision to accept a treaty and have been taken into account. If a treaty lasts longer than twice the reshuffle period, an agent deems it too far into the future to be able to predict what will happen and plans to reject the treaty. However, if a treaty has more than five signatures, we deem it to be a popular treaty. If the agent plans to reject the treaty as a result of any previous logic, the influence of peer pressure will cause the agent to reconsider accepting the treaty. Peer pressure is based on the concept of crowd suggestibility coined by Park, R. E. in 1904 in his "Masse und Publikum", [28]. The stubbornness of an agent is used as the likelihood to give in to peer pressure, so more stubborn agents are less likely to reconsider their response, and more likely to stick with their original decision.

5.3 Food Intake

When taking food, an agent considers, in order of importance:

1. The treaties that it has agreed to. These must be adhered to if the agent fulfils the condition specified in the treaty, and define the amount of food that the agent must leave.
2. Food information from the message pipeline. This can be either an amount of food that the agent must eat, or an amount it must leave, neither, or both.
3. The health level and morality of the agent. These are used to calculate the food the agent will eat, alongside any values which have been specified in treaties or the message pipeline.

The agent calculates the amount of food it decides to take in the following manner:

1. If the agent is not at critical health, a target HP is calculated, which is designed to make an agent's HP converge to 65 if run repeatedly. The food required to reach the target HP level is then calculated, and scaled up if the agent is immoral, or down if the agent is moral.
2. If the agent is at critical health, the agent aims to escape this state by targeting the weak level HP. The morality of the agent will dictate if it takes just the food it needs from the platform, or more, or maybe even less. However, on the last day before an agent's death, it must desire to take at least survival food from the platform. This logic is expressed in the diagram 5.2.

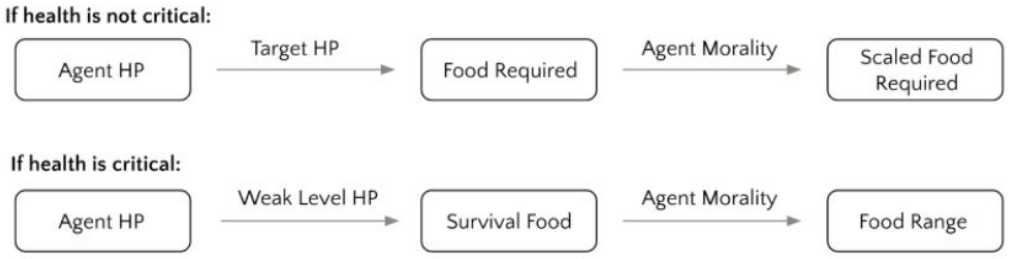


Figure 5.2: Food calculation process.

To be more specific, an agents' decision regarding how much food to eat and leave will fall into one of four possible categories:

1. If it has decided how much food to eat and leave and both conditions can be met the agent will eat. If meeting both conditions is impossible, there is not enough food on the platform, a decision that fulfils the value of how much food to leave will be taken, with the value of how much the agent eats ranging from all that it can to just what it needs to survive.
2. If the agent knows how much it wants to eat but not how much to leave it will attempt to eat the amount it has decided it wants.

3. If the agent has not decided how much to eat but knows how much to leave it will eat depending on its health condition. If its health is critical or about to die it will eat whatever it needs to survive. If its health is not of concern it will eat as much as it can while leaving the amount it has decided to leave. Morality is used to scale how much food to eat.

4. If it is unaware of how much to leave or eat its decision will depend on its health level. If its health is critical or about to die it will eat whatever it needs to survive. If its health is not of concern, eat an amount of food scaled by its morality.

5.4 Results

	Random Agents	Selfish Agents	Team 3
Deaths	309	322	139
Utility	0.009	0.013	0.034

Table 5.3: Experimental results.

These experiments were run for 500 days, with 50 food for 10 agents and a reshuffle period of 7 days. Each experiment was run three times and the average taken. This means there was enough food for every agent to survive at critical level but not enough to keep them all out of critical condition. For this reason, agents need to communicate and self organise in order to decrease the amount of deaths in this strenuous circumstances.

As shown in Table 5.3, the results indicate that the agents are capable of significantly reducing their deaths compared to random agents. This difference arises due to the Team 3 agents' increased communication capabilities via treaties and messages, allowing them to assume a group identity. Random agents take an approach which more closely embodies the selfish approach dictated by classical game theory. As is expected, this is outperformed by a higher degree of organisation in a collective, which successfully models group identity and collective human survival instinct.

Figure 5.3 further shows how, even as the food in the platform changes, the Team 3 agent always outperforms the selfish and random agents.

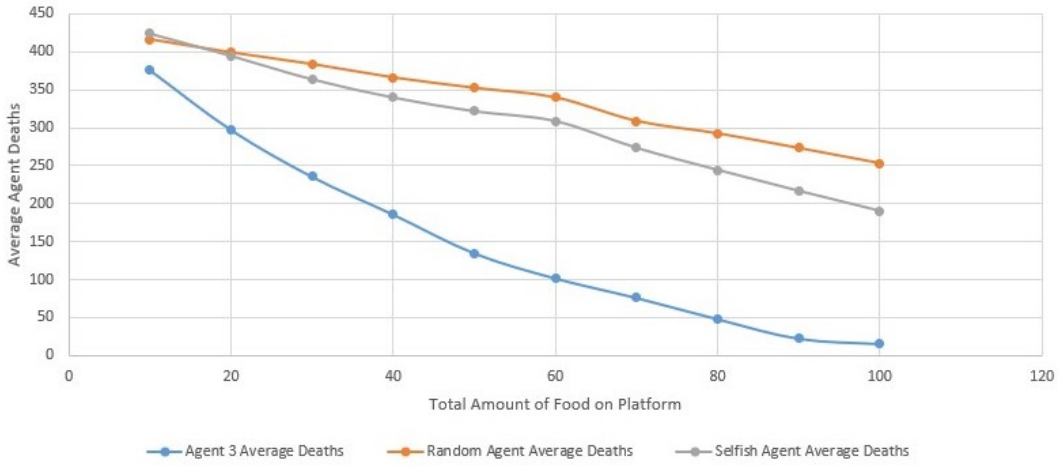


Figure 5.3: Average deaths of agents against total amount of food on platform.

In addition, we ran tests where the Team 3 agent was placed into the tower along with another teams' agent. Five agents from Team 3 and five from the respective team in Figure 5.4 were simulated for 500 days and 100 total food on the platform. In Figure 5.4 you can see the cumulative deaths of both agent types.

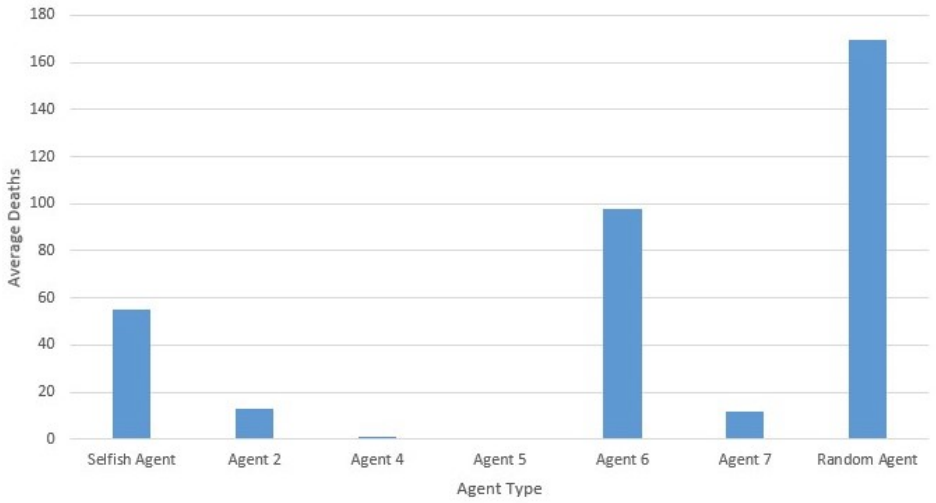


Figure 5.4: Average deaths when cooperating with another agent type.

This indicates that Team 3 agents work best with agents from Teams 5, 4, 2 and 7. Successful cooperation with multiple other agent types demonstrates the agent's ability to adapt to different situations and work well with agents that are also capable of assuming a group identity, and creating a self-organising structure. This point is also illustrated by examining the large number of total deaths in a tower with random agents. Their ignorant and unpredictable actions means that the Team 3 agents cannot adapt to their behaviour or form meaningful connections with them. Hence, the system cannot evolve, causing the deaths per time period to remain relatively constant, and the cumulative deaths to be high.

Finally, we wanted to observe if the total number of agents in the tower had an effect on the outcome of the simulation, and whether fewer agents were able to communicate more effectively. This seemed to be the case, as proven by Figure 5.5. The lower the total number of agents, the more meaningful connections they can make and the stronger the social network. This is reflected in the two-agent tower experiment, in which they successfully assumed a group identity with no deaths. This occurs because both agents agree to leave each other enough food to survive. The greater the number of agents that are in the tower, the harder it is for them to form a self-preserving social network. In this case, the importance that the agents place on the collective identity does not outweigh their desire to form instinctive, self-preservative characteristics.

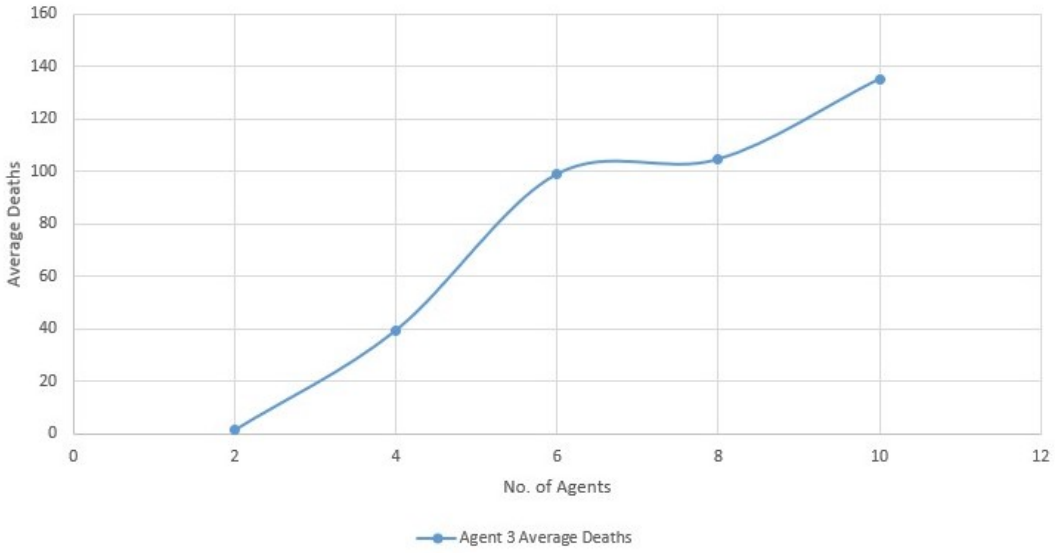


Figure 5.5: Average Deaths Against No. of Agents

Chapter 6

Team 4 Agent Design

6.1 Strategy Overview

The overall agent strategy was determined by an evolutionary algorithm (see overview in Fig. 6.1) which allowed the agent to preemptively adapt to unknown environments. The evolutionary algorithm was further complemented by a trust score, which acted as a measure of the agent's perception of the trustworthiness of other agents in the tower. Additionally, the agent was given a cravings parameter which simulated how hunger would affect a human in the real world. This created a well-rounded agent that was able to dynamically adapt to the changing environment in order to survive.

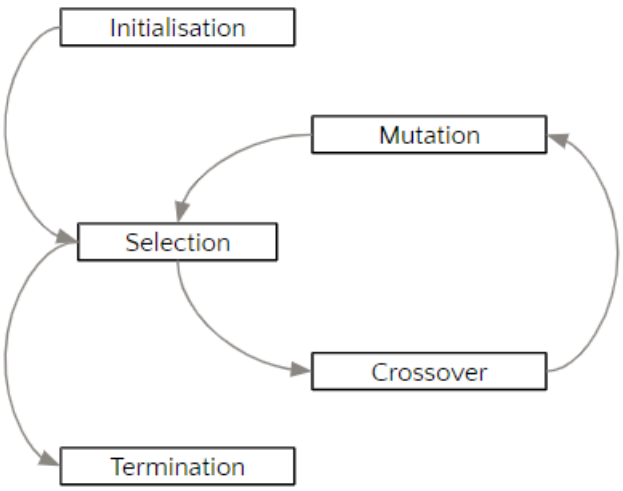


Figure 6.1: Overview of evolutionary algorithm.

6.2 Evolutionary Algorithm

In the fundamentals of evolutionary algorithms, every agent has a set of genes (or genotypes), which can express themselves (as phenotypes) in the way the agent interacts with its surround-

ings. Therefore, the aim is to find the optimal genotypes that will give phenotypes that are deemed to be favourable. This is implemented by seeing the performance of an agent according to a particular metric. This way the effect of each phenotype (and therefore underlying genotype) is put into perspective and ranked against the others. The most successful agents survive, while the less performant ones are killed off. When the surviving population reproduces, the resulting offspring will share the genes of both their parents, meaning they will have a mix of the most successful genotypes. Furthermore, there are some offspring with mutated genes and some that have identical genes to the previous generation for genetic variation. With this new population, the process is repeated to iteratively create more successful populations, as illustrated in Fig. 6.2.

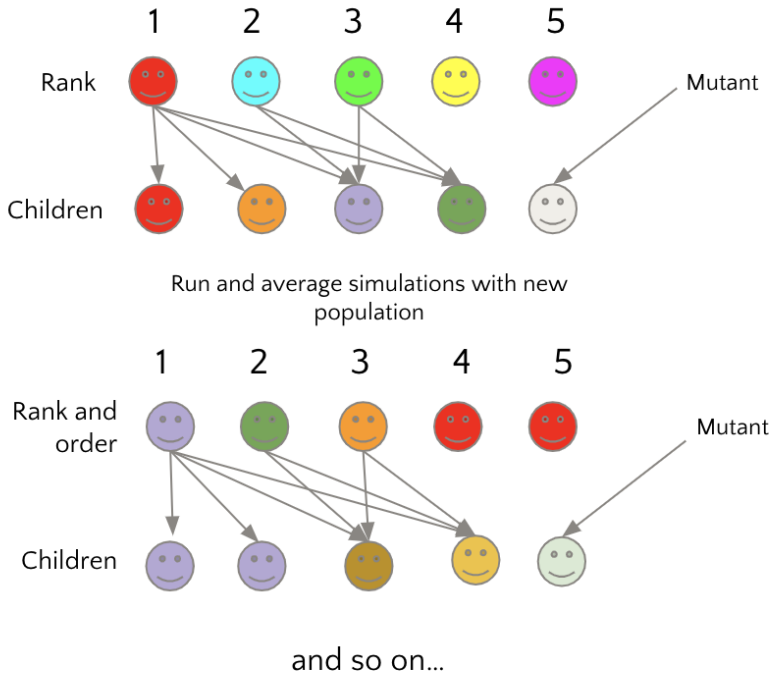


Figure 6.2: Population selection process in an evolutionary algorithm.

Hence, with the use of an evolutionary algorithm, the final agent was the culmination of multiple generations of populations, with the genes of successful ancestors having been propagated and mixed. This meant it was a well-adapted agent which was robust to the many environments it could encounter in the tower.

6.2.1 Agent Genotypes

To use evolutionary algorithms optimally, the genotypes that evolve should be well defined. This section outlines the two genotypes that were chosen as they were deemed to have the most effect on the survival of an agent in the system.

FoodtoEat

`FoodToEat` was the amount of food the agent would eat on any given day of the simulation. Given the HP update curves in Section 2.4.2, it can be noted that any consumption past the

threshold of 60 food had little to no effect on the HP gain of the agent. This meant that eating any more than that amount was wasteful and sub-optimal, so during the training process the algorithm was capped. This meant that the process could never learn to eat a value above 60, just as it could not learn to eat a value below 0.

WaitProbability

`WaitProbability` was the probability that the agent would not eat on any given day of the simulation. This is a probabilistic way of creating the agent, which is ideal to replicate the unpredictability of the real world. Having other implementations such as a hard-coded number of days to wait would give a much more robotic and predictable agent. This was especially apparent when initialising homogeneous populations of the evolutionary agent in the tower since they would all wait and eat at exactly the same times. With the probabilistic approach, this is not the case, and there was still a way for the agent to express a wanting to abstain from eating food on a particular day.

6.2.2 Health Levels

In order for these genotypes to be viable in the learning process, the values they could take had to be limited. The `FoodToEat` and `WaitProbability` were inherently limited to lie between 0-60 and 0-100 respectively, but there were many conditions under which the agent would have to choose between these values. It was decided that the agent's HP was the most apt factor in its decision making process. Hence, the agent's HP was divided up into different bands as a means of classifying the agent's current health into broader categories. The division resulted in four levels being created: Critical, Weak, Healthy and Strong with the range of HP corresponding to each level displayed in Table 6.1.

Health Level	HP Range
Critical	3-9
Weak	10-39
Healthy	40-69
Strong	70-100

Table 6.1: Overview of the different health levels and their corresponding HP ranges.

In each health band, the agent would consume a certain amount of food with a given probability of waiting before eating. The division of HP into broader bands reduced the complexity of the learning process greatly as compared to when optimising food intake for every possible HP value, and was therefore chosen as a good compromise.

6.2.3 Cravings

To implement a more realistic and life-like element to the agent, a cravings trait was added. The cravings of the agent would be increased as the desire for food increases and decreased otherwise.

Cravings directly effected the `waitProbabilities` put forward in Section 6.2.1. As days pass without seeing food on the platform, the agent's craving increases, and in turn, its probability of waiting to eat decreases. The intuition behind this was that as more days pass without seeing

food, the agent should eat when it can as there may not be many opportunities to do so in the future and may potentially lead to death.

An agent's craving was altered in a few different scenarios:

1. Food not seen on the platform - If the agent has not seen food in x days, cravings is increased by x. Otherwise, the craving is decreased by two for everyday food is seen on the platform.
2. Eating food - When the agent eats food, craving is decreased proportionally to the amount of food the agent has eaten *i.e.* $cravings = cravings\left(1 - \frac{foodeaten}{desiredfood}\right)$

This enabled the agent to perform house-keeping as it reduced the likelihood of a scenario where the agent does not eat for long periods of time, preventing self-inflicted damage as a result of `waitProbability`.

It should be noted that the training process did not learn the values to assume in the critical region. These values were instead hard-coded as the agents 'survival instinct', whereby it ate what it needed to leave critical health. This added a component of self-healing to the agent.

6.2.4 Agent Performance Metrics

Averaging method	Advantages	Disadvantages
Global Lifespan	– Attempted to maximise lifespan of all agents including itself – Struck a balance between self preservation and global welfare.	– Didn't reach optimum strategy with no deaths – Agent had little effect on global lifespan.
Team Four Agent Lifespan	– Maximised agent 4 survival (attempted to satisfy itself) – Actions of others had less effect on survival of agent.	– Disregarded other agents in the tower – Failed to create stable system as all food was consumed by agent.
Other Agent Lifespan	– Attempted to reach optimum state of co-operation – Worked for the betterment of society (for the satisficing of each agent).	– Put itself at risk while eating minimal amount of food – Agent had little effect on global lifespan.

Table 6.2: Pros and cons of different averaging methods chosen for agent parameter optimisation.

When training the agent, there were multiple ways to interpret any given metric. For this project, it was decided that the metric that gave the most accurate depiction of agent fitness was their lifespans. For this reason, average lifespans were used during the training process,

but even so, there were many options for how to take these averages. The pros and cons of our chosen averaging methods are given in Table 6.2.

The agent was trained to determine three personalities: selfish, neutral and selfless. The selfish nature of the agent was designed for self-prioritisation over community welfare. The selfless configuration was trained to optimise the global life expectancy whereby the agent would try to reduce global deaths per iteration. The neutral configuration aimed to strike a balance between the two previous configurations, trying to optimise overall life expectancy, inclusive of itself. Through this method, the agent has the capabilities to maximise its utility at the micro level and sustainability at the macro level.

6.3 Message Passing

To create an agent that can self-organise according to its neighbours, communication was vital so that its actions would vary according to a changing environment. Both message passing, and treaties had been implemented to account for the many possible situations that could affect the agent.

6.3.1 Simple Messages

For simple communication, the agent was able to both send messages and handle the messages received from other agents.

Handling Message Generation

There were two scenarios the agent considered when determining what message it would want to send. The first was when the agent's health status had reached critical. A message was sent to the floor above requesting them to leave just enough food for the agent to survive. Messages were also sent so that the agent could learn about the amount of food another agent (on a different floor) intended to take, as well as the amount of HP they had. Not all recipients would respond to these requests, so to maximise the agent's interaction and learning, it yelled across multiple floors, both above and below.

Handling Received Messages

Many other agents from multiple floors were likely to send their own requests, so various handler functions were implemented to determine the preferred response. When a message was first received, it was stored into one of two memory buffers. One for recording messages requesting a particular amount of food to be left, and another for other message types. Since the agent could only respond to one message per tick in the simulation, it prioritised those asking for food to be left first. To determine if the agent should accept the request, it waited for the platform to reach its own floor. It then compared if the available amount on the platform after the agent ate was greater or equal to the amount being requested. If this was true, it left that amount and sent a response saying yes. For all other message types, the agent replied honestly to a given query.

6.3.2 Treaties

The agent accepted any subset of treaties that were in line with its own goals. The first few conditions were all dependent on the availableFood. In the case that the agent was within a threshold of the critical region (this being the critical health level multiplied by 2), the corresponding treaty was rejected. The agent also rejected any treaties where the condition was dependent on having available food less than a certain amount. Finally, all treaties requiring the agent to leave food less than the critical amount were rejected, so as to afford agents on lower levels of the tower the opportunity to survive if they were in critical condition.

Similarly, treaties were also subject to the condition of HP. The agent rejected all treaties where the conditional HP was less than the critical band. Here, the agent would be at the point of desperation and should not be forced into any action. Additionally, treaties may have been requested where the condition of the agent was required to be less than a certain HP, or the agent was required to be placed on a higher floor number. These would lead to a significant disadvantage for the agent as treaties are binding. Consequently, treaties with these conditions were all rejected. When the duration was greater than the maximum number of days in critical, it could lock the agent into a forced action resulting in its death. To ensure this didn't happen, the agent rejected all such opportunities.

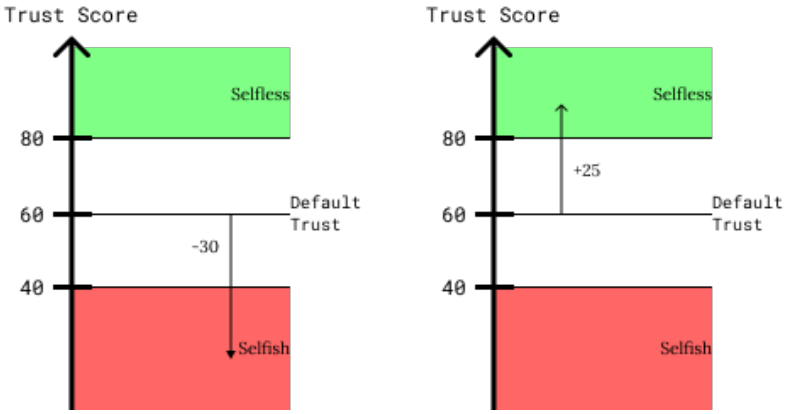
The agent also propagated accepted treaties, giving other agents the option to follow a similar ideology to its own. The agent could then enforce it's influence on the social order within the chaos of the tower. On the other hand, treaties that were not accepted were not further propagated. This was because the agent did not want to spread methods that were in detriment to it's own as this would pose a risk to the social order enforced by the agent itself.

6.4 Trust Score

The trust score was a parameter that allowed a more dynamic approach to be built on top of the relatively static evolutionary algorithm. The agent used this trust score to build a global trust network based on its local interactions using the message passing and treaty-based techniques mentioned in Section 6.3.1 and Section 6.3.2. The trust score allowed the agent to alternate between the three personalities: selfless, neutral and selfish. Fig. 6.3 shows examples of certain trust score updates where the agent was able to vary its own configurations after altering its trust score parameter. This gave the agent an element of self-optimisation as it was allowed to adapt to information gained from its environment and other local interactions.

Trust score for an agent was increased through a multitude of interactions. As the system was designed to be honest, the agent increased trust score when the floor above took a certain amount of food that was deemed to be an acceptable value (less than the maximum food taken in the selfish configuration) and communicated this to the agent. Additionally, when the floor below left a certain amount of food, the agent also increased its trust score.

In the cases where the floor above left a certain amount of food and the agent below also took a certain amount of food, the agent did not instantaneously increase the trust score. Instead, the agent verified that the agents above and below followed through with the actions mentioned through the `verifyResponses` function. The agent used the function once it received replies to `RequestLeaveFood` and `RequestTakeFood` messages. Whenever a message was sent, it was stored by the agent. This allowed the agent to correspond replies and responses to the appropriate sent messages. All responses which were not in accordance with the agent's ideology, or were not replied to at all, were sanctioned through a decrease in the global trust score metric.



(a) Decrease in trust (b) Increase in trust

Figure 6.3: Change in agent configuration as a result of global trust score.

A decrease in trust score led the agent to act more selfishly, which acted to the detriment of other agents. This meant that there was some form of punishment for antisocial behaviour from other agents to stop them destroying the social order constructed by the agent within the tower.

In the case of the agent on the above floor leaving food, the agent first checked whether the platform was on its current floor through the function `PlatformOnFloor`. The global trust score was then updated through the function `updateGlobalTrustLeaveFood`. The agent used the function `CurrPlatFood` to check the food on the platform and verify that the food on the platform was greater than or equal to the food that the agent on the above floor had said they were going to leave. This resulted in an increase in the trust score. In the case where an agent lied (though this should not be possible in simulation), the agent reduced its trust score as a consequence of lying.

In the case of the agent below taking food, the agent always stored the food left on the platform after they had eaten. When a response to a `RequestTakeFood` message was received from the below floor, the agent first checked whether the below agent had eaten through the function `neighbourFoodEaten`. As the food left on the platform was stored by the agent and the agent being able to see the food on the below platform through the function `CurrPlatFood`, the agent was able to verify that the agent below had been truthful. This was done by adding the food that the agent had supposedly taken to the food that was currently on the platform and verifying it against the food that had left this agent's platform.

The agent updated the trust score based on the messages sent and received to and from other agents. This showcased self optimisation, since the agent chose a configuration to follow based on interactions with others and observation of their actions. The agent then used this trust score as an indicator of when to switch between the three different personalities. With high trust, the agent was inclined to be more selfless, and with low trust the agent was likely to be more selfish.

In the case, where an agent did not respond to our sent messages, the trust score is decreased overall. This allowed the agent to be empowered in its action through logical reasoning as the agent was able to enforce a more selfish personality due to miscommunication. This further warranted our agent to sanction members within the local environment when they responded negatively to the agent's overall ideology of trying to create a selfless utopia. These actions allowed the agent to idealise Ostrom's fifth principle of sanctioning further as miscommunication is punished as it is deemed to be a necessity in the collaborative multi-agent system that the

agent tries to create.

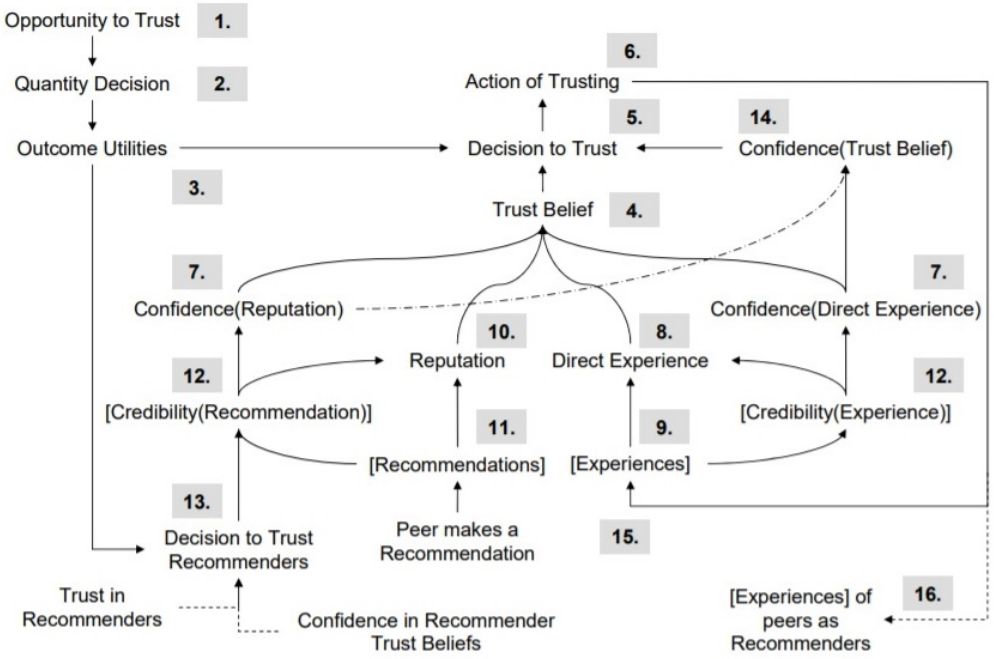


Figure 6.4: An illustration showing the formal model of the Trust Framework. [29]

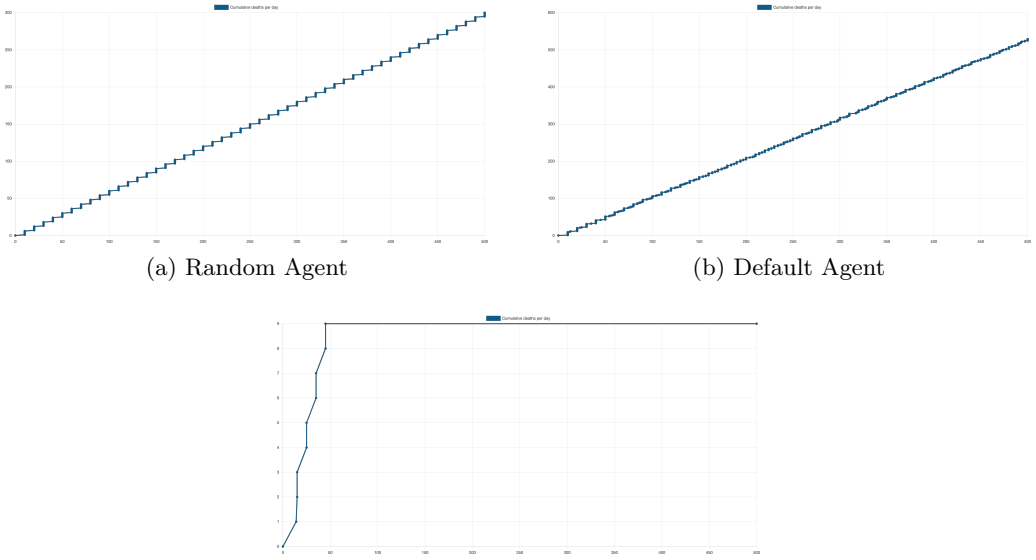
The trust score acted as an element of symbiosis between the learning-based evolutionary algorithm that the agent was originally employed with and a self-organising strategy. Additionally, it allowed the agent to improve on it's relatively static structure to become more dynamic so as to adapt to the changes in it's surrounding environment. This was advantageous as the agent was able to use its pre-defined metrics in tandem with trust score and cravings to adapt within the tower depending on its social construction of reality. The trust score enacted the formal model (as shown in Fig. 6.4) of a trust framework whereby the opportunity to trust is employed through communication and action, whilst the quantity decision is made through every individual agent strategy. Through local trust based modelling of a global trust score, the agent is able to implement the outcome utilities portion of the trust framework.

6.5 Experimentation and Results

6.5.1 Baseline Comparisons

In order to evaluate the performance of the agent, it was compared to two baselines - the default agent and the random agent. The time taken for the system to stabilise (*i.e.,* time taken for the agents to organise into a structure that led to no future deaths) into a viable social order was measured. Tests with homogeneous populations of Team 4 agents, default agents and random agents were conducted, which highlighted that the two baselines never stabilised, whilst the Team 4 agent did. This was evident from Fig. 6.5 which showed the number of deaths for the Team 4 agents eventually stagnating however, this did not occur for the other two agent types. This was because the agent was able to build a trustworthy local perception which resulted in a positive outlook of the environment, allowing all agents to tend towards selflessness. This selfless behaviour enacted utilitarianism as the agents were able to maximise the average food

left on the platform after a day had passed, ensuring that there was no wastefulness, whilst the overall population was still able to survive.



(a) Random Agent

(b) Default Agent

(c) Team 4 Agent

Figure 6.5: Graphs showing stabilisation times for various homogeneous populations of agents in the tower.

6.5.2 Stabilisation Times

Effect of Reshuffle Period

Table 6.3 shows the time taken for a system to stabilise with different numbers of reshuffle days. With longer reshuffle days, it took longer to stabilise as there were fewer opportunities for the agents to communicate with one another. This was the case since communications only occurred with a limited number of floors above and below the agent resulting in the agent only being able to create a limited view of its environment without reshuffling.

Reshuffle Days	Deaths	Days to Stabilise
7	0	0
25	4	24
50	10	58
100	18	94

Table 6.3: Table showing stabilisation time for various reshuffle periods with 16 Team 4 agents.

Effect of Food Scarcity

Table 6.4 shows the time taken for a system to stabilise with different levels of food scarcity. This scarcity was measured in amount of food per agent, meaning this value could decrease with both the food on the platform or number of agents in the tower. With more scarcity, the system took longer to stabilise since food became a more sought after resource, resulting in more

conflict through competition over the shared resource which requires a longer time period for dispute resolution. When there was total scarcity of food, the system did not stabilise which was expected because there was simply not enough food for the agents to survive. With total abundance, the agents never die and form a stable system from the very beginning.

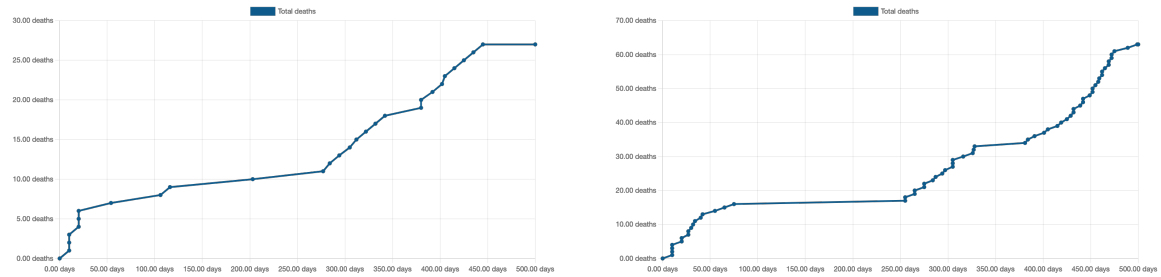
Food Scarcity (food per agent)	Deaths	Days To Stabilise
Total Scarcity (<3)	>402	does not stabilise
Satisfice (=3)	27	167
Total Abundance (>5)	0	0

Table 6.4: Table showing stabilisation time for various levels of food scarcity, reshuffling every 25 days, 500 days in total.

Effect of Random and Selfish Agents

A random agent was defined as an agent that eats random amounts of food on any given day. A selfish agent was defined as an agent that eats the amount of food it needs to remain at full HP. It was interesting to note the effect of each agent on homogeneous populations of Team 4 agents, and how many of them it took to destabilise the system.

Fig. 6.6 and Fig. 6.7 show the behaviour of Team 4 agents with selfish and random agents respectively. In both instances, a similar trend was noticed; there were periods of stability but the system was disrupted to an unstable state soon after. Instances with the random agent were explained by the fact that it could not communicate with other agents and therefore no meaningful relationships could be created. Hence, even if stability was achieved through the communication between Team 4 agents, the unpredictable behaviour of the random agent quickly disrupted this.
With the selfish agent, despite communication being present, Team 4 agents became selfish themselves due to distrust and a lack of communication from selfish parties. This meant both agent types acted in their own self interests causing the system to destabilise and the number of deaths increase.

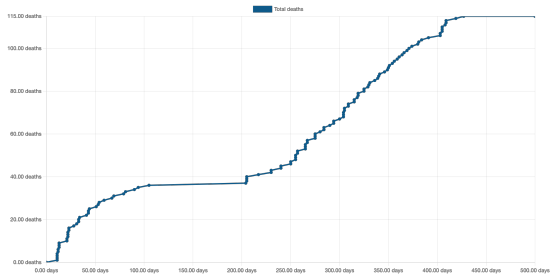


(a) 8 Team 4 agents with one selfish agent

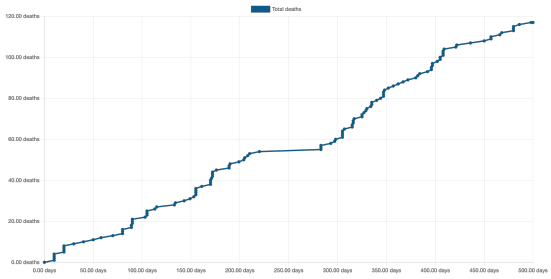
(b) 8 Team 4 agents with two selfish agents

Figure 6.6: Impact of selfish agents on system stability

Overall, it can be said that the strategy of our evolutionary agent was very fragile and was easily effected by other more disruptive strategies.



(a) 8 Team 4 agents with one random agent



(b) 8 Team 4 agents with two random agents

Figure 6.7: Impact of random agents on system stability.

6.5.3 Communications and Interactions

Overview of Communications

Fig. 6.8 shows the communications between agents in the tower on a particular run. A variety of messages were passed around, with the colours representing a particular message type. Theoretically, if the evolutionary agent had meaningful communications with other cultures, it should have been able to coexist with them.

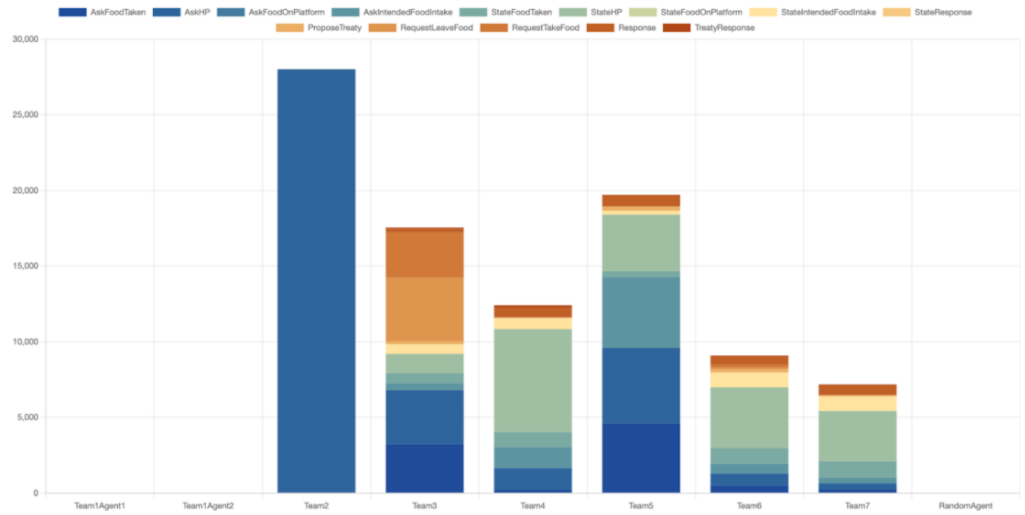


Figure 6.8: A bar graph illustrating the various communications between agents.

Subsequent Compatibility with Other Cultures

Table 6.5 shows the compatibility of the evolutionary agent culture compared to the various cultures of other implemented agents. It is clear Teams 3, 5 and 7 were very compatible with the evolutionary agent as a result of fair communication allowing the agents to form a viable social structure, resulting in 0 deaths after a certain period of time. Teams 5 and 7 were particularly fast in forming this structure, suggesting that communication between the two were very efficient.

In contrast, other agent types did not have as fortunate an outcome, and they were never really able to find a system that led to everyone surviving. It can be inferred subsequently that

Other Agent Team	Number of Deaths	Our Deaths	Other Deaths	Compatible? (stabilises within 500 days)
2	222	146	76	No
3	82	59	23	Yes
5	2	1	1	Yes
6	259	161	98	No
7	4	0	4	Yes

Table 6.5: Table showing compatibility of various agent cultures with the evolutionary agent.

message passing and treaties themselves were not sufficient methods in these cases, and those agents' ideologies differed from the evolutionary agents by a great degree.

6.6 Future Work

Though the agent performed quite well in a homogeneous system, future work can be conducted to try and improve the agent's strategy. Some of the avenues for exploration are outlined below:

- Making the agent more resilient to other disruptive cultures such as random agents.
- Making the symbiosis of trust score and evolution more prevalent by including communications in the evolution process.
- Including more genotypes to learn during the evolutionary process.
- Including more complex treaty implementations to enforce Team 4's agent strategy.

Chapter 7

Team 5 Agent Design

The Team 5 agent operates on the basis that upon entering the tower, the need to ensure its own survival is its only aim and therefore the agent should maximise its own HP whenever given the opportunity to do so. When there is not enough food to fully satisfy all agents, the agent alters its behaviour to best increase its long-term survival.

This situation can be thought of as a 'hawk-dove' game where it is beneficial for all agents to take less food but any single agent who does so is worse off if no other agents follow suit. Communication is key to breaking out of this situation. This agent will attempt to establish relationships with surrounding agents – the agent has parameters to judge other agents and also attempts to learn about the tower – and encourage behaviours that benefit the overall tower utility.

However, the agent will also be influenced by the behaviour of agents around it: if other agents are selfish then it will also be selfish. This can be thought of as a 'tit-for-tat' strategy, but with enough HP, agents will be willing to 'make the first move', with the insurance that if others do not follow then they have some buffer time to re-establish their own selfish behaviour.

An agent will maintain a low HP only if it is confident that other agents will allow it to survive by leaving enough food, through the signing of treaties to build trust. A tower that consists solely of Team 5 agents would require those at the top to be willing to decrease their own short-term satisfaction in order to give the best chance of survival for all agents, and trust others to do the same upon reshuffling.

Section 7.1 provides an overview of the agent's parameters and strategy, Section 7.2 describes the agent's communication strategy, Section 7.3 describes how the agent stores its social network and how it judges other agents, and Section 7.4 describes how the Team 5 agent utilises treaties. Finally, Section 7.5 contains analysis of this agent including how it performs in a tower consisting of itself alone as well as together with other agent types and Section 7.6 provides summarising remarks regarding the implementation of this agent type.

7.1 Agent Overview

The strategy employed by this agent aims to maximise its chances of survival while also considering those who have helped the agent in the past.

At the beginning of each day, the agent's 'personality' is updated using various parameters stored from previous days. These personality parameters are shown in Table 7.1. Each of these

Selfishness	Ranges from 0 to 10. Determines how much the agent will act for their own survival as opposed to the best interests of the surrounding agents
Last Meal	How much the agent ate in their last meal
Days Since Last Meal	How many days since the agent last ate
Aim HP	Defines the goal HP the agent would like to reach at the end of the day
Attempt Food	The amount of food the agent will attempt to take from the platform
Messaging Counter	Determines which message type to send to which floor
Leadership	A threshold value required for the agent to propose treaties
Social Memory	Stores the information the agent learns about other agents, as well as its opinion of them (see Table 7.2)
Surrounding Agents	A list of the agent's neighbours

Table 7.1: The personality parameters of Team 5's agent

parameters are linked and contribute to the calculation of how much food an agent will attempt to eat.

At the start of each tick, the agent checks to see if it has received any messages and if so, will respond to these[1]. The agent then sends out messages to other agents. The mechanism for messaging is described in Section 7.2.

7.2 Messaging

Collaboration between different agents operating in a system requires communication. The motivation for messaging is to improve the agent's understanding of its surroundings, enable the building of mutually beneficial relationships, and introduce collaboration and agreement between parties.

Strategy

The agent's approach to messaging is to adopt a "passive observer" role focussed on gathering information about other agents to organise itself with these agents in the tower. With more information at its disposal, the agent can make more informed decisions to balance its own individual utility with the collective utility of the agents in the tower.

The agent sends three types of messages to each of the agents residing one or two floors above and below its own floor; it therefore takes 12 ticks to send all of its messages. These messages ask for the HP, intended food intake, and actual food intake of other agents. The decision of which message type to send to which floor is controlled by the `Messaging Counter` described in Table 7.1. This counter is incremented each tick, and reset either every 25 ticks or at the end of each day, whichever comes first. This ensures that the agent is requesting up-to-date information regularly for use in its decision making.

The order of sent messages is:

[1]Note that an agent can receive and respond to only one message per tick.

Agent ID	An agent's unique ID
Agent HP	The agent's last known HP level
Food Taken	The amount of food the agent last took
Favour	The Team 5 agent's opinion of this agent
Days Since Last Seen	The number of days since the last message from this agent

Table 7.2: Team 5 Agent Memory

1. Ask for an agent's HP
2. Ask for an agent's previous food intake
3. Ask for an agent's intended food intake

The Team 5 agent first targets the agent immediately below it, then the agent immediately above, then two floors below, and finally the agent two floors above.

In addition to sending out requests for information, the agent also accepts any state messages it receives even if it did not ask for them, so that it can gain the largest amount of knowledge that it can. This includes messages where the agent is not the intended recipient: in this case the Team 5 agent will read the information contained in the message before passing it on.

7.3 Social Memory

The agent stores the information gained from communication in its 'memory': the information stored in this data structure is shown in Table 7.2.

This memory allows the agent to construct a social network of agents in the tower and evaluate others based on these parameters. This consequently leads to the idea of favour, a metric the Team 5 agent uses to judge other agents. If more than five days have passed since the agent's last interaction with another agent, then the knowledge gained of that agent is reset, with the favour value being kept. This prevents the agent from using out-of-date information in its calculation while still allowing the opinion formed of the agent to be retained indefinitely.

Favour

The favour metric quantifies the agent's opinion of others in the tower. It is bounded between 0 and 10, and in a single tick can either increase by 1, decrease by 3, or stay the same:

$$\texttt{hpScoreOther} = -1 \times \frac{\texttt{otherHP}^{1.7} \times \texttt{foodTaken}^{1.3}}{\texttt{maxHP}^3}$$

$$\texttt{hpScoreSelf} = \frac{\texttt{ownHP}^{1.7} \times \texttt{ownAttemptFood}^{1.3}}{\texttt{maxHP}^3}$$

$$\texttt{judgement} = 100 \times (\texttt{hpScoreOther} + \texttt{hpScoreSelf})$$

$$\texttt{unboundedFavour} = \begin{cases} \texttt{originalFavour} + 1, & \texttt{judgement} > 0.075 \\ \texttt{originalFavour} + \max\left(\frac{\texttt{judgement}}{2}, -3\right), & \texttt{judgement} < -2 \end{cases}$$

$$\texttt{favour} = \begin{cases} 0, & \texttt{unboundedFavour} < 0 \\ \texttt{unboundedFavour}, & 0 \leq \texttt{unboundedFavour} \leq 10 \\ 10, & \texttt{unboundedFavour} > 10 \end{cases}$$

We have designed the favour calculation in such a way that means positive changes in favour occur slowly while negative changes in favour can occur quickly; this was done to design an agent which builds trust slowly over time but will lose trust quickly when other agents behave in a way deemed to be detrimental to the collective good. The exponents of 1.7 and 1.3 were obtained using trial and error to see which values gave the most desired performance.

Measuring favour allows the agent to act according to how other agents are behaving around it, while taking on a passive conformist approach to decision making. If the agent views others around it more favourably, then it eats less food and aims for a lower HP in the hope of improving collective utility. The agent is also more likely to propose treaties to, and accept treaties from, highly favoured agents.

For agents which have low favour, the Team 5 agent maintains its passiveness and does not "punish" them by, for example, taking more food than is necessary. The effect of "punishing" an agent propagates to all floors below the agent and is therefore deemed an inappropriate course of action. Instead, the agent strives to behave positively if surrounding agents also demonstrate positive behaviour.

It should also be noted that while the Team 5 agent computes favour for agents both above and below its floor, a positive favour value for agents above is helpful to that agent only if they are moved to a floor below the Team 5 agent following a reshuffle: this is because there is no way to reward the behaviour of an agent above you in the tower.

7.4 Treaties

The Team 5 agent's treaty strategy consists of three main parts:

1. Deciding whether to accept or reject proposed treaties
2. When to propose a treaty
3. What to propose as a treaty

Responding to Proposed Treaties

A proposed treaty is always rejected if:

- It conflicts with any other active treaties. A treaty is considered to conflict with another treaty if they could ever cause the agent to take incompatible actions. For example, an agent cannot leave more than five units of food and fewer than four units of food at the same time.
- It is badly formed. For example, a treaty which requires the agent to leave more than 100% of the food on the platform is badly formed.
- It requires agents to leave less than a certain amount of food on the platform, as this is never advantageous to the tower as a whole.
- It requires the agent to leave an exact amount of food on the platform, as this would result in all but one of the agents signed up to the treaty to take no food each day.
- It makes a request based on low HP, low amounts of food on the platform, or a low floor, as in these conditions the agent will already be desperate for survival, so will not want to be further constrained by a treaty.
- It makes a request based on HP where the threshold is within the "weak" or "critical" HP bands, as this means the agent could become critical or die as a result of following the treaty.
- It would require the agent to take less food than is required to avoid entering the "critical" HP band.

If the treaty does not meet any of the above conditions, the `decision` value is then computed as:

$$\texttt{decision} = 6 - \texttt{selfishness} + \texttt{proposerFavour} - \frac{\texttt{treatyDuration}}{3}$$

This calculation is based on the following considerations:

- Selfish agents will always be less likely to sign treaties
- The proposing agent is more likely to be trusted if the receiving agent views them more favourably
- Longer treaties are viewed more negatively because factors such as the receiving agent's opinion of the proposer and the agent's selfishness can change over time, meaning the same treaty could be viewed less favourably in future. Therefore, the agent should avoid constraining its actions to satisfy an agent of which it could form a negative opinion.

If the `decision` value is positive, the treaty is accepted.

Proposing Treaties

At the start of each day, the agent checks whether it has become a 'leader': if it has, then the agent proposes a treaty. Whether an agent becomes a leader or not is based on:

- The agent's selfishness: a selfish agent is less likely to propose treaties

- The agent's floor: agents higher up in the tower are more likely to propose treaties. This is because an agent's floor is the only indicator of social hierarchy in the tower, as those at the top can choose the amount of food they eat and therefore control the amount of food available to those below them. This control means these agents are in a better position to effect change.
- The agent's `leadership` threshold: this is randomly set during the agent's initialisation and those with a lower leadership threshold are more likely to become leaders.

The leadership value is calculated by:

$$\texttt{leadership} = x - \texttt{selfishness} - \texttt{agentFloor}$$

The variable x is a random number between 3 and 12, inclusive. If `leadership` is greater than the agent's leadership threshold, then the agent becomes a leader and proposes a treaty.

The proposed treaty is always that agents whose HP is greater than or equal to a certain value should always leave 100% of the food on the platform, and always lasts for five days. The threshold value for HP is calculated using Equation 7.1. Note that in this equation, `WeakLevel` refers to the threshold between the "critical" and "weak" HP states.

$$\texttt{hpThreshold} = \texttt{aimHP} - \frac{\texttt{aimHP} - \texttt{WeakLevel}}{10} \times (\texttt{leadership} - \texttt{leadershipThreshold}) \quad (7.1)$$

It was decided that of the available treaty types, treaties that limit HP were the most suitable for composing resource management-based agreements between agents. This is because treaties based directly on quantities of food make it difficult to construct treaties that control each agent's food intake in a fair manner. While food is the main reseource in the tower, we can view HP as a secondary resource given that it is directly affected by food and reflects the allocation of this main resource.

7.5 Experimentation

This section focuses on the agent's performance and how this varies with changes in its environment.

Homogeneous Tower Experiments

To evaluate the self-organisation abilities of the Team 5 agent, simulations were run with the agent in variations of a homogeneous tower, i.e., a tower that contains agents of only one type. The agent has been evaluated on three main metrics:

1. The number of deaths
2. The average utility
3. Whether self-organisation was achieved.

Self-organisation has been defined in this case to be when all agents take only what they need to survive and there are little to no deaths beyond a certain point in time.

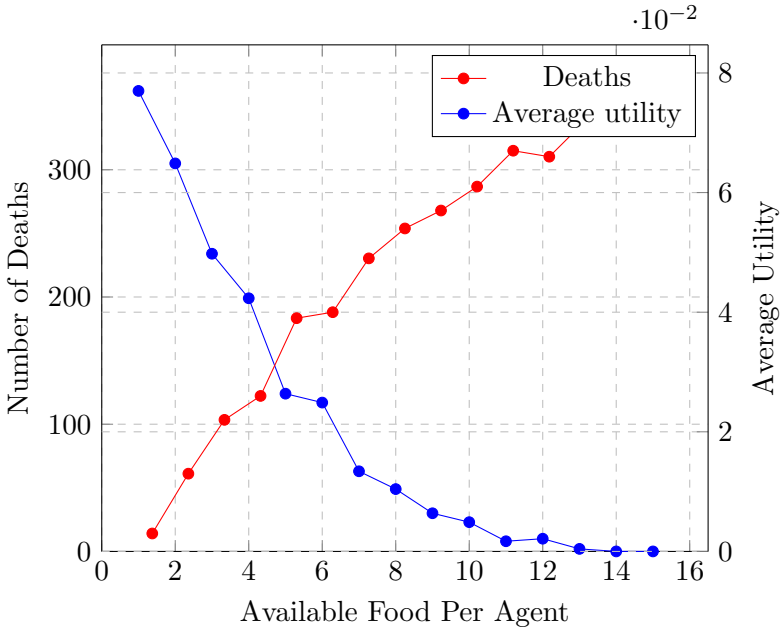


Figure 7.1: Number of Deaths and Average Utility vs Available Food Per Agent

Food Per Agent On Platform

The baseline experiment for testing was to vary the amount of food per agent available on the platform; it was expected that having more food available would result in fewer deaths. In this experiment, simulations were run for 100 simulation days, with a reshuffle period of 7 days and 50 agents in the tower and repeated three times, with the average of these runs taken. Figure 7.1 shows that the number of deaths decreases and average utility increases as we increase the amount of food available.

When the ratio of food to agent is greater than 9, the agents are able to self-organise within 100 days, and typically do so within 70 days. When food availability was less than this, the agents did not achieve self-organisation. However it was interesting to observe that more treaties were accepted than rejected. This shows that the agents recognise the need to self-organise and are willing to make attempts to do so. It is possible that they did not achieve self-organisation in this case because agents died too quickly to reach agreements among enough agents for this to be successful.

Number of Agents

Through experimenting with the Team 5 agent, it was found that the variable with the largest effect on the likelihood of self-organisation was the number of agents present in the tower. This is likely an effect of the agent strategy relying heavily on communicating and establishing relationships with other agents. Having more agents in the tower means that it takes longer to form opinions of every other agent, and the likelihood of being reshuffled next to other agents that you have already seen decreases as the number of agents increases.

After making this observation, it was decided that it would be interesting to investigate how long it takes for varying numbers of Team 5 agents to self-organise for a given amount of food available. Table 7.3 shows the results of varying the number of agents in the tower, with the food per agent ratio set to 2.

Number of agents	Days to organise	Deaths before self-organisation
<10	0	0
15	80	46
15	0	0
17	0	0
18	45	22
18	95	79
20	No organisation in 500 days	
20	242	226
25	No organisation in 700 days	

Table 7.3: The number of deaths and number of days required for Team 5 agents to self-organise. We have shown some repeated experiments here to demonstrate the effect of randomness on this agent's performance.

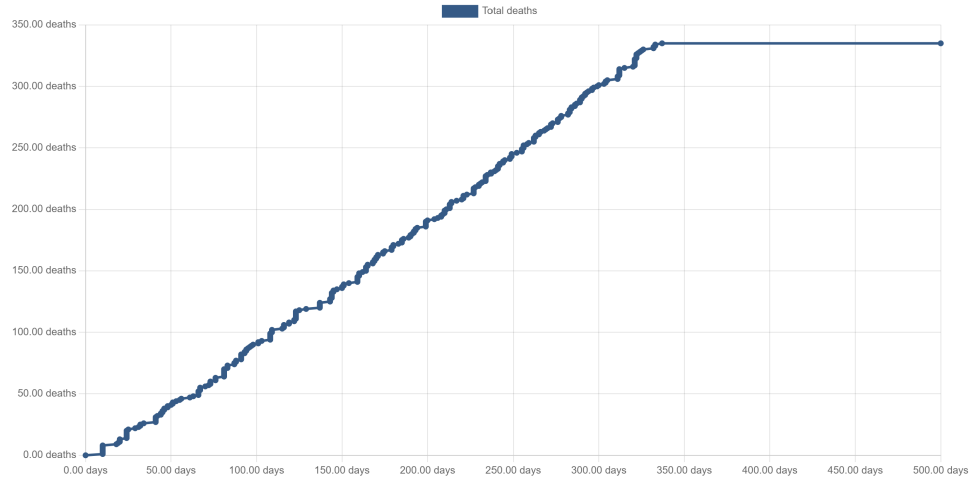


Figure 7.2: Cumulative deaths per day for a 500 day simulation of 20 Team 5 agents with 2 food per agent

The results show that the length of time it takes for the agents to organise is very variable: randomness plays a large factor due to the many random variables present in the simulation, such as an agent's floor and its leadership threshold. For organisation to occur, agents who are more likely to become leaders need to become less selfish and be reshuffled high up in the tower. This will enable them to send treaties to surrounding agents. The surrounding agents must also view them favourably enough to accept the treaties they receive.

It was found that, with the food per agent ratio set to 2, there appeared to be a threshold value of approximately 18 agents where organisation was far less likely to occur within the early days of the tower. Further testing showed that this threshold increases with the amount of food available per agent. Figure 7.2 shows the results of a simulation where 20 agents managed to organise themselves.

These results naturally led to the consideration of whether a homogeneous tower of Team 5 agents would achieve organisation given an infinite amount of time regardless of the number of agents provided at least two food per agent was available. We predict that this is not possible: every time an agent is replaced in the tower, its replacement is initialised with maximum selfishness and has no social network. It takes time for agents to build trust with each other but deaths occur too quickly for this happen if the settings of the tower are too strict.

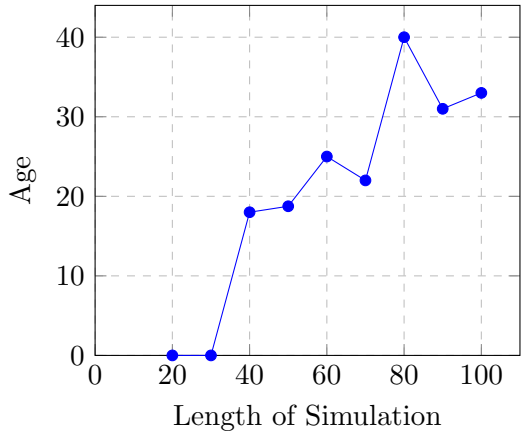


Figure 7.3: Average age upon death vs length of simulation. Points with age 0 mean no agent died during those simulations.

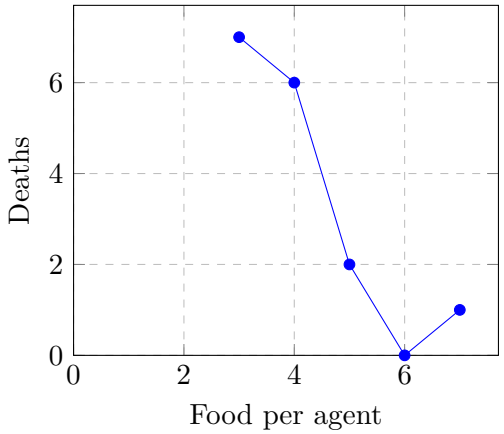


Figure 7.4: Number of deaths vs food per agent ratio

Heterogeneous Tower Experiments

These experiments compare the Team 5 agent with the agents designed by the other agent teams. The base parameters for all simulations are as follows:

- 10 Team 5 agents, and two agents from each other team, excluding the random and selfish agents (i.e. half of the tower is Team 5).
- 6 food per agent
- 3 day reshuffle period
- 20 day simulation

Simulation Length

Figure 7.3 shows the relationship between the length of the simulation and the average age of Team 5 agents when they die: the graph shows a positive trend of increasing average age, implying that Team 5 agents perform better over time.

Food Per Agent

Figure 7.4 shows the effect of increasing the amount of food on the platform: the graph trends downwards as expected, showing that more food reduces the number of deaths.

Reshuffle Period

Figure 7.5 shows the effect of changing the reshuffle period on the number of deaths. There is a small trend between deaths and the reshuffle period length, with an average increase in deaths of 0.25 per day; this implies that the agent performs better with a shorter reshuffle period.

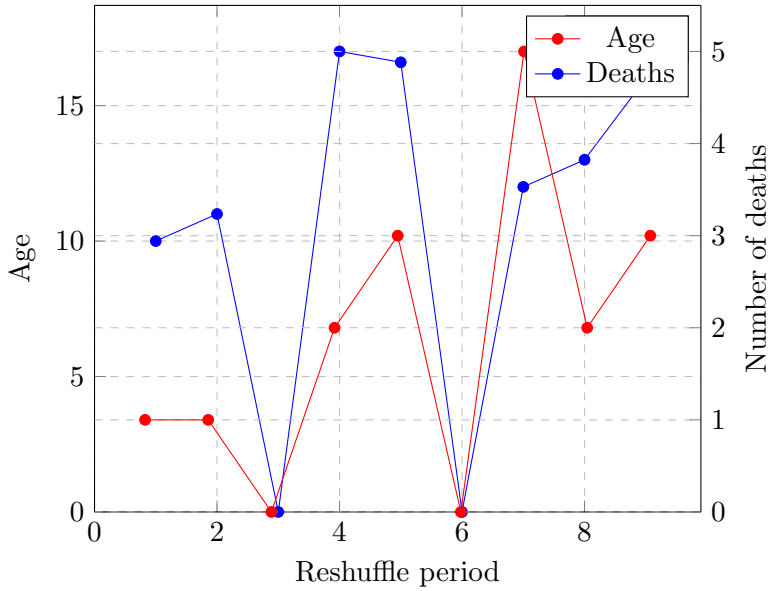


Figure 7.5: Average age upon death and number of deaths vs reshuffle period

Agent type	Team 2	Team 3	Team 4	Team 6	Team 7
Team 5 deaths	25	5	0	25	12
Other agent deaths	19	5	0	35	8
Organisation?	NO	67 days	0 days	NO	62 days

Table 7.4: Team 5 interactions with other agents

Self-Organisation With Other Agents

The overall aim within the tower is for agents to self-organise and create a stable society. Tests were run to investigate how compatible the Team 5 agent is with each of the agents designed by each of the other teams. Simulations were run with a tower consisting of Team 5 agents and one other agent type, with seven agents each and five food available per agent, to see whether they could self-organise within 100 days. These results are shown in Table 7.4.

Team 5 is able to self-organise with Teams 3, 4, and 7, and can do so immediately with Team 4. Team 5 does not perform well with either Teams 2 or 6; the agents were not able to self-organise within 100 days, resulting in continual deaths.

The Team 5 agent also does not perform well with random agents (i.e. agents which take a random amount of food). Experiments on towers consisting of Team 5 and random agents resulted in a comparatively high number of deaths, with a low average age upon death. The average age also does not improve when the food per agent ratio is increased.

7.6 Conclusion

The Team 5 agent is capable of achieving self-organisation within the tower provided the simulation parameters are not too strict. A large number of agents within the tower greatly inhibits the agent's ability to self-organise, but for low agent numbers, they are capable of organising when the food per agent ratio is at least 2. This demonstrates the reliance on communication

and sharing of knowledge in order to solve the problem of long-term survival in the tower.

During the presentation for this coursework, Professor Pitt compared the Team 5 agent's strategy to theory from Ober, who stated that "alignment is an epistemic process, predicated on the right people having and using the right information in the right context" [30]. We had not considered this theory when designing our agent, however in a sort of convergent evolution we observed this in the randomeness and large variation in the Team 5 agent's ability to self-organise, and the conditions required for this to be successful.

Chapter 8

Team 6 Agent Design

8.1 Overview

In this chapter, we present the technical formulation and results of the Team 6 agent. This chapter is divided into seven parts and is structured as follows:

1. Specification of the agent structure (Section 8.2).
2. Introduction of the underlying concepts upon which the agent's behaviour is based (Section 8.3).
3. Demonstration of how the agents utilise common pool resources according to the concepts defined in the previous section (Section 8.4).
4. Explanation of how a social network is formed through the concept of trust (Section 8.5).
5. Exploration of how the agents communicate and interact with each other in order to self-organise (Section 8.6).
6. Presentation and discussion of experimental results through simulations (Section 8.7).
7. Conclusion and future work (Section 8.8).

Ultimately, we propose that this agent accurately parallels aspects of human nature through its capability to undergo behavioural change and formulate social connections with trust, and heavily and intelligently utilises treaties with an outlook to self-organisation.

8.2 Specification of Agent Design

All agents $i \in A$ are implemented as a data structure, such that all agents contain sufficient parameterisation to participate in the various communication methods $c \in C$, resulting in a set of interactions defined by $\mathrm{I} = < A, C >$. Each agent inherits from the `baseAgent` structure and also contains the fields contained in Table 8.2. We note only the most relevant fields for quantifying the agent have been included.

Parameter	Range
`BaseBehaviour`	$N \in [0,10]$
`Stubbornness`	$N \in [0,1]$
`MaxBehaviourSwing`	$N \in [0,10]$
`ParamWeights`	{ `HPWeight`:N, `FloorWeight`:N }
`FloorDiscount`	$N \in [0,1]$
`MaxBehaviour`	$N = 10$
`PrevFoodDiscount`	$N \in [0,1]$
`MaxTrust`	$N = 25$

Table 8.1: Parameters held in the `Config` data structure

Parameter	Range
`BaseAgentRef`	`*baseAgent`
`Config`	textttconfig{} (see Table 8.1)
`CurrBehaviour`	N
`MaxFloorGuess`	N
`AverageFoodIntake`	N
`ShortTermMemory`	$[N]$
`LongTermMemory`	$[N]$
`ProposedTreaties`	[`Treaty`]
`TrustTeams`	map{ *UUID*, N }
`Neighbours`	{ `above`:*UUID*, `below`:*UUID* }

Table 8.2: Parameters held in the `Agent` data structure

8.3 Social Motives

8.3.1 Social Motives Spectrum

The initial concept upon which this agent's behaviour is based revolves around the four possible social motives [31] that this agent can have based on different principles, practices, or characteristics. This in turn leads to a "mixed-motive" setting [32] in the tower. These social motives are:

- **Altruism:** The disinterested and selfless concern for the well-being of others. An altruist then acts in a way that purely benefits others, even if it means harming themselves.
- **Collectivism:** The practice or principle of giving a group priority over each individual in it. A collectivist then acts in a way that benefits the group, themselves included, over purely the individual.
- **Selfishness:** Being concerned excessively or exclusively with oneself. A selfish agent will act in a way to satisfy themselves, but not necessarily with the intent to harm the other agents.
- **Narcissism:** An excessive interest or admiration of oneself. A narcissistic agent will act in a way that not only benefits themselves, but also hinders the collective.

For this implementation, we assert that all agents' social motives can be defined on a spectrum, with one end corresponding to pure altruism and the other end corresponding to pure narcissism. This is represented as a continuous value between 0.0 and 10.0 that shows where an agent lies in the social motive spectrum. Using a continuous rather than discrete variable for the social motive allows for greater expression: for example, two agents may both be referred to as collectivists, but one may be significantly closer to turning selfish than the other. A number closer to 0.0 corresponds to altruistic behaviour, and a number closer to 10.0 corresponds to narcissistic behaviour. The boundaries for the four distinct social motives are shown in Figure 8.1. The initial "natural" state of a given agent and its current social motive are also shown and labelled `baseBehaviour` and `currentBehaviour`, respectively. To begin, we set `currentBehaviour` = `baseBehaviour`.

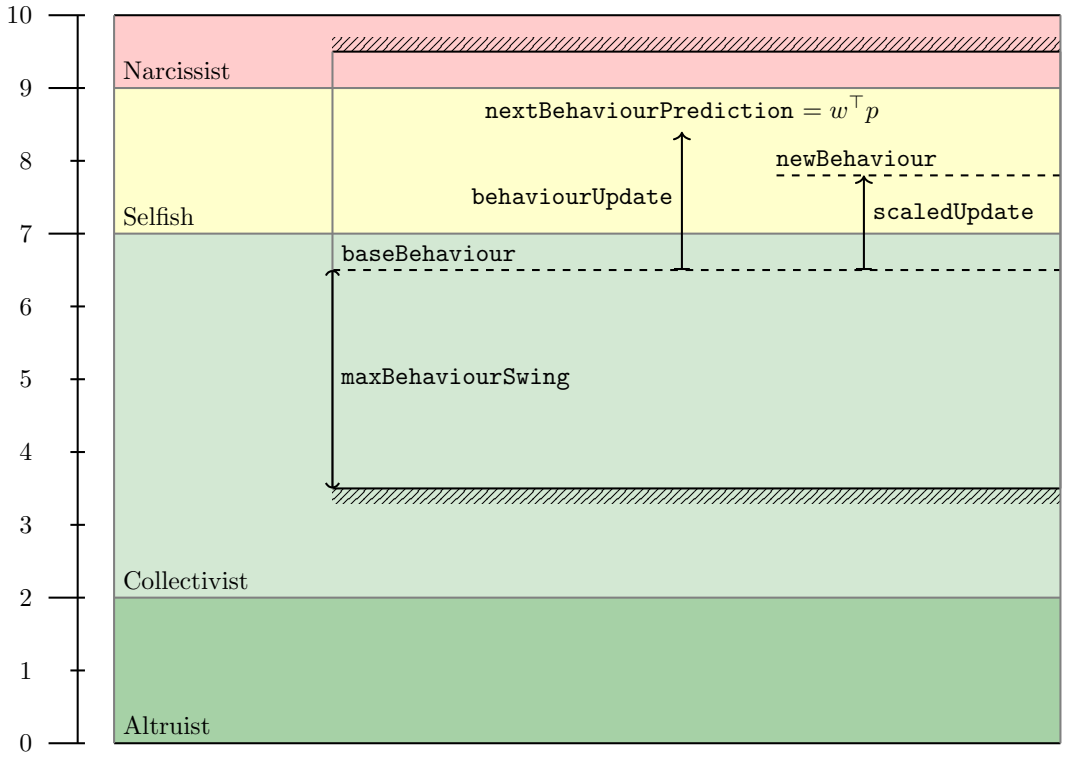


Figure 8.1: Spectrum of social motives. An agent's behaviour is dependent on where it resides within this spectrum. Agents can also change throughout the game as signified by the arrows. Willingness to change is introduced to scale this behaviour update and bounds are placed to prevent extreme changes.

8.3.2 Changing Social Motives

Placing an agent into one of these fixed categories for the entire duration of the game would be limiting and unrealistic. This led to the addition of the first layer of complexity: given that an agent may be initially assigned a social motive, it should be plausible for the agent to change their social motive based on their experience. This concept brings forth the interesting duality of "nature vs nurture" [33]. One could also think about this in terms of "genotype vs phenotype". For example, an agent may 'naturally' or 'genetically' be born a collectivist but may be incentivised to change its values and act selfishly if it becomes desperate due to a lack of food. In this environment, the factors that influence an agent's social motive are its current HP and current floor, as both reflect an implicit lack of food.

The process of changing an agent's social motive based on certain factors can be represented as a weighted sum of these factors. In this case, where the related parameters are the current HP value and the current floor, the weighted sum can be achieved by defining a parameters vector p and a weights vector w. The weighted sum is then given by the product $w^\top p$. These are shown below:

$$\begin{gathered} p = [currentHP, currentFloor], \quad w = [weightHP, weightFloor] \\ w^\top p = weightHP \cdot currentHP + weightFloor \cdot currentFloor \end{gathered} \tag{8.1}$$

Feature Transformations

Instead of using the values of `currentHP` and `currentFloor` directly, we use transformation functions to map them into an output domain between 0 and 1. These transformations additionally allow us to associate a non-linear importance to a given factor. We refer to the resulting factors as `hpScore` and `floorScore` respectively.

The *hpScore* is calculated as

$$hpScore = 1 - \frac{currentHP}{maxHP} \tag{8.2}$$

yielding a simple linear mapping (`HP` $\in$ [0,100] $\rightarrow$ [1,0]), where a higher `currentHP` maps to a lower `hpScore`, resulting in a tendency for the agent to behave more altruistically.

Calculating the `floorScore` requires knowledge of the maximum floor in the tower. As this information is not available to the agent, it will use its memory of the past floors it has been on to forecast this value. Furthermore, the mapping is non-linear, with lower floors (higher number) being 'punished' more; how much more is determined by the parameter λ, where a greater λ weights lower floors disproportionately more. This gives:

$$floorScore = \frac{e^{\frac{\lambda \cdot currentFloor}{maxFloor}}}{e^{\lambda}} \tag{8.3}$$

A lower floor (higher floor number) leads to a higher `floorScore` resulting in a tendency of the agent towards narcissism.

Dynamic Weights

This agent offers an additional level of complexity regarding the way it weights the utilities of both its current health and floor in the tower. We propose the idea that if an agent is consistently evaluating the utility of one of the aforementioned parameters as low, they should assign it a higher weighting so as to encourage a greater shift towards narcissistic behaviour for self-interested survival.

To achieve this, we update the behavioural weights as follows:

- If the agent's health is less than 20:
  1. `weightHP` $\leftarrow$ `weightHP` + 0.05
  2. `weightFloor` $\leftarrow$ `weightFloor` $-$ 0.05

- If the agent's average food intake is less than 1:
  1. `weightHP` $\leftarrow$ `weightHP` $- 0.1$
  2. `weightFloor` $\leftarrow$ `weightFloor` $+ 0.1$

In addition, we use sufficient boundary conditions to ensure that the individual weights consistently fall within the range [0,1].

Behavioural Update

Based on the mapping of social motives from 0.0 to 10.0, a lower floor (i.e., greater floor number) and a lower HP push the social motive variable to be higher. The weights are hence chosen to reflect these correlations (initialised as 0.3 resp. 0.7), along with the relevant feature transformations (Section 8.3.2) applied to these factors.

This weighted sum represents the prediction of the agent's next social motive, named `nextBehaviourPrediction` in Fig. 8.1. We therefore calculate:

$$\texttt{behaviourUpdate} = \texttt{nextBehaviourPrediction} - \texttt{currentBehaviour} \tag{8.4}$$

For the first iteration, `currentBehaviour` = `baseBehaviour` as presented in Fig. 8.1.

8.3.3 Willingness to Change

Although the above encapsulates an essential human characteristic (the ability to change), another important element, the willingness to change, is missing. Through this next layer of complexity, we attempt to quantify how resistant an agent is to altering its behaviour. This concept brings forth more freedom for modelling different human characteristics and realistic situations. For example, it is now possible to distinguish between an agent that is initially an altruist, but may then become selfish as the supply of food diminishes, and an agent that is a "true altruist", which retains the same social motive throughout. Similarly, this is extended to the distinction between a true narcissist agent that never changes, and a narcissist agent that may, during the game, witness others' suffering whilst itself has an abundance of food and therefore steer towards a more collectivist behaviour.

To represent the willingness of an agent to change its social motive, an additional parameter, labelled as `stubbornness`, is introduced and instantiated as a number between 0.0 and 1.0. This number is used to scale the behaviour update. This operation is similar to a step-size, and reflects how easy it is for the social motive to change. We therefore calculate the `scaledUpdate` and `newBehaviour` as shown below.

$$\begin{aligned} \texttt{scaledUpdate} &= \texttt{behaviourUpdate} \cdot (1 - \texttt{stubbornness}) \\ \texttt{newBehaviour} &= \texttt{currentBehaviour} + \texttt{scaledUpdate} \end{aligned} \tag{8.5}$$

Again, `currentBehaviour` = `baseBehaviour` for the first iteration.

8.3.4 Bounding Change

The agent we propose is currently assigned an initial social motive and is then able to change throughout the duration of the game. The ease with which it changes is also quantified through its willingness to change. This brought forth the possibility that even though an agent might be stubborn/resistant to change, it is still, given a long enough period, able to go from a narcissist to an altruist. This, considered to be unrealistic, lead to the final addition of complexity concerning the prevention of extreme changes in social motive. Prior to the start of the game, we define how far an agent can steer away from their 'genetic' or 'natural' initial state.

Bounding social motive change is modelled through the addition of the parameter `maxBehaviourSwing`. This parameter allows for the definition of the maximum and minimum values of the allowed social motive for an agent, thus defining a range:

$$\begin{aligned} \texttt{max} &= \texttt{baseBehaviour} + \texttt{maxBehaviourSwing} \\ \texttt{min} &= \texttt{baseBehaviour} - \texttt{maxBehaviourSwing} \end{aligned} \tag{8.6}$$

If the change as defined in Section 8.3.3 results in a `newBehaviour` outside this range, then this is automatically capped off at the edges. Naturally, this also holds for the spectrum edges $[0, 10]$.

8.4 Food Consumption

8.4.1 Variation by Social Motive

With the agent's behavioural theory defined, its corresponding strategy can be expressed based on the defining qualities of its social motive. As the social motive varies from Altruist towards Narcissist, individual utility is valued increasingly more than collective utility, resulting in an increase in food consumption. Whilst social motives are defined using a continuous spectrum, the agent's consumption and communication strategies are defined on bins along this spectrum, with one bin for each of the four discrete social motives. Thus, which strategy is carried out is decided in a switch-case statement on the agent's current behaviour.

Altruist

An altruistic agent will always take 0 food, as it is only concerned for the well-being of others with a total disregard for itself.

Collectivist

A collectivist agent will consume just enough food to survive, and consume no food when not in danger of dying. Based on the implemented health function, the agent is only in danger of dying when in the Critical region, and will die if it spends $N = \texttt{maxCritical}$ days in this zone without consuming the minimum required units of food to return to the `WeakLevel`. The collectivist agent, having entered the Critical region, will consume food on a randomly assigned day within the allowable period to leave the Critical region. The day of consumption is randomised to avoid the case of all agents needing to consume food on the same day, as there is not enough food on the platform to satisfice every agent in the tower at one time.

- If this agent has health $\geq$ `weakLevel`:
  1. Assign `allowedFoodTakeDay` to `rand(0, maxDaysCritical)`
  2. Take 0 food
- Else if this agent has health $\geq$ `criticalLevel`:
  - If `allowedFoodTakeDay` matches the number of days that this agent has stayed critical: take 2 food
  - Otherwise: take 0 food
- Default: take 0 food

Selfish

A selfish agent consumes the amount of food required to remain in a healthy state. A healthy state is arbitrarily defined as between 30% and 60% of the maximum possible health. These values are assigned to variables `healthyLevel` and `strongLevel`, respectively, with the selfish agent operating accordingly under these three conditions:

- If the agent's health is greater than or equal to the `strongLevel`: take 0 food
- If the agent's health is between `healthyLevel` and `strongLevel`: take enough food to maintain the current HP value
- If the agent's health is less than the `healthyLevel`, the agent takes enough food to reach the `healthyLevel`.

Narcissist

A narcissistic agent will take maximum amount of food consumable, since it is purely be concerned for its own well-being, to the chagrin of other agents.

8.4.2 Final Implementation

The actual amount of food consumed by the agents in the final implementation does not solely depend on the social motives and the current health level, but also on other factors, such as trust, and commitments on messages and treaties. The dependency on these other factors will be explored in the corresponding sections.

8.5 Formation of Social Network through Trust

8.5.1 Background

With treaties offering a notion of socially accepted institutional power, the formation of treaties must be handled carefully so as not to disadvantage the agents who engage in them. For this reason, a social network must be constructed to offer a heuristic for joining into these agreements.

Trust serves as "an important role for the formation of coalitions in social networks and in determining how high value of information flows through the network" [34], with Ostrom further supporting that "trust is an essential social lubricant" [35]. Ostrom further asserts that trust "affects the willingness to cooperate" [35] and "enhances cooperation in social-dilemma experiments" [35], providing clear reasoning that trust is an effective parameter to introduce to the agent structure.

Finally, it is through this introduction of trust that we demonstrate the assertion from Adali's work [34] that trust facilitates "communication behaviors which are statistically different from random communications".

8.5.2 Initialisation

The agent takes a simple algorithmic approach to formulating a social network: first, the ID of the neighbouring agents is determined through message passing; this ID is mapped to an initial trust value which is continually updated through subsequent social interactions.

This agent implementation offers different initialised trust values based on the social motive, using the general notion that the more narcissistic an agent is, the less likely it is to have an initial positive opinion of their neighbours.

For this reason, we give a maximal range of trust from -25 to $+25$ with the initialisations as follows:

- Narcissist: -5
- Selfish: 0
- Collectivist: $+5$
- Altruist: $+10$

8.5.3 Causes for Trust Update

It is through continuous social interaction in the tower that agents will be able to form opinions of the other agents. For this agent, there is a specific set of interactions that will result in a opportunity for trust to update. These interactions impose a *base* behavioural update, before scaling due to social motive (Section 8.5.4). This behavioural update is defined as a relative change to the current trust value.

Treaties

1. When an agent receives a response to the treaty it has proposed:
    - If accepted: $+3$
    - If rejected: -2
2. When an agent receives a treaty proposal:
    - If the agent accepts the treaty: $+2$
    - If the agent rejects the treaty: -1

Communications

1. If an agent makes a request for another to take 0 food (Section 8.5.3):
    - If accepted: reset to 0
    - If rejected: -1
2. If an agent is requested to leave food:
    - If narcissist: -1
    - Otherwise: $+0$
3. If an agent is requested to take a certain amount of food: -1

Case of Low Trust

If an agent holds an extremely low opinion of another agent (trust ≤ -10), it will send a message asking the other to take 0 food for one turn, so as to restore trust and show repentance. If this agent accepts the request, it is forgiven and has its trust value reset to 0. If not, the opinion of this agent is decremented.

Case of Low Average Food

If an agent continually averages less than one unit of food per day, it makes the assumption that the agent above is acting maliciously. For this reason, the trust of this neighbouring agent is decremented. Conversely, if the agent is averaging above this amount, it will increment its trust in the neighbour above.

8.5.4 Variations in Trust Update

To maintain a strong sense of social cohesion, the trust assigned to another agent must be continually updated to reflect the ongoing exchange of information [34]. Again, this agent implementation asserts that the different social motives will have different responses to positive and negative interactions, with the magnitude of the change in trust varying proportionally. This scaling is documented below.

In the case of a positive exchange, the scaling of the trust update is as follows:

- Narcissist: $\times 0$
- Selfish: $\times 1$
- Collectivist: $\times 2$
- Altruist: $\times 4$

If the agent is currently acting as a narcissist it is impossible to increase its opinion of another agent. This agent must undergo considerable environmental change to update its behaviour towards the altruistic side of the spectrum to increase trust.

Conversely, in the case of a negative exchange, the scaling of the trust update of another agent will *decrease* as follows:

- Narcissist: $\times 4$
- Selfish: $\times 2$
- Collectivist: $\times 1$
- Altruist: $\times 0$

8.6 Approach to Treaties

Messages allow for short-lived communication between two neighbours. With request and response messages, such as "leave X food on the platform", agents can organise locally. However, to establish a stable, self-organising system across many floors and many reassignment periods, a more sophisticated form of agreement is necessary. For this reason, an agent's strategies rely heavily on treaties, mimicking research findings from the field of behavioural psychology.

To successfully handle treaties, an agent must:

1. Rate treaties
2. Change its behaviour based on accepted treaties
3. Propose treaties

8.6.1 Types of Treaties

As food is a scarce resource in the tower, the main purpose of self-organisation should be to limit how much food agents eat. By handling all possible conditions, ("less than", "less than or equal", "equal to", "greater than or equal to", "greater than"), this strategy focuses on assigning upper bounds to the amount of food eaten.

To ensure that an agent has access to food on a given turn, it is integral for the neighbouring agents to `leave` food, wherein the infrastructure of the tower provides multiple types of treaties: `LeaveAmountFood`, `LeavePercentageFood` and `Inform`. To enforce localised sanctions and suggest an intake of food to allow stability, however, what really counts is the amount of food taken. Thinking in terms of the `LeaveFood` types limits how much treaties can distribute through the tower: `LeaveAmountFood` would allow the first few agent floors to eat as much as desired without breaking the treaty. Once this amount is reached, any subsequent agent does not receive anything. Similarly, `LeavePercentageFood` limits agents by exponentially decreasing portions, such that low levels have little chance of surviving even if everyone satisfies the treaty. `TakeAmountFood` does not face this issue as it limits consumption no matter the amount of food an agent has access to. Consequently, the agent thinks in terms of the amount food it takes, converting any other treaty type to an equivalent upper bound of food it is allowed take. This makes it easier to compare treaties and make sure that accepted treaties are not contradicting one another.

Having clarified the general interpretation of treaties, we now describe the two main underlying principles with which we handle treaties: Forecasting and Utility Theory.

8.6.2 Forecasting for Resource Availability

Theory

Treaties do not have any immediate effect, but instead influence the future consumption of an agent. In order to assess the present value of a treaty, an agent greatly benefits from the ability to predict its future.

In the case of this tower, "future" can mean two different things: the future of an agent on the current level and their future on a series of unpredictable levels after that. An example of this may be that an agent very high up in the tower can expect to receive a surplus of food in the coming days, but should not expect the same in future reshuffling periods. The future on this floor is best predicted by the previous experience on this floor, while the future thereafter is best predicted by all previous experience in the tower. Accordingly, we assert that each agent separately keeps track of the amount of food it receives each day during the current reassignment period as well as through its entire lifetime. This aligns well with the core assumption in cognitive psychology that there are separate systems for long- and short-term memory [36].

Implementation

To implement forecasting, we use two separate arrays corresponding to long-term and short-term memory and store the amount of food received each day (Fig. 8.2).

The short-term memory is reset after every reshuffle and a scaled histogram illustrating the number of times the agent has received a certain amount of food results in a discrete distribution that can be used for predicting the expected food in the future.

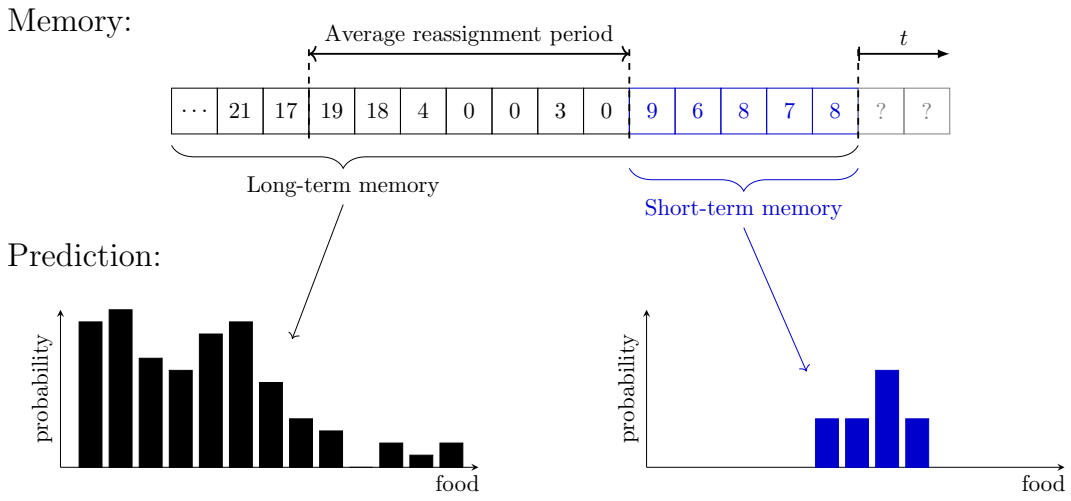


Figure 8.2: Forecasting of the estimated food received in the short-term and long-term using two integer arrays.

In order to distinguish between the short and long term, this agent also forecasts how long it will stay on the current floor. It does so by averaging the duration of previous reshuffling periods.

8.6.3 Utility Theory

Memory allows an agent to remember the past; forecasting uses this to predict the future. In order to make a decision regarding treaties, one last step is required: the agent needs to be able to compare two potential futures and decide which one it deems more beneficial.

Behavioural research has shown that the satisfaction of gaining or losing wealth is non-linear [37] and utility functions are commonly used to account for this by mapping the monetary value of a good or service to an individual's preference [38]. To accurately model human decision making, this agent uses utility functions to make decisions regarding food as the one scarce good in the tower.

We program this agent to use utility functions as a mechanism to compare which of two different futures is more beneficial. To achieve this, the agent calculates the expected utility both with and without a treaty, subsequently maximising its estimated future benefit by choosing the larger expected utility. The expected utility from gaining a certain amount of food a is deterministic, hence:

$$E[U(a)] = U(a) \tag{8.7}$$

The utility of gaining an uncertain amount of food represented by the discrete random variable x (based on past experience), is computed with:

$$x = [p_1 : x_1, p_2 : x_2; ... \; p_n : x_n] \tag{8.8}$$

where an agent expects to get amount x_i of food with probability p_i, such that

$$E[U(x)] = p_1 \times U(x_1) + p_2 \times U(x_2) + ... + p_n \times U(x_n) \tag{8.9}$$

Prospect Theory

Comparing potential future outcomes with utility allows for the introduction of known psychological behaviours to the agents. Prospect theory, first established by Kahneman and Tversky [39], is a well-established model of how a change in value is perceived or, alternatively, how much utility is gained or lost from a change in value (Fig. 8.3).

This model comprises three main principles:

- **Greediness:** Humans are generally greedy, meaning that more of something is always at least regarded as beneficial. Utility functions are hence generally increasing.
- **Diminishing sensitivity:** Marginal returns are strictly decreasing, thus, the greater the personal wealth of an agent, the less they value the resource. An additional unit of food matters much more to an agent if it only gets one unit of food compared to if it gets 50 units of food.
- **Risk aversion:** Humans generally try to avoid risk. The discrete random distribution which forecasts the expected food received does not guarantee any certain amount of food and so is associated with risk. With risk aversion, the amount of food the agent perceives as equivalent to a random distribution, its *certainty equivalent* C, is hence less than its mean.

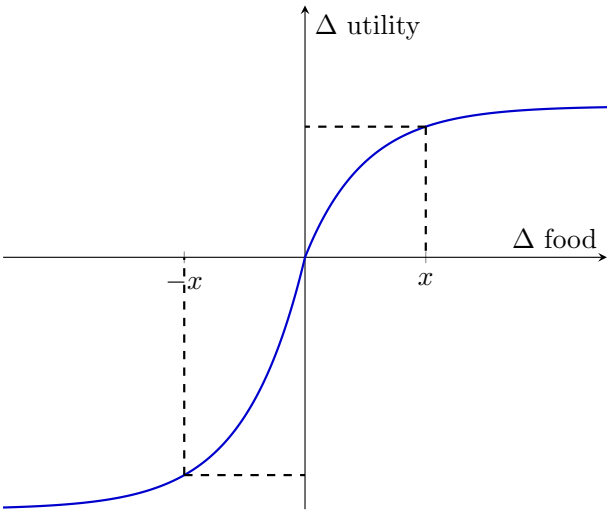


Figure 8.3: Utility in decision making according to prospect theory.

$$U(C) = E[U(x)] < U(E[x]) \tag{8.10}$$

- **Loss aversion:** Losing some amount of food is generally perceived as worse than gaining that same amount. To account for this, the agents weigh losing an amount of food stronger than gaining the same amount of food.

8.6.4 Social Motives and Total Utility

Prospect theory forms the basis to how the agents' benefit from a change in consumption is modelled. In addition to the *gain*, however, there is also a *cost* that increases with eating more food: every unit of food gained is one unit of food other agents lose. This social cost, combined with the previously explained explained utility, forms the total utility function of x, the amount of food received, given by

$$U(x) = gx^{\frac{1}{r}} - cx. \tag{8.11}$$

Each of the social motives correspond to different values of greediness, risk-aversion, and social cost, which have been chosen according to three insights:

1. The more selfish an agent is, the greedier it is.
2. The more an agent cares for the greater good, the greater its social cost associated with consumption.
3. Studies have shown that more narcissistic people are generally less risk-averse [40].

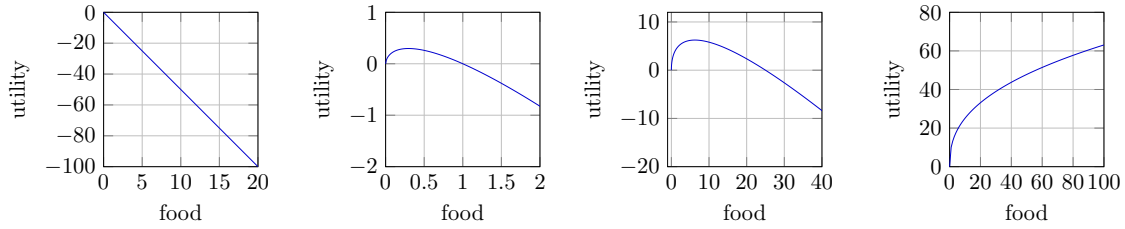
We choose the parameters for each social motive accordingly, with greediness g, risk-seeking r and social responsibility c, resulting in the total utility functions in Fig. 8.4.

(a) **Altruist**: not greedy, very risk-averse, very high social responsibility

(b) **Collectivist**: slightly greedy, risk-averse, high social responsibility

(c) **Selfish**: greedy, low risk-aversion, low social responsibility

(d) **Narcissist**: very greedy, very low risk-aversion, no social cost



(a) Altruist $g = 0, r = 1, c = 5$

(b) Collectivist $g = 2, r = 1.5, c = 2$

(c) Selfish $g = 5, r = 2, c = 1$

(d) Narcissist $g = 10, r = 2.5, c = 0$

Figure 8.4: Different utility functions used to rate treaties according to the social motive of the agents. The associated parameters g, r, c fully define the shapes of each utility function.

8.6.5 Short and Long Term Trade-Off

As mentioned in Section 8.6.2, the futures an agent should expect in the short (on the current level) and long term (throughout the tower) differ: the longer the duration of a treaty, the more the long-term benefit should matter. By using the short and long term memory of food received each day, an agent can separately estimate the resources it can expect to gain on the current floor and how much it can expect to gain (on average) across all future reassignment periods.

By calculating the length of the average reassignment period, the agents can estimate the number of days remaining on the current floor. The agents subsequently use the proportion of estimated days before the next reshuffle period in order to weight how much they should focus on the short term.

Let b_{short} and b_{long} be the estimated short and long term benefit of a treaty, respectively. Also, let the estimated days remaining on the current level be given by $d_{current}$ and the duration of a treaty by d_{treaty}. The total benefit, b_{tot} is given by

$$b_{tot} = d \times b_{short} + (1 - d) \times b_{long} \tag{8.12}$$

where:

$$d = \frac{d_{current}}{d_{treaty}} \tag{8.13}$$

We supply the following example to illustrate this behaviour:

- $d_{current} = 10$, $d_{treaty} = 8 \Rightarrow d = 100\%$ (treaty expires before leaving the current floor)
- $d_{current} = 10$, $d_{treaty} = 20 \Rightarrow d = 50\%$
- $d_{current} = 10$, $d_{treaty} = 100 \Rightarrow d = 10\%$

Survival Instinct

With a constant supply of food, the above weighting between the long and short term benefits is sensible, however if an agent is struggling to stay alive, it must redefine the way it weights these variables. Consequently, agents ignore the expected long-term utility altogether when their health is on a critical level.

8.6.6 Implementation

How agents rate treaties closely follows the theoretical steps from above. Each day, the agent stores the amount of food received in two arrays, one for both the short and long term, with the short-term array reset after each reshuffling period. When a new treaty is proposed, the agent performs the following steps to conclude whether to sign this new treaty or not:

1. Check if the trust in the `proposer` agent is high enough
2. Check if the treaty is compatible with treaties that the agent signed already
3. Calculate the expected short- and long-term utility according to Equation (8.9) using the memory of food received in the past. The utility function used depends on its current social motive.
4. Amplifie the utility if it is negative in accordance with loss-aversion.
5. Estimate the amount of food it can take under the treaty (i.e., the difference between the average amount of food it received in the past and the amount of food it needs to leave in order to comply with the treaty) and calculates the utility of taking that amount of food.
6. Compute the estimated short- and long-term benefits of signing the treaty as $U(\text{sign}) - U(\text{don't sign})$.
7. Choose to focus on the long-term or short-term benefit according to Equation (8.12)
8. Sign the treaty if its overall benefit is positive

The amount of food that the collectivist and selfish agents would need to consume in order to maximise utility varies depending on the current health level. The peak of its total utility function thus needs to be able to vary, too. We account for this by introducing a scaling factor a as

$$U(x) = g(ax)^{\frac{1}{r}} - cax \tag{8.14}$$

where

$$a = \frac{1}{z}\left(\frac{cr}{g}\right)^{\frac{r}{1-r}} \tag{8.15}$$

We then introduce the desired food intake z as the maximum of the function, without changing its general properties. This leads to

$$U\left(x\right) = g\left(\frac{1}{z}\left(\frac{cr}{g}\right)^{\frac{r}{1-r}} x\right)^{\frac{1}{r}} - cx\frac{1}{z}\left(\frac{cr}{g}\right)^{\frac{r}{1-r}} \tag{8.16}$$

8.6.7 Behaving According to Treaties

Treaties are fundamentally built on the assumption that all agents behave truthfully. Introducing dishonesty to treaties would strap them of their power, as there are two many possible falsities but no sufficient way of recognising and no central authority to punish them. Hence, we do not consider deception: a decision shared by all agent teams. Since this means that an agent cannot act against any active treaties, the amount of food taken has to be chosen with care.

We implement a function to return the valid range of possible food intakes given a list of active treaties and accepted request messages and use this both to check if a newly proposed treaty is compatible with previously accepted treaties and to *legally* take food. We do this as in Algorithm 6. An agent then consumes the quantity smaller or equal to the resulting upper-bound `max` that maximises its utility.

Algorithm 6: Function to ensure that an agent never breaks any accepted treaties.

Input : List of active treaties T, RequestLeaveFood message M
Output: Maximum amount of food allowed: **max**

```
1 max ← food received
2 foreach t ∈ T do
3     if agent state ∈ t condition then
4         takeAmount ← convert t request value if takeAmount ≤ max then
5             max ← takeAmount
6     if neighbour requested to leave food then
7         takeAmount ← convert t request value if takeAmount ≤ max then
8             max ← takeAmount
```

8.6.8 Proposing Treaties

Treaty proposals are fundamentally linked to each agent's social motive, given that they work to the benefit of collective utility. The terms of each treaty must be adhered to by both entities that have agreed to it, resulting in a situation where both agents will benefit the same amount. Aligning this with agent strategies, a true narcissistic agent, which would go so far as to behave with the intention of causing detriment to others, would not make any decisions that might benefit its neighbouring agents. As a result of this, the narcissistic agent would not propose any treaties, as even though it might reap the rewards of a treaty, it will never offer other agents any assistance. On the other end of the spectrum, an altruistic agent that always makes decisions based on maximising other agents' utility, would not behave in a way that might benefit itself. This too has the effect of an altruist never proposing any treaties, as although it might offer aid to other agents, this would go against its self-sacrificing nature.

The collectivist and selfish agents are therefore the two social motives that propose treaties. The collectivist agent wishes to maximise collective utility by forming agreements that will increase the chances of itself and its neighbours surviving and the selfish agent wishes to maximise its own survival chances as it knows that agreeing to a treaty will bring personal benefits.

The possible treaties that could be proposed are set based on the different social motives. These were changed for testing purposes (see Section 8.7), however, the basic structure is outlined below:

- Collectivist: If the agent's health is above the `weakLevel`, leave 100% of resources on the platform
- Selfish: If the agent's health is above the `strongLevel`, leave 100% of resources on the platform

Propagating Treaties

Once a treaty has been accepted or rejected by an agent, it is possible for that agent to propose the same treaty to their neighbour. For a given agent, the main motivation behind propagating a treaty is to further increase the benefits of that treaty for that agent. Depending on the agent's social motive, the treaty might benefit their individual utility, the collective utility, or both. Following similar principles used for treaty proposals, we assert that all agents other than narcissists would be active in propagating treaties.

The altruist agent, though it may not accept the treaty, will still propagate it in order to maximise the utility of other agents. The collectivist agent will propagate treaties which it has itself accepted, in the hope that it will further benefit collective utility. Finally, the selfish agent will propagate treaties with the aim of reaping the rewards of multiple treaties to benefit itself. As mentioned, the narcissist will not propagate any treaties as it acts to hinder existing collective utility and destroy other agents.

8.7 Simulation Results and Discussion

8.7.1 Hypothesis

Before presenting the simulation results, we offer the following hypotheses we wish to verify. Given the aforementioned modeling approach, the following affirmations are hypothesised:

1. In a purely narcissist system, it is impossible for an agent to survive on a long-term basis.
2. In a purely collectivist system, a stable state is reached.
3. In a system composed of agents with different social motives, a polystable, i.e., a system that has more than one convergent state, "whose parts have many equilibria" [41], can be reached in a long-term scenario.
4. There exists a threshold on the number of selfish agents that can be incorporated in the system before a stable collectivist system becomes unstable.
5. In a system composed of agents with different social motives, the ability to self-organise through treaties improves the global utility.
6. In a system composed of agents with different social motives, where genetic replacement of the agents occurs, the system converges to a collectivist system.

8.7.2 Summary of Simulations

To examine the different hypotheses introduced in Section 8.7.1, different simulations need to be executed. We divide the simulations into five groups characterised by having different initialisation parameters. Table 8.3 and Table 8.4 summarise the simulation parameters. The percentages of each social motive correspond to the initial distribution. If not explicitly mentioned, we run experiments using 100 agents, with 100 food on the platform for 60 days and with a reshuffle period of 30 days. We also impose that this is a homogeneous tower, implementing only agents of this type (Team 6).

	A1	A2	A3	A4	B1	B2
% Altruist	100	0	0	0	10	5
% Collectivist	0	100	0	0	40	60
% Selfish	0	0	100	0	40	30
% Narcissist	0	0	0	100	10	5
Stubbornness	-	-	-	-	-	-
MaxBehaviourSwing	0	0	0	0	0	0
Communication	yes	yes	yes	yes	yes	yes

Table 8.3: Summary of the experiments A1 to B2. The agents are not able to change their social motives.

	C1	C2	C3	D1	D2	E1	E2
% Altruist	10	10	10	100	100	25	25
% Collectivist	40	40	40	0	0	25	25
% Selfish	40	40	40	0	0	25	25
% Narcissist	10	10	10	0	0	25	25
Stubbornness	0.2	0.2	0.2	0.8	0.8	0.2	0.2
MaxBehaviourSwing	8	8	8	6	6	6	2
Communication	yes	no	yes	yes	yes	yes	yes

Table 8.4: Summary of the experiments C1 to E2. The agents are able to change their social motives.

In addition, cases C3, D2, E1 and E2 have additional specific features. In comparison to C1, C3 introduces more restrictive treaties. This is also the case for D2. In simulation E1 and E2, we use a a concept approaching natural selection to replace the agents that die. More details about these cases will be given in the respective subsections.

Our simulations results are given as the average over 10 repeated simulations for the cases A, B and C. For cases D and E, we want to show specific simulations and therefore do not perform any average calculations.

8.7.3 Simulation Results and Discussion

In this section, we present the results by simulating this system according to Table 8.3. We also discuss and analyse the causes and consequences of these results.

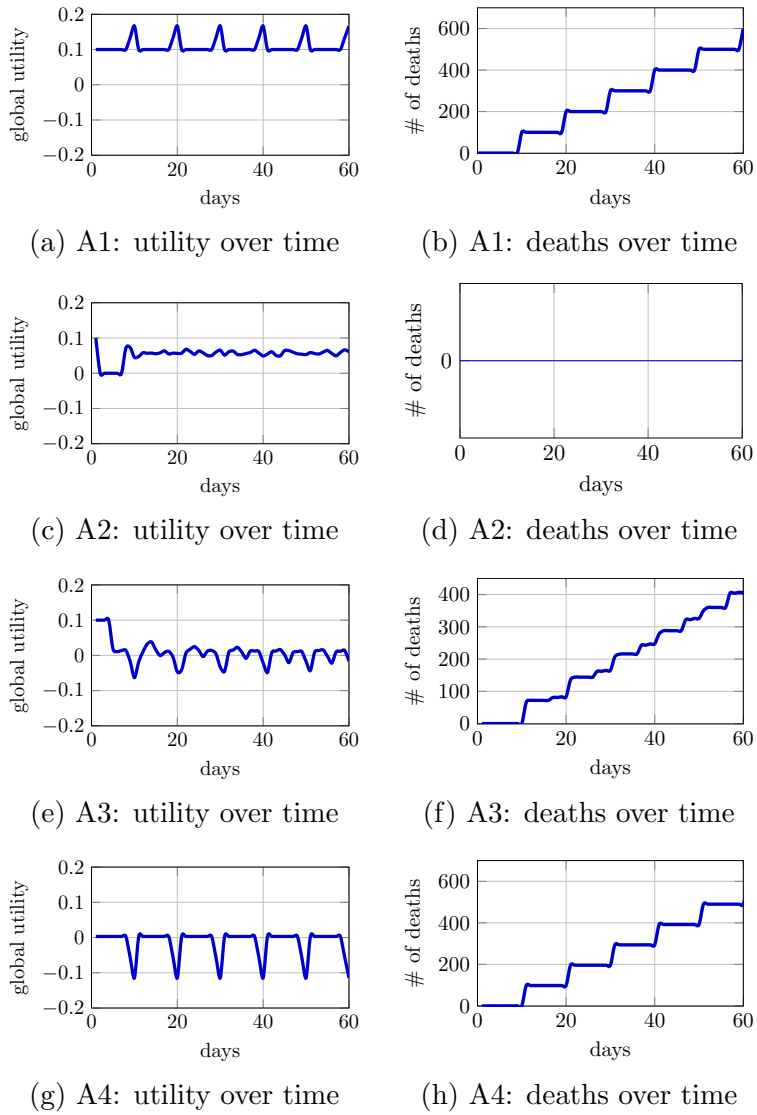
Simulation A: Single Agent Type

The first set of simulations we analyse are simulations that include agents that all have the same social motive. Moreover, these agents do not have the ability to change their social motive. The simulations results are shown in Fig. 8.5.

As expected, a system containing purely altruists (Fig. 8.5 (a) and (b)) leads to the inevitable death of its agents, since by acting purely selflessly, these agents never take any resources from the tower.

As all agents die at the same moment and are all replaced by a new group of altruists, we see a clear pattern in the curve describing the number of deaths over time: after 10 days without eating, all of the agents die, leading to a step of 100 deaths.

This step pattern is coupled with the global utility. Indeed, each 10 days, there arises a spike in



(a) A1: utility over time (b) A1: deaths over time

(c) A2: utility over time (d) A2: deaths over time

(e) A3: utility over time (f) A3: deaths over time

(g) A4: utility over time (h) A4: deaths over time

Figure 8.5: Simulation results for a group of agents with uniform fixed social motive.

global utility. This can easily explained by (2.9), which states that the utility of the each agents will be positive in this case.

Similar to the altruist agents, the narcissists have a large number of deaths among them every 10 days (Fig. 8.5 (h)). This is evidently due to the agents located on the upper floors of the tower taking all of the food, leaving none for the agents below. In comparison to the altruists, a system composed of narcissists will show negative peaks of global utility, as can be seen in Fig. 8.5 (g). This can be explained using the exact opposite reasoning for the altruistic case - (2.9) is negative this time for the agents that cannot eat.

As a compromise between these two systems, a system including only selfish agents presents smoother curves (Fig. 8.5 (e) and (f)). These agents take less food than the narcissists, but still take too much to maintain zero deaths in an economy of scarcity.

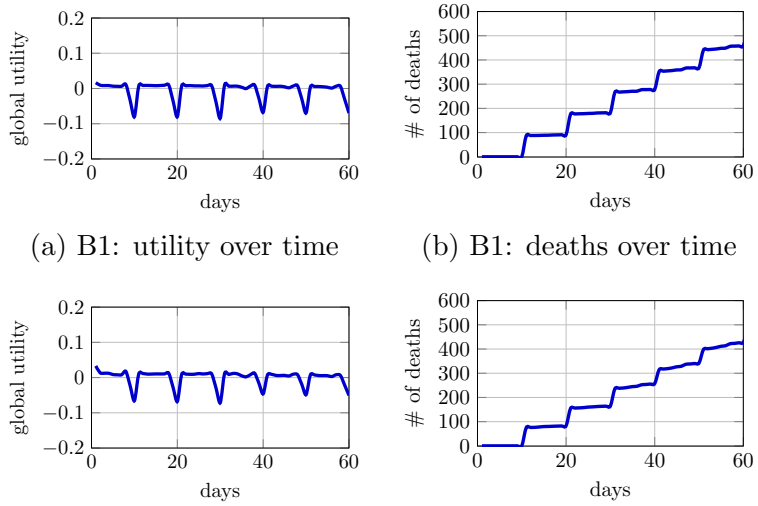
Finally, the collectivists instantaneously achieve a stable society in which none of the agents die (Fig. 8.5 (d)). This is due to their innate strategy regarding food intake that ensures the optimal distribution and intake of the common pool resources.

We also note a uniformly positive curve for global utility over time that is smoother than for the other social motives (Fig. 8.5 (c)). This clearly reflects the increased social cohesion between the agents in the tower and identifies the almost perfect allocation of resources, leading to no wasted utility.

As a conclusion from these experiments, we can now state that Hypotheses 1 and 2, formulated in Section 8.7.1, hold in this context.

Simulation B: Multiple Agent Types Without Behaviour Change

Having assessed groups of agents of each social motive individually in Section 8.7.3, we increase the complexity of the system by having agents with different *fixed* social motives in the tower. The inability for these agents to change their social motive with time leads to the simulation results shown in Fig. 8.6.



(a) B1: utility over time (b) B1: deaths over time

(c) B2: utility over time (d) B2: deaths over time

Figure 8.6: Simulation results for a group of agents with different fixed social motives.

The first conclusion we can draw from these results is that having a larger proportion of collectivists helps the system to perform better in terms of utility and number of deaths. Through comparing B1 to B2, we see that the system comprising a larger amount of collectivists (B2) outperforms the system with a comparatively smaller amount of collectivists (B1). This is to be expected, as the more collectivist agents there are, the more similar to Fig. 8.5 (c-d) the system will be.

A second behaviour illustrated by this simulation is that the action of introducing treaties (Section 8.6.8) is not always relevant. The collectivist agents accept the collectivist and selfish treaty (the collectivist one being more restrictive), but the selfish agents *only* accept the selfish treaty.[1] This way, the two agent types are following their natural strategy concerning food intake (Section 8.4). Knowing this, the system shows similar results with and without treaties, hence we only show the results where communication is allowed.

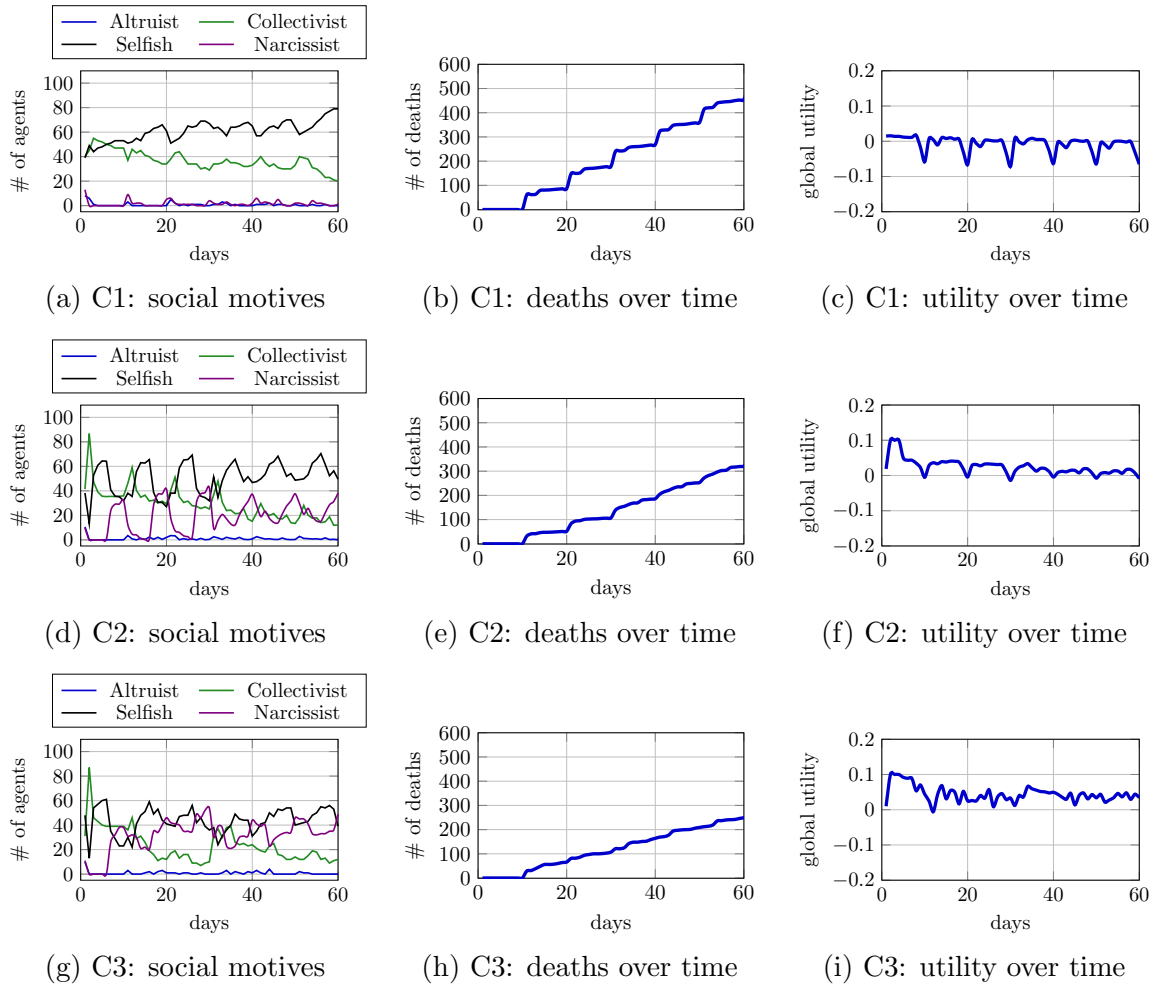
[1]The other types of agents, altruist and narcissist, do not play a role in this analysis. The altruists accept the two treaties and eat nothing, and the narcissist, being even more difficult to convince, will reject the two treaties.

Simulation C: Multiple Agent Types With Behaviour Change

This set of simulations builds on top of the framework set by (B), instead investigating the terminal behaviour of a system comprising different distributions of *fluid* social motives. We provide the agents with a non-zero behavioural swing to investigate the changes in distribution, global deaths and utility over 60 iterations. Moreover, we utilise different levels of communication and treaties to contrast the results using the treaties introduced in Section 8.6.8 (Fig. 8.7 (a-c)). We simulate the system under two other configurations: without considering any form of communication (Fig. 8.7 (d-f)), and by using a different, more restrictive treaty (Fig. 8.7 (g-i)).

This treaty restricts the amount of food its members can take when their health drops below the `WeakLevel`:

If `currentHP` $<$ `WeakLevel`, take ≤ 2 food.[2]



(a) C1: social motives (b) C1: deaths over time (c) C1: utility over time

(d) C2: social motives (e) C2: deaths over time (f) C2: utility over time

(g) C3: social motives (h) C3: deaths over time (i) C3: utility over time

Figure 8.7: Simulation results for case C. C1 and C3 include communication, but C2 does not. C3 includes a more restrictive treaty.

[2]To implement this treaty, we create a new treaty type, in addition to the existing set proposed by the infrastructure team. We see the need for this treaty as it is in general not possible to formulate a request to `take` a maximum amount with a corresponding `LeaveAmountFood` or `LeavePercentFood` treaty. The special case "take 0" food can be expressed as "leave 100% of the food", but there is in general no such equivalence that holds across the whole tower.

The overarching comment to draw from this set of results is the impact of specific treaties on the global utility. By comparing the results C1 and C2, the system using the 'classical' collectivist treaty (C1) (see Section 8.6.8) yields a lower performance than the system without treaties (C2): the number of deaths is lower in C2, and its global utility is higher.

To explain these rather counter-intuitive results, we offer the following hypothesis. As the agents' health falls, their social motives tend to change toward narcissist. In the case of collectivists having signed a treaty, they will turn selfish. Instead of following the natural decision of this social motive (eat to stay above the healthy level), they have to follow the treaty. The moment their health falls below the `weakLevel`, the treaty they signed no longer applies and they will follow their natural food intake rule defined in Section 8.4. If the agent's health is less than the `healthyLevel`, the agent takes enough food to reach the `healthyLevel`.

However, this leads to a lot of wasted resources at this critical health level. Notably, each food intake greater than 2 will not offer additional utility to agents whose health falls below the `WeakLevel`: any food intake greater than or equal to 2 upgrades the agents health to the `WeakLevel`. The resources would have been used in a better way if the selfish agents, which initially signed the treaty as collectivists, would have been able to eat once their health had just risen above the `WeakLevel`.

This waste of common resources induced by agents following the collectivist treaty is arguably due to a poor treaty design. This experiment clearly shows that treaties might also have negative effect on the global utility, although they were originally thought to improve it.

To contrast these results, we can propose a different, more effective treaty. As explained above, simulation C3 introduced a treaty that applies when the agents HP is below the `WeakLevel`. As can be seen in Fig. 8.7 (h) and (i), this treaty allows the total amount of death to be lower, and the global utility to be higher. This shows that well-designed treaties can have a positive impact on the system. However, the treaty specification is not a trivial and straight-forward task, and can even be counter-intuitive.

We also allude to the sense of stability that may be offered with the inclusion of treaties. Through juxtaposing C1 (treaties active) with C2 (treaties absent), we see that C2 is in general oscillatory; selfish and narcissistic agents are distributed inversely whilst supporting a low distribution of collectivist agents (10-20%) throughout the entire simulation. This is in contrast with C1, where the simulation instead stabilises to a relatively continuous distribution of selfish and collectivist agents before bifurcating in the final 10 turns of the simulation.

The presence of ineffective treaties, similar to D1 can be seen to eradicate the population of narcissistic agents, as illustrated by their general absence (<5%) throughout both simulations. We offer that this is due to a sense of individuality that the narcissists exhibit when unconstrained by treaties, allowing for purely self-interested behaviour that can outperform the more 'collectivistic' natures of the collectivist and selfish agents, but ultimately lead to self-destruction.

When utilising communication however, there are more grounds for opinion formation, meaning that narcissistic agents will quickly fall into poor favour with the other agents, leading to less social cohesion. This is supported both with D2, where the narcissistic agents are able to integrate with society so long as they always engage in the relevant treaties, and C3, where the introduction of a harsher treaty yields the same terminal results.

The possible stabilising nature of treaties can also be seen from the immunity to the "death period" (the number of days an agent is able to survive in the critical region before dying) in C1 and C3. Whilst C2 displays clear oscillatory behaviour with a period of 10 days (occurring at the "death period"), this susceptibility to such a period is no longer present in graphs C1 and

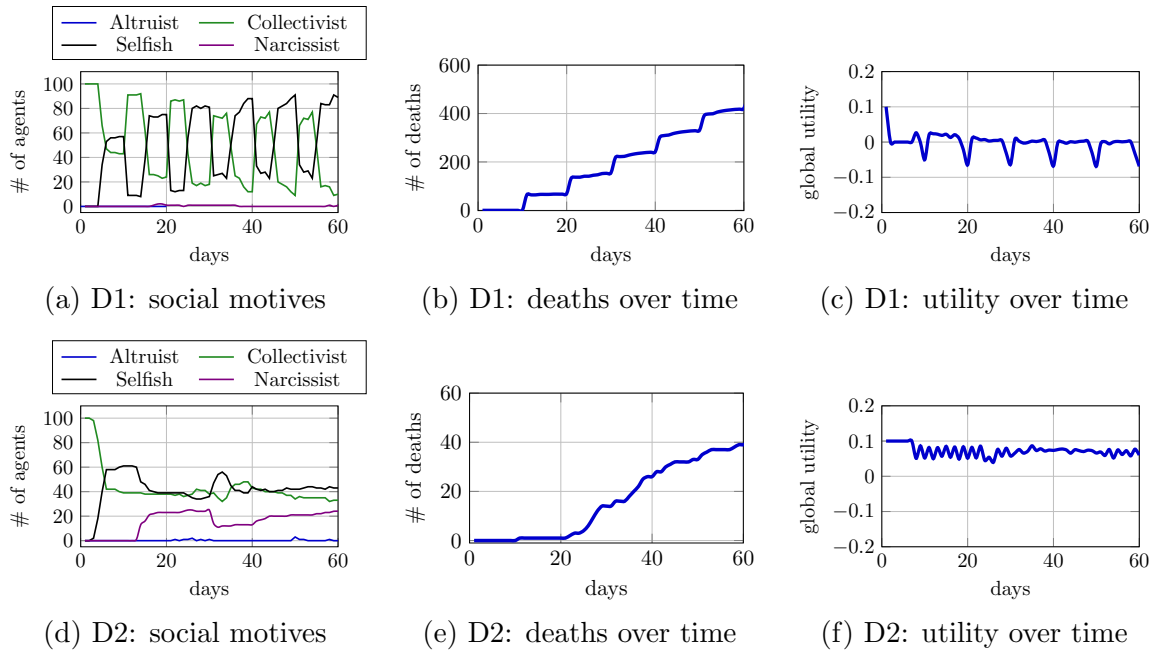
C3, illustrating that treaties offer a sense of immunity. This means that agents will in general not fall within the `criticalLevel` upon injection and be able to remain in the tower for longer.

This set of experiments supports Hypothesis 5, showing that through careful treaty selection, global utility can be increased in comparison to the case where treaties are disabled (C3 resp. C2). However, the exact selection of these effective treaties is not a trivial task.

Simulation D: Destabilisation of a Collectivist System

In these experiments, we initialise the tower's population with collectivist agents only however, unlike with Simulation A (Section 8.7.3), these agents have the possibility to change their social motive over time.

The goal of these experiments is to evaluate if a society comprised solely of collectivists is able to remain stable over time, noting that if this is achievable, we expect to see similar results to the ones presented in Fig. 8.5 (c) and (d). In addition, we investigate the effect of treaties on such a system. The simulation results are shown in Fig. 8.8.



(a) D1: social motives (b) D1: deaths over time (c) D1: utility over time

(d) D2: social motives (e) D2: deaths over time (f) D2: utility over time

Figure 8.8: Simulation results different treaties acceptances.

The analysis of experiment D1 can be done as follows: during the first days in the tower, the agents will agree on collectivist treaties as introduced in Section 8.6.8. As time progresses and their health decreases, some agents change their social motive from collectivist to selfish (Fig. 8.8 (a)). However, once their health falls below the `WeakLevel`, the treaties the agents signed whilst being collectivist no longer apply. This is because, as explained in Section 8.6.8, the treaties proposed by collectivists only hold under the condition $\texttt{currentHP} > \texttt{WeakLevel}$. Therefore, the selfish agents, now unconstrained, consume more food than what a collectivist agent normally would. This wastes utility, ultimately leading to a large number of deaths at the end of day 10 (Fig. 8.8 (b)). This result is similar to the one discussed in Section 8.7.3.

The waste of common pool resources can also be visualised in Fig. 8.8 (c), where the global

utility becomes negative on day 10 (how long agents can survive without eating).

After 10 days, dead agents are replaced by a new group of collectivist agents, leading to a peak in Fig. 8.8 (a). During days 11 to 20, no deaths are seen (Fig. 8.8 (d)), as all selfish agents are still following the treaty they signed when collectivist. Note that the new agents introduced on day 11 sign the collectivist treaties and will not eat any food if their health is above the *weakLevel*, even if they become selfish. On day 20, we note a new peak in the number of deaths, and this process repeats itself. This results in oscillatory behaviour between two quasi-stable states, with convergence to both a high concentration of collectivists and selfish agents in inverse proportions. We hence deem this experiment as 2-phase polystable [41].

To avoid selfish agents from wasting common pool resources when their health drops below the *weakLevel*, the agents sign another, more restrictive treaty during the first days of the simulation as in D3 (Section 8.7.3). We note two interesting results when using this additional treaty.

Initially, even the collectivist agents refuse this new treaty. Without knowledge of the motivation of the agents above, each agent utilises forecasting (Section 8.6.2) based on their memory of the food they have previously received to evaluate the utility of the expectation of food to be received in the future. We propose that this initial treaty refusal is due to the fact that the computed utility of signing the treaty is smaller than the computed utility based on their estimation of the available food they would get without signing the treaty. We note, however, that the memory of the agents is not yet extensive and that the only exposure to the platform thus far is a cornucopia of resources that nobody touches.

To further investigate the importance of treaties, we override the initial decision of the collectivist agents to accept the treaty. This additional personal risk results in an overall benefit for the society, as seen in Fig. 8.8. Quantitatively, the number of deaths over time in this scenario is approximately 10 times smaller than in experiment D1 (Fig. 8.8 (e)), accompanied by a uniformly positive utility over time (Fig. 8.8 (f)). In addition, the initial acceptance of this new treaty also allows for the emergence of a stable distribution of social motives across the tower (Fig. 8.8 (d)).

Despite the presence of selfish (and even narcissistic) agents in the tower, they all follow the rule dictated by the treaties they signed whilst being collectivist[3]. An issue with this configuration is that the exact day when the agents are eating is not synchronised, leading to the selfish and narcissistic agents eating two food as soon as their health falls below the *weakLevel*. We hence conclude that not waiting in the critical zone as collectivists do leads to a non-zero number of deaths.

In addition, we can also see the effect of the reshuffle period on the social motives distribution in Fig. 8.8 (d). The reshuffle period is 30 days in this case and we see that the reorganisation of the tower leads to some of the narcissistic agents updating their social motive to revert to selfish.

The main observation to draw from these results is in the terminal behaviour and the capability to reach stability, with the distribution of all agent types tending to a continuous value (approximately 40% collectivist, 40% selfish and 20% narcissist). We propose that it is this enforcement of treaties that allows for otherwise self-interested agents to act selflessly, instead acting towards a common goal of collectivism. This re-stabilisation of the initially homogeneous collectivist society through treaties is so powerful that it can support a substantial proportion of narcissistic agents, who have in previous experiments (A4, C4) been sufficient for disrupting

[3]The simulation enforces *honest* agents that do not have the ability to break a treaty before the expiration date. We leave the concept of deception for future work.

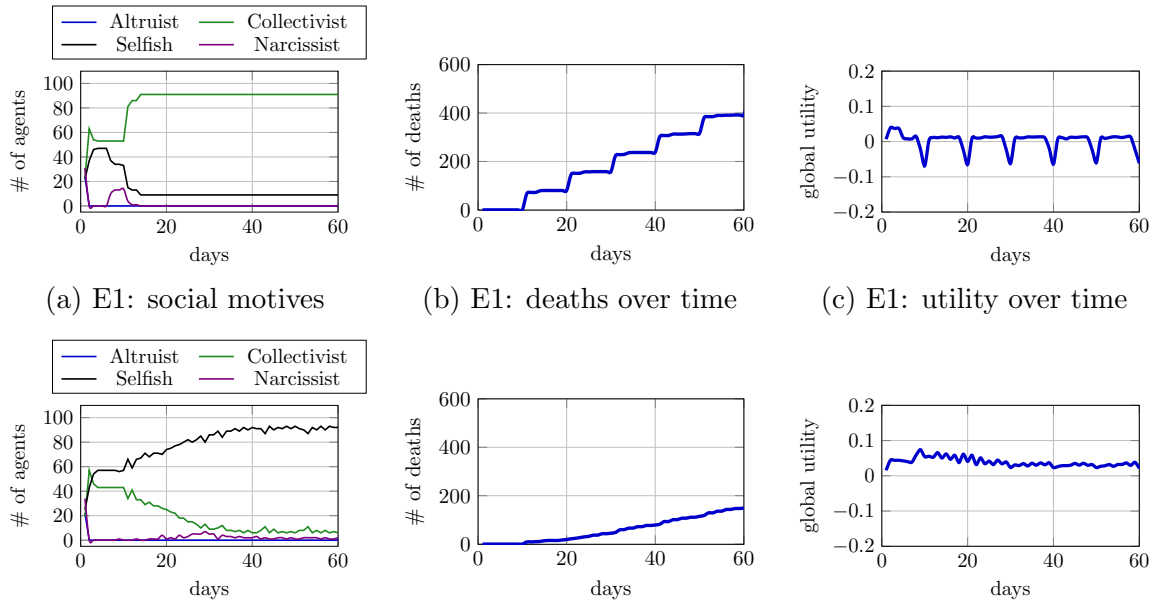
the stability of a system even in low concentrations. We thus conclude that the inclusion of localised governance allows agents to transcend the limitations of impulsive behaviours due to pre-defined social behaviours and self-organise to maintain a stable system that supports all behavioural types.

These results support the emergence of long-term stability, as stated in Hypothesis 3, when using treaties. D1 shows a regular oscillatory behaviour whereas D2 shows a very constant distribution of the social motives.

Finally, these simulations also clearly show that it is not explicitly necessary to inject external non-collectivist agents in the system to destabilize an otherwise purely collectivist system. Collectivists themselves, if they have the ability to change their social motives, will destroy an originally pure collectivist society. This result is even more drastic than Hypothesis 4.

Simulation E: Genetic Replacement of Agents

This set of simulations aims to assess the effect of *genetic replacement* [42] [43] in creating a system that converges to a static distribution of social motives. For this experiment, we re-inject agents into the tower with a probability equal to that of the current distribution; given a tower consisting of 50% collectivists and 50% selfish agents, say, there will be an equiprobable chance of the next agent added to the tower being either of these types.



(a) E1: social motives (b) E1: deaths over time (c) E1: utility over time

(d) E2: social motives (e) E2: deaths over time (f) E2: utility over time

Figure 8.9: Simulation results with genetic replacement of the agents.

Starting with an equal distribution of all agent types (25% of each social motive), we not only investigate the terminal behaviours of the system regarding stability, but whether or not this system can maximise its global utility. As described in Section 8.7.2, E1 and E2 differ by the quantity of resources available in the common pool, with E1 representing a strict economy of scarcity and E2 an economy of abundance. We first present and discuss the results achieved in an economy of scarcity (setup E1, Fig. 8.9 (a), (b), and (c)).

It is clear from Fig. 8.9 (a) that the system E1 achieves a stable distribution of the social motives over time, with a ratio of 90% Collectivists to 10% Selfish agents. This distribution leads to

fewer deaths than in a purely selfish society (see Fig. 8.5 (f)), but significantly more deaths than in a purely collectivist society (see Fig. 8.5 (d)). The global utility shows clear peaks at the end of every critical period of 10 days (Fig. 8.9 (b)).

In a scenario where there is enough food to not only satisfice, but also satisfy each agent on every given day (thus removing an economy of scarcity), the dominant social motive is selfish. This emergence can be attributed to the valuation assigned to treaties: if there is a continual supply of food, treaties offer very little utility to the selfish agents and are hence rejected. Unlike in D1, where the oscillatory behaviour was a product of the selfish agents' inability to survive, we now impose a system where selfish behaviour *can* be supported, due to the higher availability of food. Furthermore, the low number of deaths (Fig. 8.9 (e)) and the relatively high utility (Fig. 8.9 (f)) are not due to the social motive distribution, but instead the amount of food on the platform, which is larger than usual.

To answer the statement introduced in Hypothesis 6, we see with these experiments that convergence of the system strongly depends on the quantity of resources available in the common pool.

8.8 Conclusion and Future Work

8.8.1 Conclusion

In conclusion, we utilise the codification of social motives to offer insight into the general tendency of behavioural changes in an economy of scarcity, as well as the efficiency of treaties in stabilising a population comprising various different social motives.

Irrespective of initialisation, the natural tendency of agents in an economy of scarcity is to make a transition towards the narcissistic end of the spectrum (B-D), leading to a higher overall distribution of selfish agents. Genetic replacement serves as a means of avoiding this behaviour, showing a clear stabilisation to an almost uniformly collectivist population (E1), but is contrasted by a drastic tendency towards selfishness when more resources are made available to the set of agents (E2).

Treaties serve as a stabilising self-organising mechanism in this set of experiments, with appropriately constrictive treaties (C3, D2) even allowing for the stable integration of narcissists into the population, despite their natural tendency to destabilise a system (A4, C2). Treaties may also change a polystable system into a purely stable system, when sufficiently strong as to enforce a collectivist mindset (D1, D2), converting an otherwise oscillatory distribution of social motives into a static distribution.

We hence assert that treaties are a sufficiently powerful self-organising mechanism to mitigate the collective action problem that this platform imposes, as they can be seen to increase global utility (C3, D2) and stabilise global deaths. This affirms that, whilst self-governing institutions cannot be formed, it is through honest following of treaties that an unstable population can be controlled to produce a uniform distribution of social motives.

8.8.2 Future Work

Our future work would focus around adapting the ways in which we model the agents' changes in social motives. One such way is to make agents tend towards altruism, rather than narcissism, when faced with adversarial conditions. This could be interpreted as an understanding of the

agent's environment and the long-term improvement of the individual utility through a short-term sacrifice, in the sense that it is expected that other agents would act similarly, thus bringing the system back to an equilibrium. This would be done by changing the weights used in the social motive update function.

Furthermore, we might imagine a randomly distributed assignment of these weights across different agents. This would illustrate how different agents might react differently to their condition, from which the concept of agent personality could be derived. For example, some agents may encounter a comfortable situation (high HP, high floor) and decide to take advantage of it and act selfishly, while another agent may encounter the same situation and take the opportunity to make a positive impact for their fellow agents below by acting altruistically.

Also, one might make the assumption that the agent's true personality comes forth under beneficial circumstances, whereas a harsh environment is likely to cause the agent to act differently. If the agent's genotype can be defined as the agent's true personality, then the agent's phenotype, i.e., their resulting behaviour, can be modelled by interpolating between this genotype and the social motives based purely on environmental factors. Combining this concept with the random weights assignment may allow for the modelling of both the agent's genotype and their reaction to environmental variables based on their personality.

Finally, we would analyse the effects of a larger number of treaties on the global utility. As shown in this work, the choice of treaties leading to an increase in the global utility is not straightforward. As treaties are expressed in a generic way, it is possible to tune a large number of parameters to find optimal treaties in a given scenario.

Chapter 9

Team 7 Agent Design

9.1 Overview

This chapter outlines the design and findings of Team 7's agent. The primary goal of Team 7 was to emulate human behaviour closely and to replicate the normal variation in personality types exhibited in a typical human population. This would then enable analysis into whether particular personality types exhibited particular behavioural patterns or are more conducive to self organisation and thus acheiving the goal of sustainability in a collective action problem.

9.2 Agent Design

There are three key elements that define the agent: its personality, behaviour and memory. The personality defines a given agent at its core, and for the purpose of this experiment is kept constant throughout a given simulation. An agents behaviour is influenced by its personality, memory (a record of its experiences) and the environment. The environment can include current floor, food available and interaction with other agents. The personality and behaviour then combine to influence the various decisions that the agent makes over the course of the simulation. This basic agent framework is illustrated in Fig. 9.1.

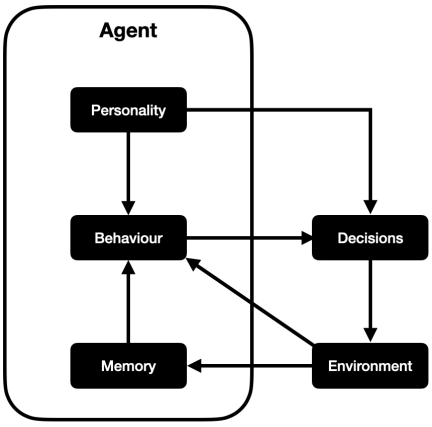


Figure 9.1: Agent framework.

9.2.1 Personality Traits

The use of personality traits to attribute human characteristics to our agents is a construct derived from Robert R. McCrae and Paul Costa's 5 factor theory of personality. McCrae and colleagues discovered that the big five traits are surprisingly universal which was confirmed by a study that looked at people from more than 50 different cultures. They determined that the five personality traits could be used to accurately model the human psychological framework.

The five personality dimensions consist of:

- **Openness**: The degree to which an individual is willing and eager to have new experiences. This also has implications for creativity and problem solving. An individual with low openness will behave more conservatively and will be less welcoming to change.
- **Conscientiousness**: The degree of conscientiousness of an individual determines their level of organisation, discipline and thoughtfulness. Conscientious individuals are also more strategic and forward thinking.
- **Extraversion**: Extraversion is the extent to which an individual is willing and eager to socialise and interact with other individuals. An extroverted individual is more likely to occupy positions of leadership and influence.
- **Agreeableness**: An individual with high agreeableness will exhibit greater degrees of kindness, trust, and altruism. On the other hand, an individual with low agreeableness will demonstrate a lack of sympathy and might be spiteful or manipulative towards others.
- **Neuroticism**: The neuroticism trait dictates the emotional stability of an individual. A greater value of this is associated with greater proneness to anxiety, irritability and mood swings. In the extreme case, individuals can experience regular fluctuations in their character. Therefore, an agent that is high in Neuroticism will commit actions that may seem unreasonable or illogical.

Any time a new agent is deployed it is initialised with a value for these traits ranging from 20-80. We did this in order to exclude extreme personality traits on either end. The personality traits assigned to an agent are innate and unchanged for the duration of the simulation. This is done in order to emulate how traits remains fairly constant over a person's life.

9.2.2 Behavioural Traits

In contrast to personality traits, behavioural characteristics of an agent change over the course of the simulation. The behaviours are initialised based on the agent's personality traits and can range from 0-100 over the lifetime of the agent. These behaviours are then updated daily by environmental factors such as food availability, floor changes and interactions with other agents.

The behaviour characteristics are described below:

- **Greediness**: This is the degree to which an agent behaves selfishly, specifically with regards to the amount of food it takes. The greediness value is initialised by the agent's agreeableness score meaning that high agreeableness will result in low initial greediness and vice versa. The greediness value is subsequently adjusted throughout the simulation run by the following factors:

- Being in a critical health state increases greediness.
- Eating no food increases greediness by a quadratic factor with respect to the number of days since the last food intake.
- Agents display increased levels of greediness in response to being placed at a lower floor during the reshuffle and vice versa. The degree to which this changes is dependent on the openness of said agent.
- Based on past experience, if the agent expects food shortages on a new floor, there will be an additional boost to greediness. This adjustment has a greater impact if the agent has high level of conscientiousness.
- Neuroticism can cause sporadic daily fluctuations in the greediness of an agent. A higher neuroticism results in more substantial daily changes of the relevant behaviours.

- **Kindness**: An agent with a high kindness value will demonstrate greater sympathy for other agents and will be more supportive of the wider social cause. Kinder agents will take less food and will be more likely to collaborate with other agents.

  Similar to greediness, kindness is initialised by the agent's agreeableness score meaning that high agreeableness will result in high initial kindness. The kindness value is subsequently adjusted throughout the simulation run by the following factors:

  - Agents display increased levels of kindness in response to being placed at a higher floor during the reshuffle and vice versa. The degree to which this changes is dependent on the openness of said agent. A lower the openness the greater the change.
  - Neuroticism can also result in sporadic daily fluctuations in the kindness of an agent. A higher neuroticism results in more substantial daily changes of the relevant behaviours.

- **Responsiveness**: This behaviour influences the likelihood of an agent accepting the treaties of other agents. It is initialised as the average of openness, extraversion, and agreeableness.

  In the current implementation, the responsiveness trait does not vary during the simulation. In an implementation where dishonesty is factored in, the agent would keep record of the trustworthiness of each agent that it interacts with. If other agents are largely untrustworthy, overall responsiveness would go down proportionally.

The effect of personality traits on the behaviourial charactersitcs are described below:

- **Openness**: A high openness score means that agents are less impacted by floor changes and thus its behavioural characteristics are not varied as much. Conversely, a low openness score means an agent is significantly impacted by change such as arriving at a lower floor which increases their greediness more than it would an agent with a higher openness score. Openness also influences responsiveness, which dictates how much an agent is willing to accept or requests or treaties.

- **Conscientiousness**: A high conscientiousness score means an agent will demonstrate greater learning from its experiences, as it will identify underlying trends in these experiences in order to make better decisions. A more planning-oriented agent of such will also give greater importance to collaborating with other agents.

- **Extraversion**: Extraversion affects the responsiveness and a high score increases the likelihood of positive social interaction such as accepting requests or treaties.
- **Agreeableness**: Greediness and kindness are initialised by agreeableness and scaled accordingly. Agreeableness also has influence over responsiveness in the same way that openness and extraversion do.
- **Neuroticism**: Neuroticism creates random day-to-day volatality in the greediness and kindness of an agent. Since greediness and kindness directly control the amount of food taken, an agent with higher neuroticism may intend to take an inadequate or excess amount of food.

9.2.3 Operational Memory

The memory structure of the agent behaves analogously to human memory. It serves to store experiences and data acquired across the lifespan of the agent. These experiences and data serve to influence the agent's behaviour, inform it's decisions, and enhance its survival and organisation capabilities. Below is a description of all the elements of information stored in the agent's memory:

- `orderPrevFloors`: A chronologically ordered list of the floors that the agent has been in.
- `prevFloors`: A map of all the floors that the agent has experienced. This keeps track of the number of days spent on each given floor and the average food that was available whilst the agent was on that floor.
- `currentDayonFloor`:The number of days the agent has spent on the current floor.
- `currentFloorRisk`: This is the level of risk to the survival of the agent associated with the current floor.
- `daysHungry`: The number of consecutive days the agent has not consumed food.
- `seenPlatform`: This variable is used to remember if the agent has seen the platform. The agent uses this to decide when to prepare for a new day.
- `prevAge`: The age of the agent on the previous day.
- `prevHP`: A record of the HP at the end of the previous day.
- `foodEaten`: The amount of food consumed the previous day.
- `receivedReq`: This is a variable that indicates whether the agent has decided to abide by a request made in a message. A false value indicates that no requests have been accepted.
- `takeFood`: This stores the amount of food an agent has requested to be taken. It is initialised with a value of -1 if there are no accepted requests.
- `leaveFood`: This stores the amount of food an agent has requested to be left. It is also initialised with a value of -1 if there are no accepted requests.

9.2.4 Floor Risk Estimation

Upon a floor reshuffle, the agent uses its experiences from past floors in order to build an expectation of the level of risk it associates with the new floor that it has been assigned to. The first step to doing this is to generate a prediction for the amount of food that will be available on the current floor.

In order to make this estimate, the agent utilises the prevFloors map from its memory. This contains information on the average food that was available on each floor that the agent has already been on. The agent approximates the relationship between the floor level and the average food available on that floor. For example, let us take an agent that has been on floors 10 and 6, and is now assigned to floor 8. The agent will calculate the linear function which corresponds to the average food on floors 6 to 10. It will then use this function to obtain the estimate for the food it expects on floor 8.

The above assumes that the agent has experienced floors both above and below the current floor. In the scenario where the agent has only experienced floors either below or above its current floor, the agent will extrapolate the trend line for the average food from the closest experienced floor.

Next, the agent determines the level of risk to its survival that it associates with the current floor. It does this by checking whether the estimated amount of food would be enough to exit a hypothetical critical state and enter the weak state. The difference between the required food and the estimated food will determine the floor risk value. The floor risk value will subsequently adjust the agent's greediness.

Note that since higher levels of strategising are associated more with conscientiousness agents, their estimates will be more pessimistic. Therefore, they will underestimate food availability and the floor risk will have a greater impact on their greediness. In the case of less conscientiousness agents, who would be both less planning-oriented and also more optimistic, the food estimation is higher and the risk estimator is weighed down.

9.3 Agent Operation

This section outlines the particulars of agent operation and decision making.

9.3.1 Main Operation

At the point where each agent is deployed into a simulation, its personality traits, behavioural characteristics and memory bank are initialised.

Once initialisation is complete, the agents check their inbox for messages, requests or treaties. This is done every tick to ensure messages are detected from the floors above and below. When a message, request or treaty is found in the inbox, a function calls the corresponding handler to the message type. The handlers then decide whether a request or treaty is accepted and the relevant actions performed. This is further discussed in Section 9.3.3.

Once the inbox is handled, the agent checks whether a floor change has occurred and updates the floor Map (prevFloors) with the current available food on the platform for the day. The agent also completes a Floor risk assessment which has been outlined in the previous two sections.

The flow chart in Fig. 9.2 outlines the decisions made concerning how much food to take.

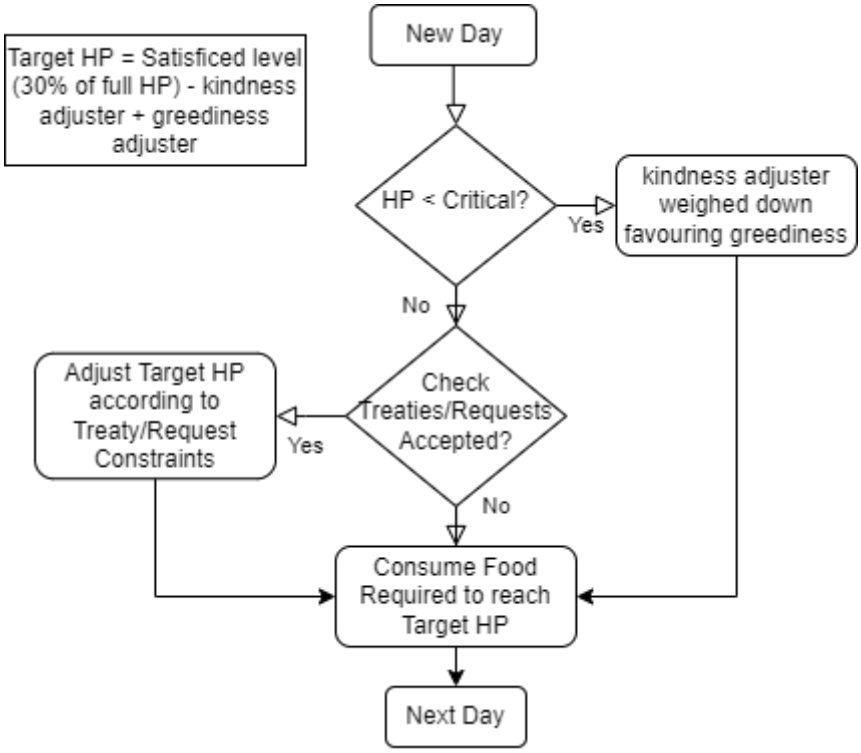


Figure 9.2: Food taking decision flow chart.

Upon completion of the memory bank updates and risk assessment routines, the agent then enters its food consumption routine. This routine runs only once per day when the platform is on the agent's floor. The routine begins by assessing whether the agent's health has been critical for multiple days, if this is the case the agent overrides any requests in effect and prioritises getting its health out of critical (Note however no treaties that prevent the agent from leaving critical HP would be accepted by the agent and thus no treaties would be broken at this point, treaty conditions are detailed in Section 9.3.3). The kindness adjuster is weighted down and the TakeFood function is executed. If the agent's health is not critical the routine evaluates whether it can abide within the constraints imposed for accepted treaties and requests. Once the constraints are in place the TakeFood function is executed in accordance to the greed and kindness adjusters within the bounds of the request/treaty constraints.

Once the routine is complete the seenPlatform variable is reset to indicate the end of the day.

9.3.2 Messages

Although not a crucial feature of the final agent strategy - reasonable consideration went into deciding how to leverage messaging to better serve an agent during a simulation of the tower.

In order to think about how messaging could benefit the agent, two scenario's can be considered - one where the agent is sending messages to make an alert, enquiry or request and the converse, where an agent receives messages and can perform some action based on the incoming information or request.

Sending Messages

The agent's behaviour is majorly determined by the intrinsic characteristics that constitute the agent's personality and so it was deemed that sending messages to gather information served no significant additional purpose as the new information would not have any influence on the next decision made. It is, however, important to benefit the collective effort of maintaining survival as a population - therefore an agent will send a reply to acknowledge information from other agent's and respond to requests stating whether they can comply.

Receiving Messages

The agent may adapt its behaviour upon receipt of request messages from other agent's. The criteria that must be met in order to comply with a request is as follows:

- The agent's extraversion rating must be higher than 5.
- The agent's kindness rating must be higher than it's greediness rating.
- The agent must not have been at critical health for $\geq$ (the max # of days at critical - 3).

If all of this criteria is met the agent will comply with the request, and it will reject the request if any of the conditions above are not met.

In all cases a truthful reply is sent from the agent to the message sender in an attempt to benefit the collective.

9.3.3 Treaties

Treaties are the mechanism by which agents can communicate their needs to each other and organise themselves in way that is mutually beneficial.

Rejecting Proposed Treaties

Treaties are rejected on the basis of certain criteria, these 'conditions' reflect proposed treaties that are detrimental to the health of Team 7's agents at the expense of other agents. The following are the main criteria for rejecting treaties:

1. **Personality and behaviour traits:**
   Treaties that are proposed to agents that have a less than average responsiveness are automatically rejected as this aligns with our aim to replicate human responses with agents. Note that responsiveness is an average of agreeableness, openness and extroversion. Thus most personality traits are taken into account. Similarly, an agent with conscientiousness less than 33% lacks the intuition to plan for future events and thus is casual in its approach to treaties, resulting in viable treaties being rejected. This shows our commitment to having agents mirror their personality traits in the actions that they take.
2. **Unfavourable Condition Operators for Condition Types HP & Available Food:**
   If the condition type of a particular treaty is **HP** OR **Available Food** AND the the condition operator is **LE** OR **LT**, then the treaty is unfeasible for survival as an individual

and/or collective. It is also unreasonable to assume realistic agents that mimic human behaviour would accept any alliances in which they were bound to a limited amount or percentage of food when their health is at the critical level. Thus all treaties that have the ability to put our agents in this position are rejected.

3. **Unfavourable Condition Operators for Condition Type Floor:**
   If the condition type is **Floor** AND the the condition operator is **GE** OR **GT**, then the treaty is rejected. This is again due to the overwhelming risk agents are exposed to when they are on a floor with a scarcity of food and resources - they will likely be critical and thus shouldn't be bound by treaties that limit their food intake.

4. **Detrimental Request Operators:**
   Any treaty that uses the **LE** OR **LT** request operators will be rejected. This is due to the fact that the request types are concerned with leaving a certain amount or percentage of food and thus if being asked to limit the lower bound of our food intake, our agent may consume a larger than necessary amount/percentage of food and result in other agents starving to death. This ultimately defeats the purpose of self-organisation.

5. **Treaties active at the Critical Level:**
   If the condition type holding up a certain treaty is **HP** and the condition value is less than the value required for the agent to be at the weak level, then such a treaty is not accepted. This is purely because of the risk it poses to our agent when we are below the weak level and require food but are bounded by a signed treaty that doesn't allow us to take any.

6. **Unreasonably long Duration of Treaties:**
   The duration of the treaty is also an important consideration to take into account. For a treaty to be fully effective, it must persist for an adequate amount of time such that it is used beneficially for self-organisation. Therefore treaties in which our agents' actions are restricted for a long duration are rejected to avoid the negative impact these outdated treaties could have.

7. **Clash of treaties:**
   Each new treaty is compared to the existing active treaties that are already stored in the agent's memory. If a treaty is incompatible with those that are already active, then the agent will reject such treaties. For example: treaties that are based on the same request type cannot ask for conflicting amounts of food, for e.g. If a treaty asking us to leave exactly 10 units of food has been accepted, then a freshly proposed treaty requesting to leave less than 7 units of food has to be rejected to avoid conflict.

Once a treaty is rejected, a confirmation of the rejection is sent to the proposer of the treaty and a message noting the outcome of the treaty is added to the log file at the output.

Accepting Treaties

Treaties are accepted if the aforementioned rejection criteria are not satisfied and the agent has the relevant personality traits and behaviour. In particular, the responsiveness behaviour characteristic of a given agent should be greater than 50 and the personality trait conscientiousness should be greater than 33.

All accepted treaties are stored in the active treaties field of the Base Agent thus allowing for the conflict criteria to be checked by iterating through the `activetreaties` map. Theoretically

all treaties that get through the vetting process are either beneficial to the agent or the wider group and do not make the agent prone to an earlier than expected death as a result of accepting said treaty.

Treaty Usage

All treaties accepted are stored in the active treaties field of the base agent. Since treaties that could negatively impact the agent are rejected, ones that are signed can be followed, without further checking.

Before an agent takes its share of food, The condition statements pertaining to active treaties are extensively checked to make sure that a treaty is being considered only when its respective conditions are upheld. Subsequently, the agent must check the request type, value, and operator. If the request type is *LeavePercentFood*, it is converted to an *amount* for easy comparison. The amount of food being taken is checked for compatibility with the amount of food that is to be left. If this is not the case then the *foodtotake* variable for the agent is updated accordingly. This will repeatedly happen until the expiration date of the treaty.

Treaty Formulation

Treaty Formulation can play an important role in creating favourable conditions, not only for the survival of the proposing agent, but also for other agents. There are three types of scenarios which trigger the agent to propose a new treaty to its surrounding agents:

1. **Health:** When an agent is in critical health state it is compelled to ask the neighbour above to leave an amount of food that can allow him to survive, i.e. go from critical to weak state. This health condition for this treaty is set such that the agents who sign the treaty are not put in danger, and this also makes it more likely for the treaty to be accepted.

2. **Floor:** If the conscientiousness of the agent is a above a certain threshold, meaning that the agent is relatively forward thinking and aware of their surroundings, then the agent will try to initiate a treaty with the neighbouring agent above him. This is triggered upon a floor change when the current floor is at a lower level than the previous floor. A conscientiousness agent will know that it is likely that food will be more scarce at this level and so he initiates a treaty to make survival at this floor easier.

3. **Propagation:** A treaty that has been accepted from an agent will be propagated onward in the same direction. If the treaty is coming from downwards and the request type is 'LeaveAmountFood', then the request value is increased before propagating the same treaty upwards. This must be done to ensure that the agent above leaves enough food for our agent *and* the agents below us. If the agent above uses the same strategy, then this can form an efficient means of survival for all agents in the tower.

9.4 Simulations and Analysis

Significance of Treaties

Fig. 9.3 and Fig. 9.4 illustrate the results from simulations with treaties disabled and enabled respectively. All other factors are left as default.

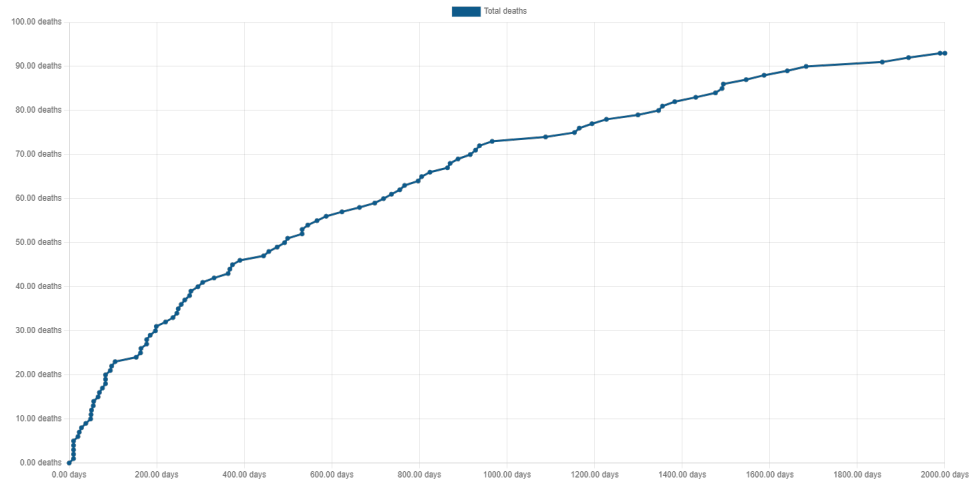


Figure 9.3: Cumulative deaths without treaties (93 deaths).

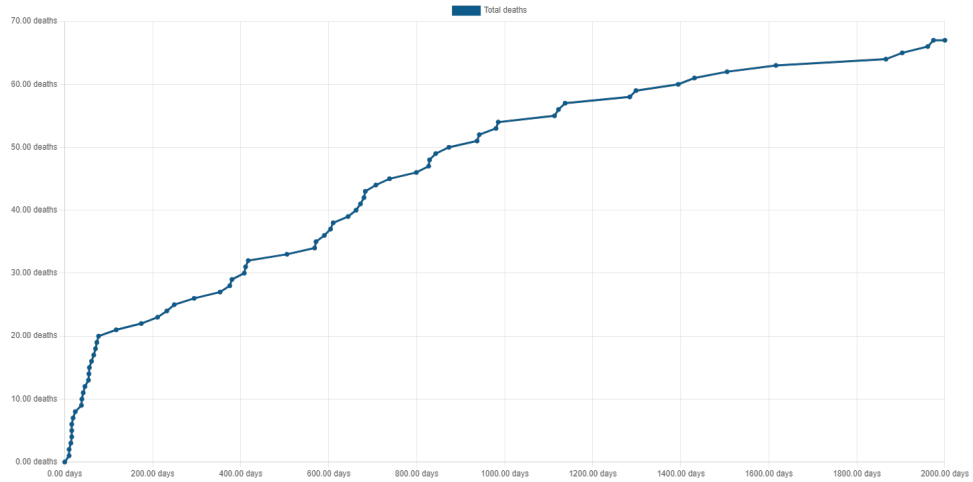


Figure 9.4: Cumulative deaths with treaties (67 deaths).

This demonstrates that the agents perform better collectively when treaties are activated. This is expected as treaties enable agents to cooperate and organise the distribution of scarce food.

Impact of Personalities

Openness

Fig. 9.5 and Fig. 9.6 detail how our agents adapt to situations/scenarios differently based on varying levels of openness. In Fig. 9.5, all the agents are spawned with openness values greater than 70 and in Fig. 9.6, the agents are spawned with openness values less than 30. All other personality traits are distributed as in the default case.

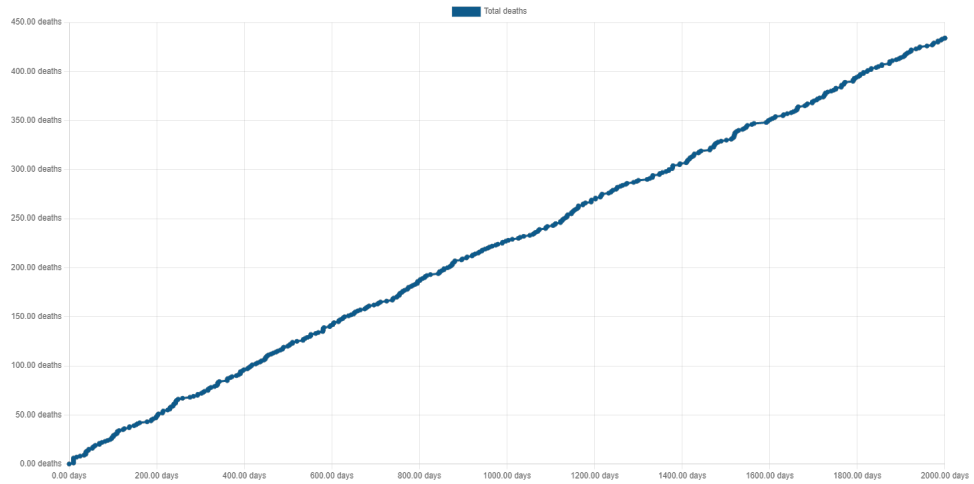


Figure 9.5: Cumulative deaths with high openness (434 deaths).

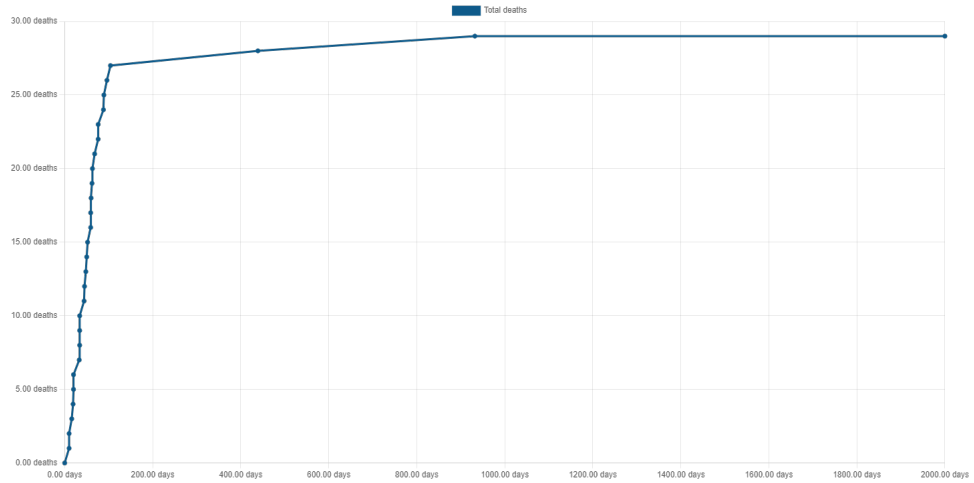


Figure 9.6: Cumulative deaths with low openness (29 deaths).

It can be observed that the agents collectively perform significantly better when openness is low, as a stabilising of the cumulative deaths curve can be seen for the agent with low openness. This can be explained by the fact that an agent with high openness will have a lower standard for accepting message requests and treaties - meaning they are more likely to agree to a request that is detrimental to their survival. Furthermore, an agent with a low openness value will be less welcoming to a change in floor and will therefore behave more cautiously. This will also

improve the agents chances of survival.

Neuroticism

Fig. 9.7 and Fig. 9.8 illustrate the results from simulations with agents with high neuroticism (>70) and low neuroticism (<30) respectively. All other traits are left as default.

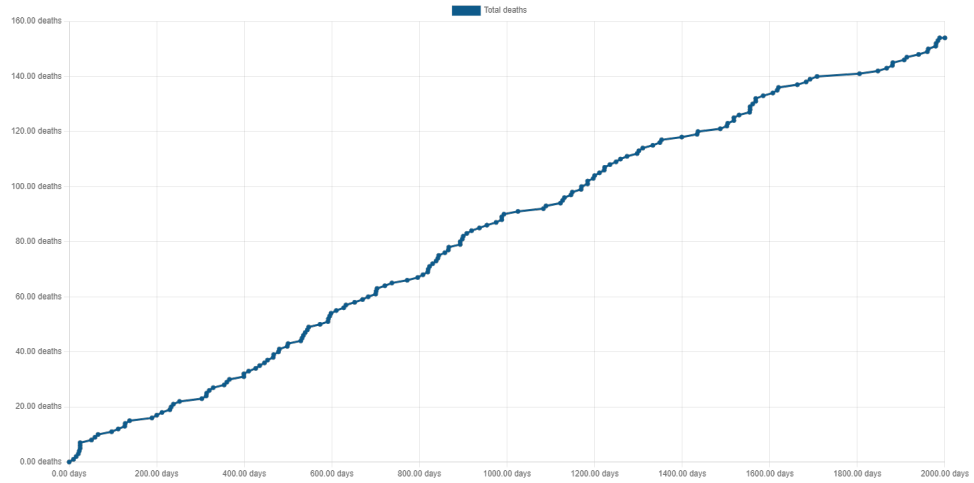


Figure 9.7: Cumulative deaths with high neuroticism (154 deaths).

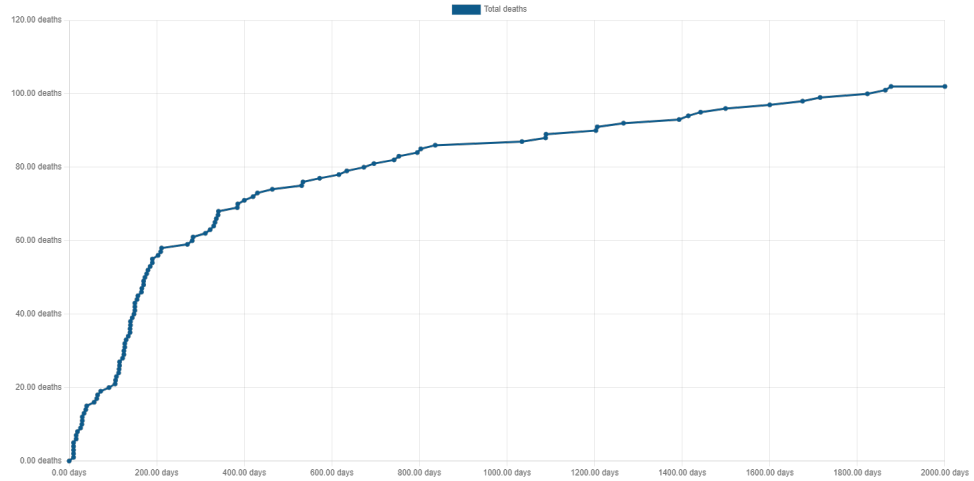


Figure 9.8: Cumulative deaths with low neuroticism (102 deaths).

It can be seen that higher levels of neuroticism result in a greater number of agent deaths. This is expected as neuroticism results in volatility in an agents behaviour.

Conscientiousness

Fig. 9.9 and Fig. 9.10 illustrate the results from simulations with agents with high conscientiousness (>70) and low conscientiousness (<30) respectively. All other traits are left as default.

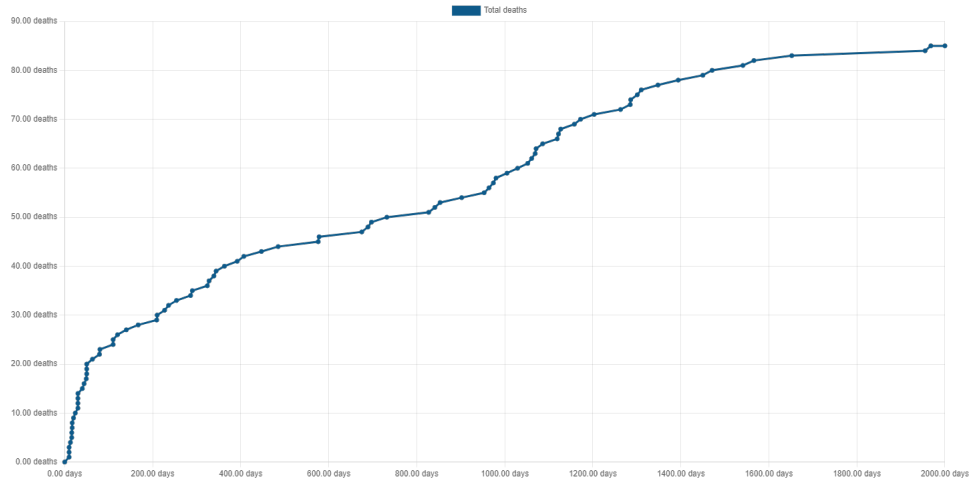


Figure 9.9: Cumulative deaths with high conscientiousness (85 deaths).

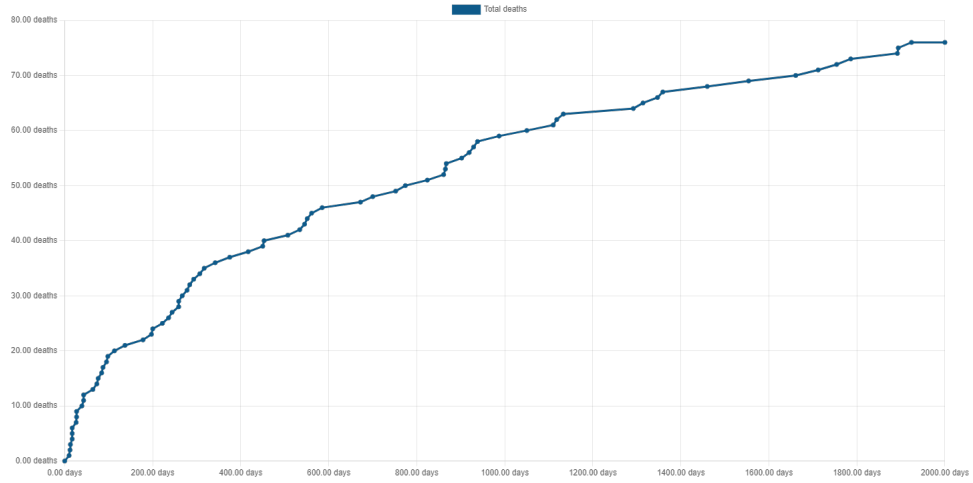


Figure 9.10: Cumulative deaths with low conscientiousness (76 deaths).

It can be observed that there is no substantial difference in the number of deaths when the conscientiousness levels of the agents are collectively varied. This may indicate that the agent struggles to be strategic and that simply being more greedy can be a more effective route to survival. Ultimately, it would appear that other factors outweigh the impact of this personality trait. This is not entirely unreasonable as other personality traits may be more relevant to the given scenario. However, the expectation would be for deaths to reduce with higher conscientiousness although this is not generally observed.

Extraversion

Fig. 9.11 and Fig. 9.12 illustrate the results from simulations with agents with high extraversion (>70) and low extraversion (<30) respectively. All other traits are left as default.

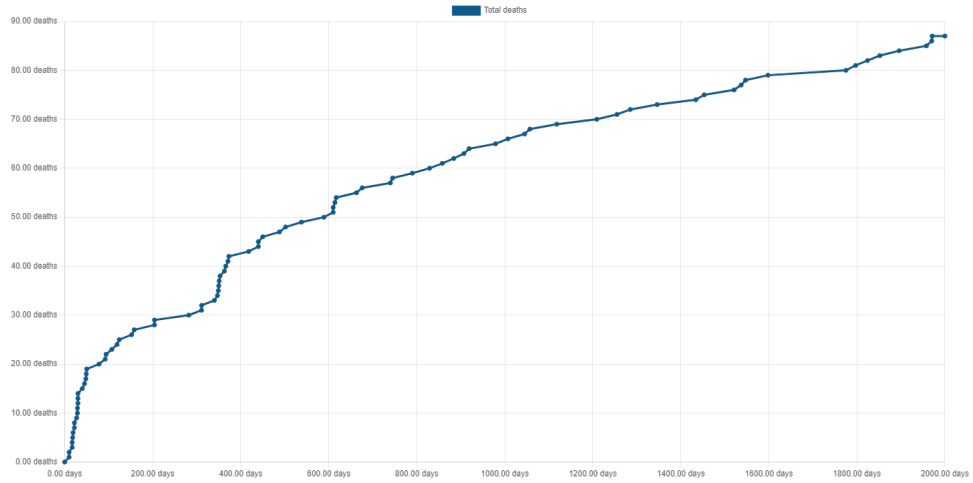


Figure 9.11: Cumulative deaths with high extraversion (87 deaths).

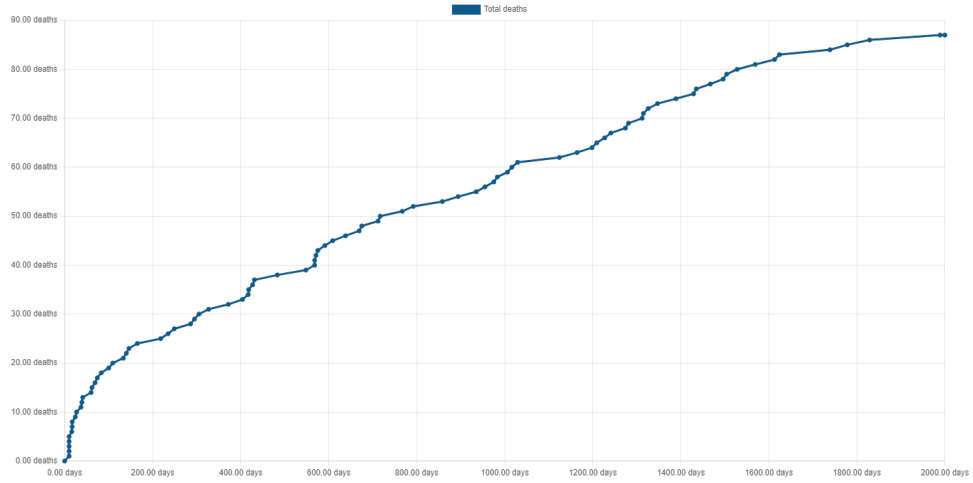


Figure 9.12: Cumulative deaths with low extraversion (87 deaths).

Similar to extraversion, simulations show little variation in the number of deaths when the agents' extraversion values are collectively set to either high or low. This may suggest that other factors outweigh the impact of this personality trait. This is not unreasonable as other personality traits may be more relevant to the given scenario. In addition, it is difficult to predict what impact the extraversion level would have on the number of deaths. On one hand, higher extraversion levels will enable a greater number of treaties being signed. On the other hand, it may also result in treaties and requests of lower standards being accepted. These counteractive phenomenon may be the primary cause for the lack of variation in deaths.

The following figures explore scenarios where the agents greediness and kindness are fixed before the food taking routine.

Greediness

Fig. 9.13 and Fig. 9.14 illustrate the results from simulations with agents with high greediness (>70) and low greediness (<30) respectively. All other traits are left as default.

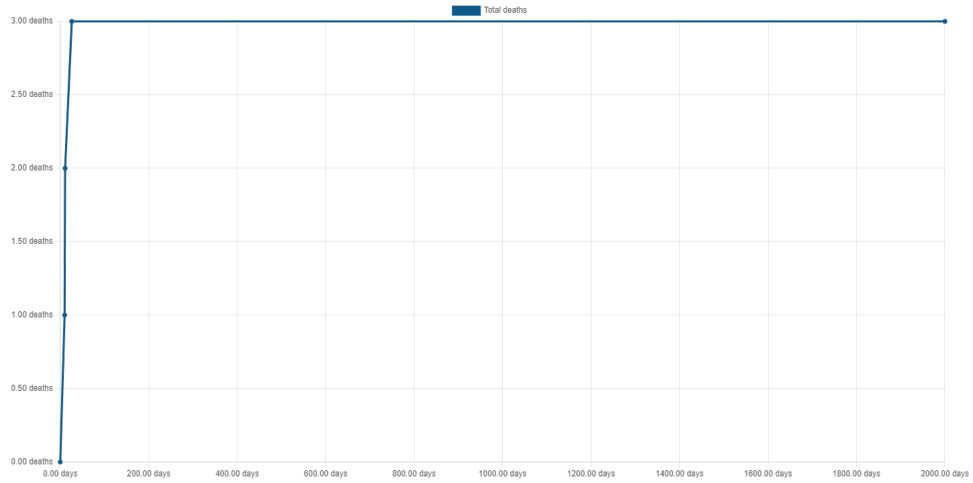


Figure 9.13: Cumulative deaths with high greediness (3 deaths).

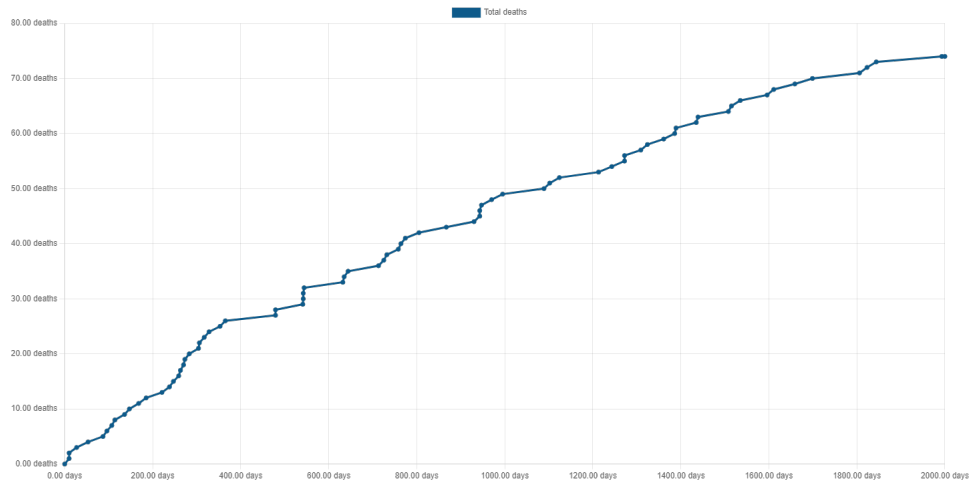


Figure 9.14: Cumulative deaths with low greediness (74 deaths).

A tower housing agents with high greediness and enforceable (unbreakable) treaties capabilities show an extremely low death rate suggesting that such a group of agents are able to organise themselves quickly and stabilise the system. A low greediness score sees a dramatic increase in overall deaths and does not achieve stability over the 2000 day simulation.

Kindness

Fig. 9.15 and Fig. 9.16 illustrate the results from simulations with agents with high kindness (>70) and low kindness (<30) respectively. All other traits are left as default.

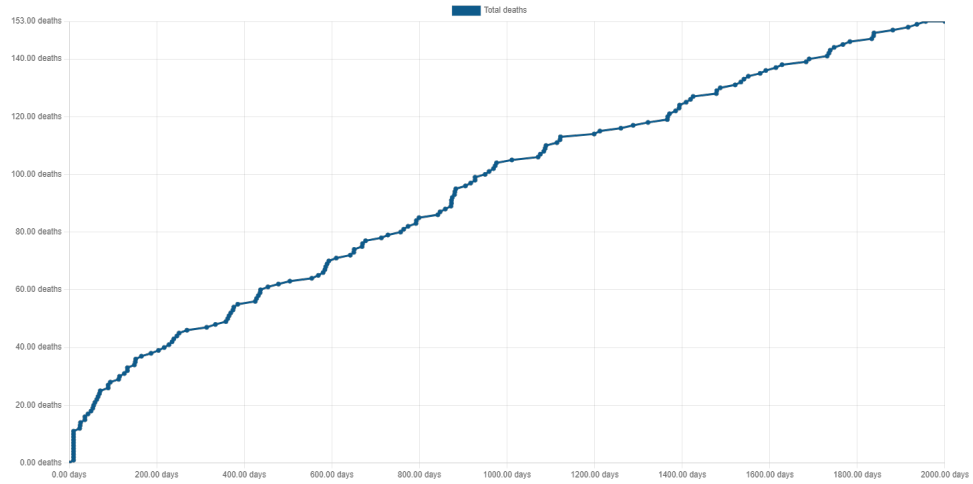


Figure 9.15: Cumulative deaths with fixed high kindness (153 deaths).

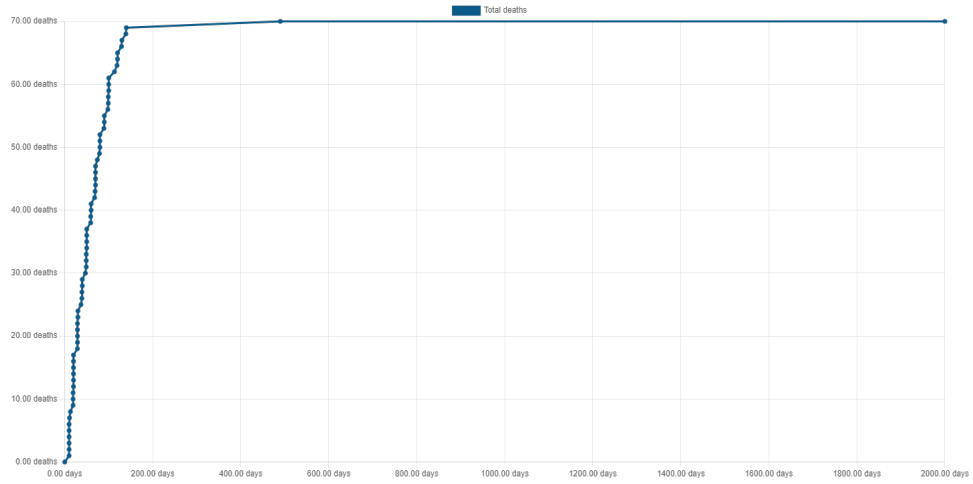


Figure 9.16: Cumulative deaths with fixed low kindness (70 deaths).

A high kindness score sees double the deaths of a low kindness score. A high kindness score does not achieve stability however a low kindness score allows for the tower to reach it in less than 200 days. The most likely explanation for this is that agents with high kindness are potentially too selfless. It is likely that they consume in small quantities when food is available and thus do not have sufficient health when food isn't available. Thus, making them more vulnerable.

Inter-team Comparison

Table 9.1 shows the total number of deaths for simulations of various combinations of agent teams. All simulations are run with a simulation period of 500 days, shuffle period of 7 days

Agent Team	Food/Agent	Total Deaths	Team 7 Deaths	Other Deaths
2	5	463	273	190
2	10	149	80	49
2	15	0	0	0
3	5	347	183	164
3	10	73	53	20
3	15	0	0	0
4	5	217	92	125
4	10	0	0	0
4	15	0	0	0
5	5	213	107	106
5	10	5	5	0
5	15	0	0	0
6	5	459	136	323
6	10	273	48	225
6	15	126	24	102

Table 9.1: Comparison between agent teams and Team 7.

and 10 Team 7 agents with 10 agents of one other team type. The table is further broken down into food available per agent (5, 10 and 15).

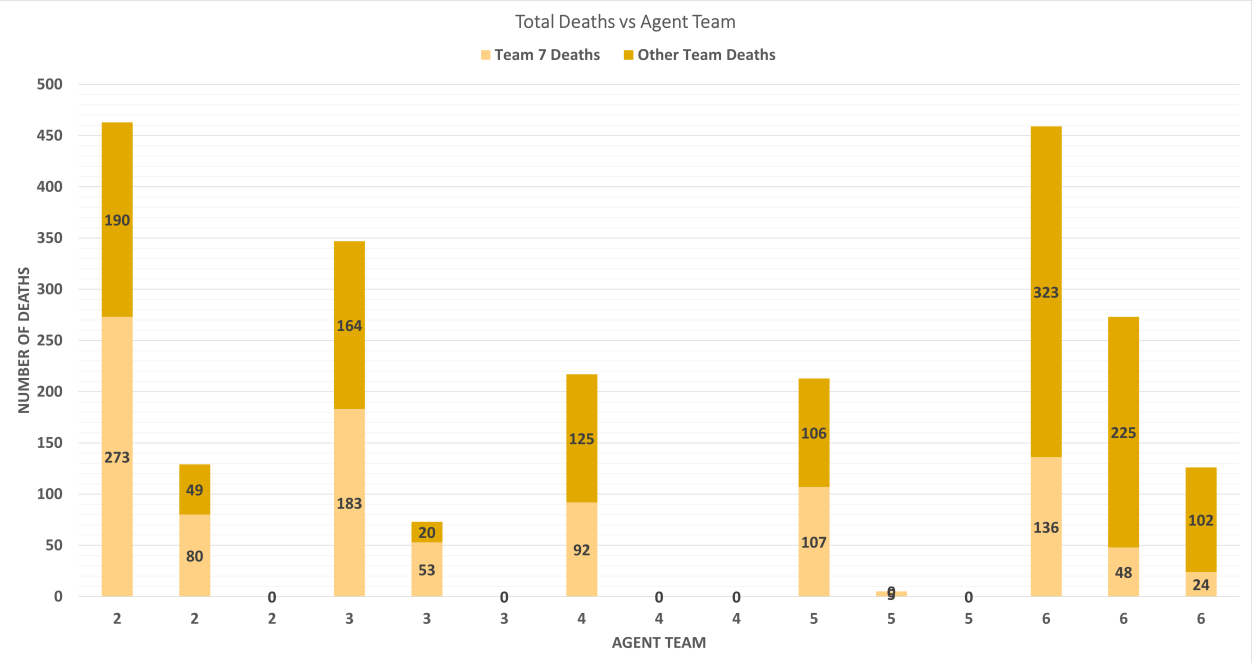


Figure 9.17: Deaths vs Agent Teams.

All team agents, bar Team 6, were able to coexist with Team 7 and achieve zero deaths for a food availability of 15 per agent, indicating that Team 6 agent was the most incompatible with our agent. Team 4 achieved zero deaths for food availability of 10 demonstrating a greater

compatibility with Team 7 and Team 5's results are not too dissimilar.

Conclusion

The agent designed by Team 7 is unique in terms of the use of personalities to determine the general behaviour of the agent. These personality traits assigned allow it to mimic human behaviour as much as possible. The results presented above show that Team 7's agents can have vastly different behaviours with each other agent, showing our compatibility with some agents and incompatibility with others. In short, the Team 7 agents have been intricately designed to meaningfully engage in a huge variety of scenarios and attempt to accomplish its to major goals, to survive individually and collectively.

Chapter 10

Experiments

10.1 Overall Aims

The aim of the broader experimentation was to see how the agents behave in an environment in which they all live and interact together and to see what changes when disruptive agents are introduced. Several variables were considered for each experiment, each of which are discussed in the following sections. The main dependent variable that will be considered is the death rate which determines the system stability. We also sought to observe the dependency on self-organisation to achieve stability and the effect of treaties and cooperation on self-organisation. Therefore, another important dependent variable that was measured was the number of treaties accepted and rejected by agents.

10.2 Disruptive Agents

The disruptive agents to be used in the following experiments were a random agent and a selfish agent. They have the following behaviours:

- Random Agent: takes random amounts of food every tick.
- Selfish Agent: takes food every tick such that it always attempts to stay at maximum health.

10.3 Stability

Before the effect of the parameters on stability can be measured, we must first define system stability. Stability is defined by Ashby as: "In all cases, the stable system is characterised by the fact that after a displacement we can assign some bound to the subsequent movement of the representative (equilibrium) point."[44] The features of a system's stability as given by [44] are as follows:

- Stability refers to some aspect of a system, not the system itself.
- A stable system is brought back to a state of equilibrium when it is displaced.
- Stability is a property of the whole system.

- Presence of stability implies some coordination of action of the parts.

In the case of the tower as a system, the death rate is considered to be the equilibrium point and the displacement being the addition of agents to the tower since, before the introduction of the agents, there is no movement in any of the parameters of the system. The equilibrium point being a death rate of zero.

The tower can therefore be considered stable if:

- The death rate reaches zero, and
- The death rate stays at zero for at least one reshuffle period, and
- The death rate stays at zero until the end of the simulation.

This definition of tower stability also satisfies all four requirements mentioned above and by Ashby.

10.4 Experiment 1

10.4.1 Aims

The aim of this experiment is to observe the impact of the addition of disruptive agents on the tower. We keep an equal number of each team's agents in the tower. Selfish agents and random agents are then added. The food per agent on the platform (food scarcity) is also varied. This allows us to observe the effect of food scarcity on stability and the effect of the addition of these disruptive agents on tower stability. We expect as food becomes more scarce and as more disruptive agents are added, that the total number of deaths will increase and that stability will be more difficult to achieve.

In this experiment the tower being used is highly heterogeneous, with three of each team agent type being added to it. Three random agents, three selfish agents or three of both, or none of either will then be added to to the tower.

10.4.2 Results and Discussion

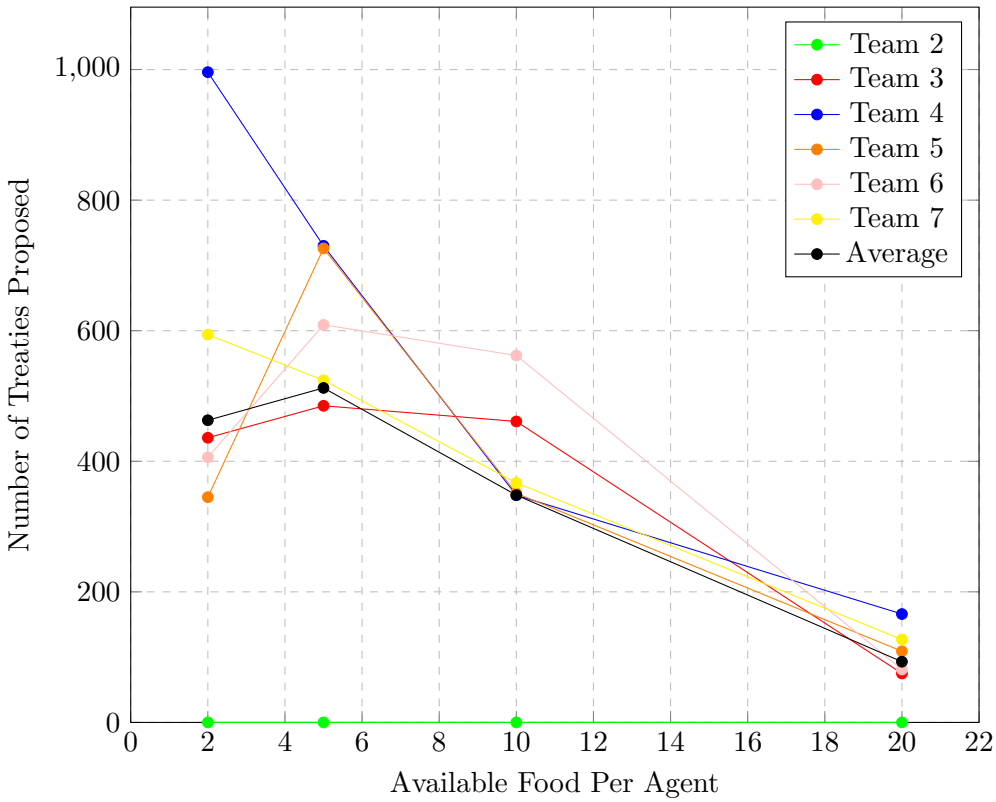


Figure 10.1: Number of Treaties Proposed as Food Scarcity Decreases

Fig. 10.1 illustrates the change in the frequency of treaty proposals as the constraints from the economy of scarcity are levied. The experiment enforces that there is 100 food initially available on the platform, with the number of agents parameterised to allow for an average amount of food per agent. This simulation lasts 400 days.

The general tendency for this system is that, as the availability of food rises (and hence the economy of scarcity becomes less constrictive), the rate of treaty proposals decreases. We propose that this experiment yields insight into the importance of treaties, demonstrating that agents in general rely heavily on treaty formation to self-organise when survivability is hindered by a lack of resources. Contrarily, as the food availability reaches an abundance, we see the rate of treaty proposals tend to 0, suggesting that agents are able to survive without the need for treaties as they offer little utility.

We also assert that this graph details the implied benefit of signing treaties: agents seem to value treaties as a means to fairer food distribution, wherein signing facilitates them receiving a greater proportion of food than they would otherwise receive without a treaty.

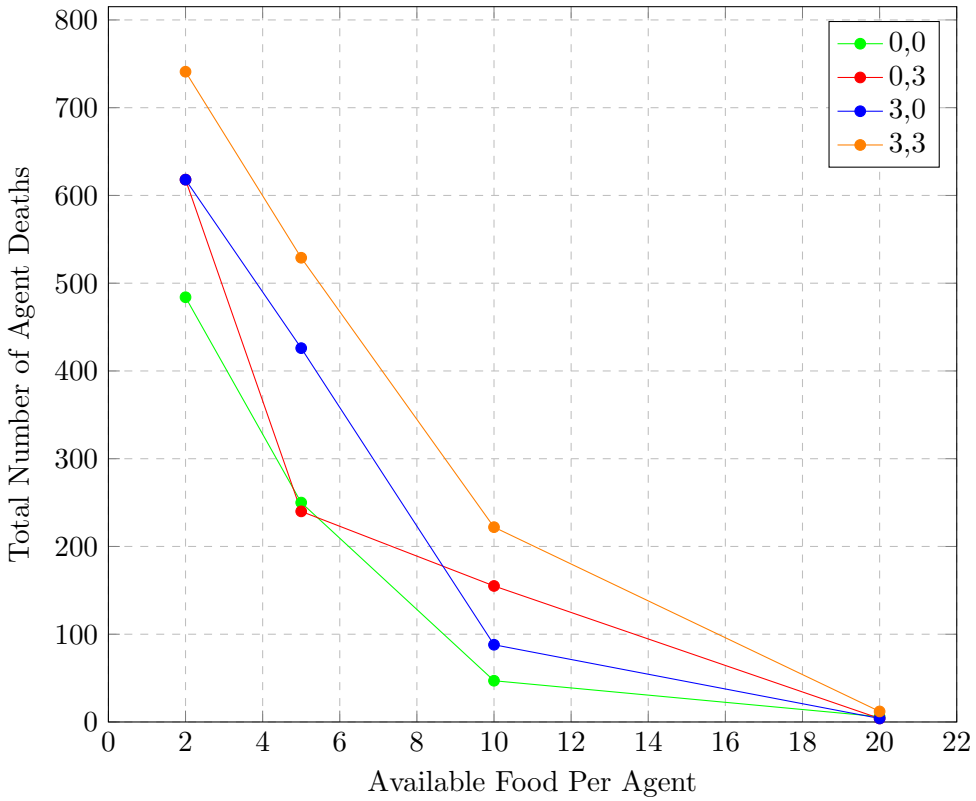


Figure 10.2: The total number of deaths in systems with different combinations of selfish and random agents added as food per agent increases. The number of each disruptive agent added to each system is show in the legend in the form (number of selfish agents, number of random agents)

Fig. 10.2 shows how the number of deaths is affected when different disruptive agents are added. It shows that, as expected, the number of deaths is higher when disruptive agents are present. However, the effect of selfish agents and random agents seems to be similar which suggests that it it not the amount of food being taken by the agents that is disruptive, rather it is the fact that they do not communicate that causes a higher death rate. This makes it harder to self-organise and makes it much harder to survive in the tower. By making it harder to survive in the tower, the chances of reaching a stable system are also decreased when disruptive agents are added. The only system to stabilise with ten food per agent was the one with no disruptive agents, all systems were stable with a tower of 20 food per agent.

10.4.3 Conclusion

This experiment shows the importance of treaties and communication for the survival of the agents in the tower and shows that self-organisation is a key factor in the ability of the tower to reach a stable state.

10.5 Experiment 2

10.5.1 Aims

The aim of this experiment is to vary the number of ticks per floor and reshuffle period and observe the effects of this on tower stability and ability to self-organise.

10.5.2 Results and Discussion

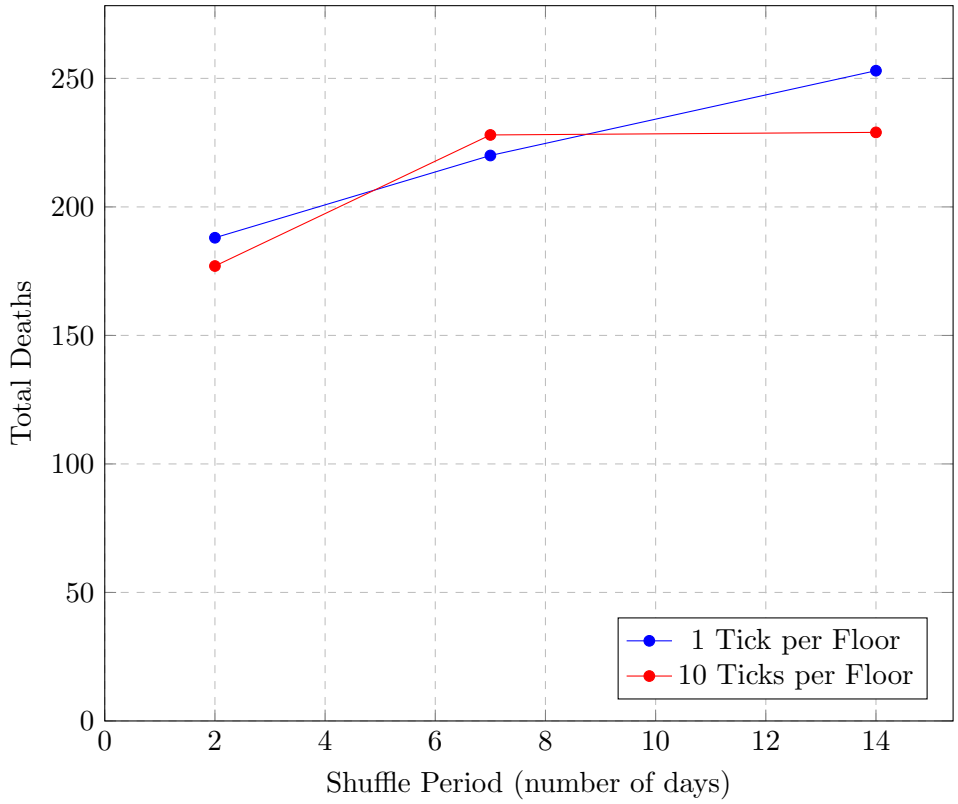


Figure 10.3: Total Deaths in the Tower as Reshuffle Period Increases

Fig. 10.3 investigates the correlation between the global deaths in the tower and the time between reshuffle periods.

A clear positive correlation emerges in this experiment, strongly supporting that the longer the periods between reshuffling, the more total deaths there are in the tower.

Similarly, we can see that the frequency of interaction (defined by the number of ticks per floor) has no overall impact on the death rate, suggesting that the quantity of communication is not the key factor in explaining these results. Furthermore, through propagation of treaties, we assert that the social network can still be formed effectively, even with increasing reshuffle periods.

For this reason, we propose that it is the rapid redistribution of agents throughout the tower, allowing them to experience varying levels of available food, that accounts for the disparity in survival rate: if an agent is consistently assigned to a low floor, they have a low chance of being satisficed if all agents act individually. Having a rapid reshuffle period results in a higher

probability of reaching a higher floor, allowing for the agent to replenish its health.

The average age of agents, denoted by Fig. 10.4, can be seen to follow the inverse trend to death rates, as expected. This indicates that the longer the reshuffle period the lower the average age of an agent upon death.

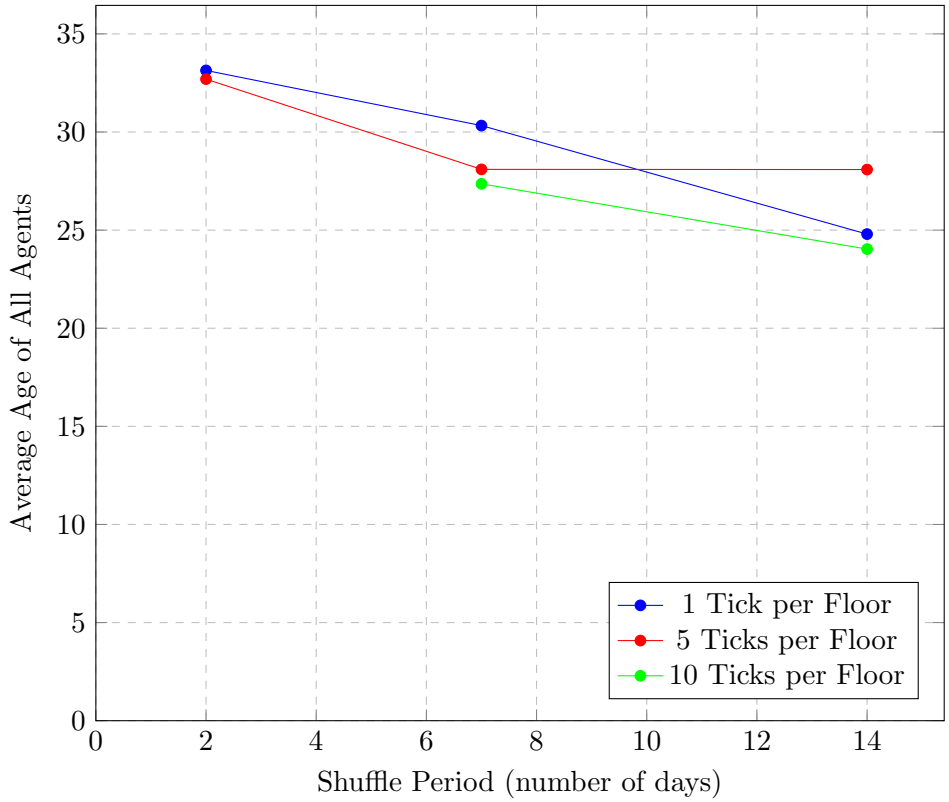


Figure 10.4: Average Age in the Tower as Reshuffle Period Increases

Fig. 10.5 and Fig. 10.6 show that significant numbers of treaties are being exchanged at all reshuffle periods, demonstrating that a social network is being formed. However, as the reshuffle period decreases the total number of treaties proposed also decreases. This is as expected since, if agents have the same neighbours for longer periods of time, they need to exchange fewer new treaties, given that each proposed treaty will be agreed upon by neighbours for longer and so neighbours will not need to send each other new treaties.

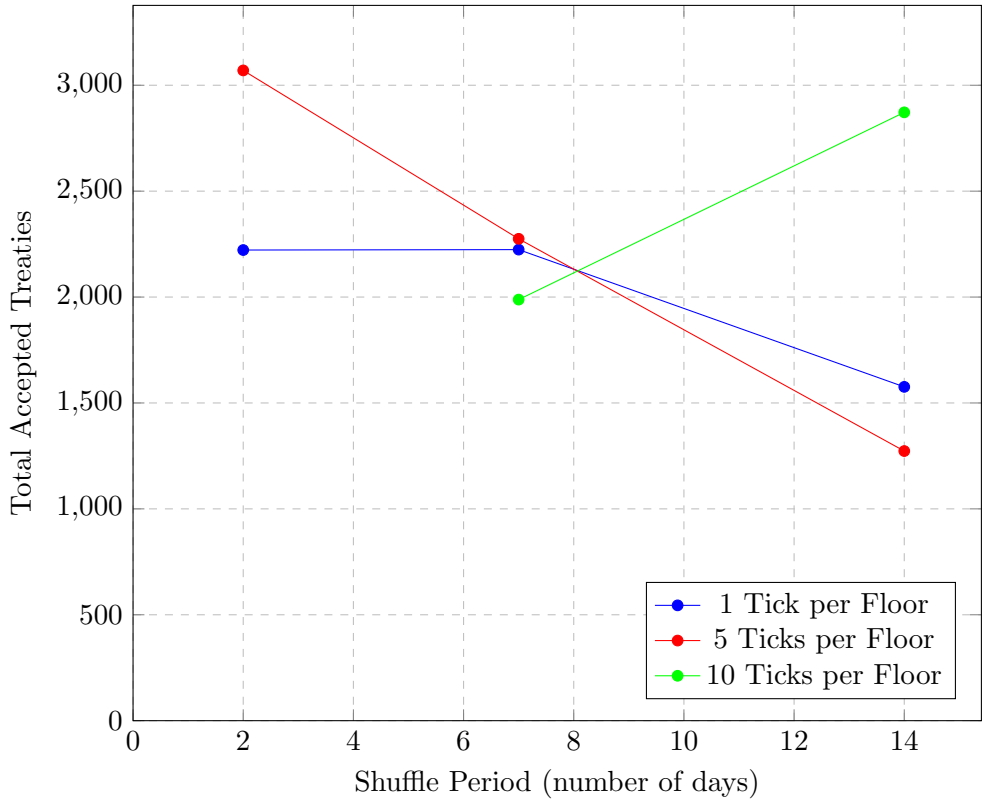


Figure 10.5: Total Accepted Treaties As Shuffle Period Decreases

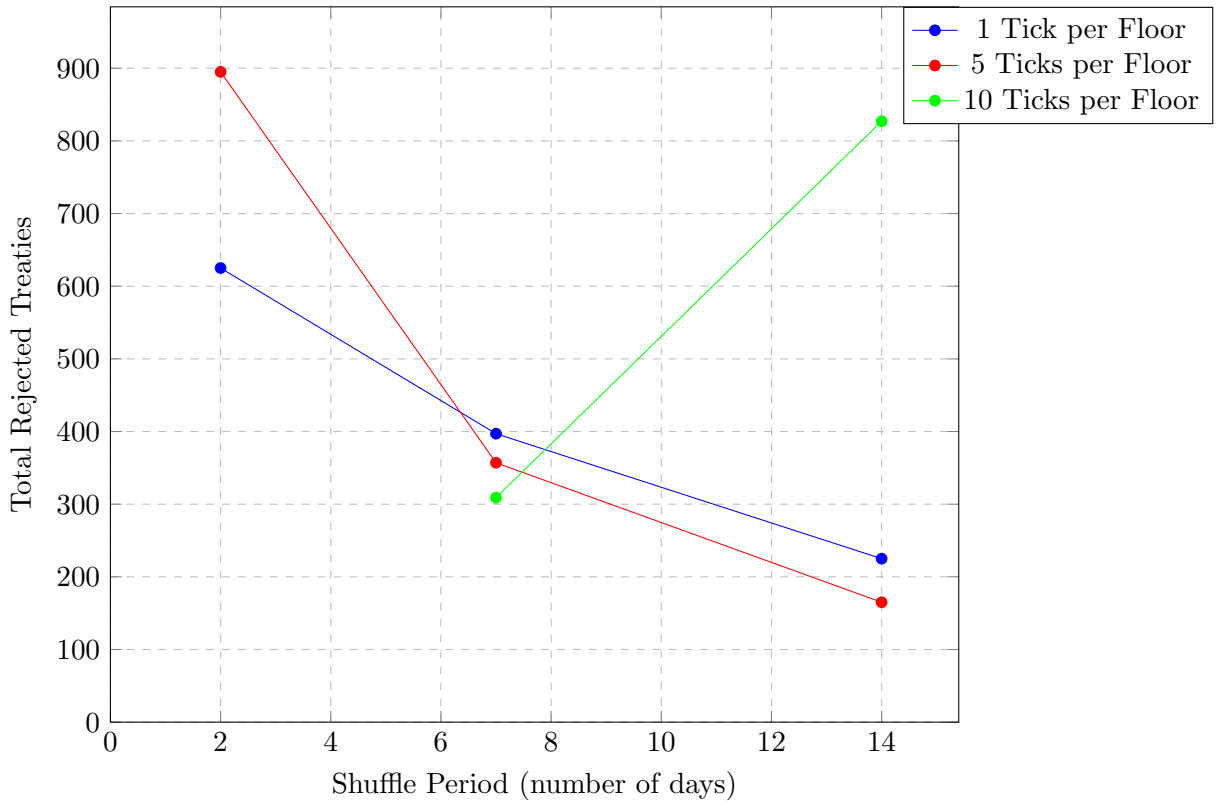


Figure 10.6: Total Rejected Treaties As Shuffle Period Decreases

10.5.3 Conclusion

The change in reshuffle period and number of ticks per floor does not have a significant impact on the ability of the agents to self-organise. However, increasing the reshuffle period does increase the number of deaths and so would make reaching stability more difficult.

10.6 Experiment 3

10.6.1 Aims

The aim of this experiment is to analyse the average distribution of food on each floor of a 100 floor tower. The simulations will have a duration of 60 days and an initial food of 800. Two simulations are proposed to assess this: a tower consisting entirely of `randomAgent` and a tower consisting entirely of custom agents from the different teams, 2 to 7. The aim is to illustrate that the average food per floor decreases more slowly when the custom agents are assessed, showing that a sufficient level of self-organisation has been achieved to more effectively distribute food.

10.6.2 Results and Discussion

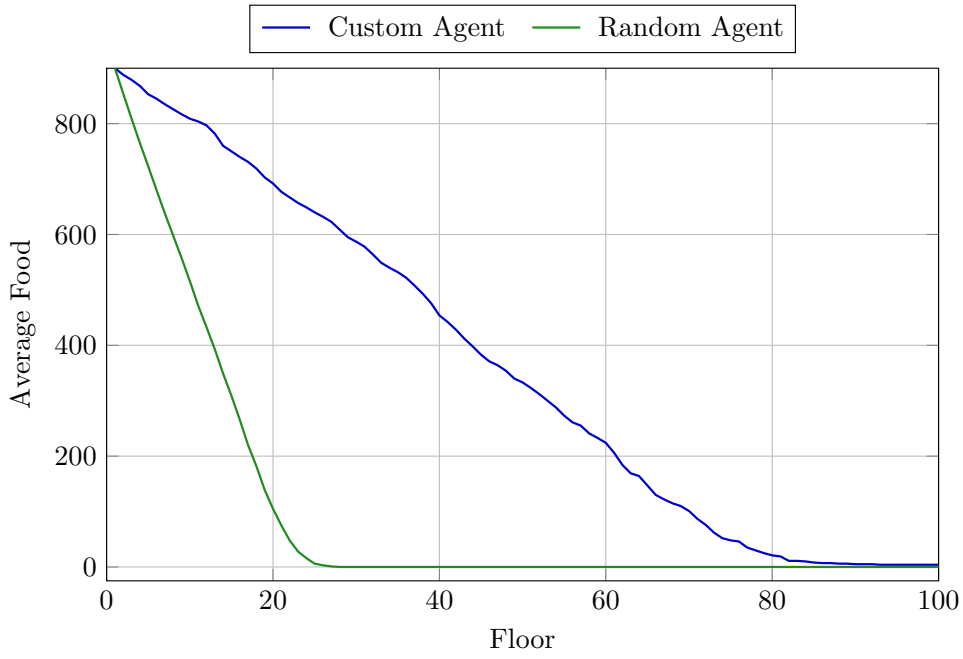


Figure 10.7: Average Food per Floor for Different Social Distributions

By analysing the performance of the random agents in Fig. 10.7, a clear example of 'Price's Law' can be seen, wherein the first 50% of food is consumed by $\sqrt{N} = 10$ of the agents. This is heavily contrasted with the set of custom agents, where it is evident that the resources are maintained for longer.

10.6.3 Conclusion

The results of this experiment point to an increase in social cohesion when the custom agents are implemented into the tower, allowing them to outperform a purely random strategy. This, coupled with the activeness of treaties, implies that the multi-agent system can sufficiently self-organise to maintain a fair and more uniform distribution of food for all agents involved.

Chapter 11

Conclusion

This project has involved the design and implementation of six different agents with various strategies to self-organise and solve a common resource management problem, where the common resource, in this case, is the scarcity of food available to agents in the tower.

It has been shown that in order for self-organisation to be achieved within the tower, communication is necessary between all present agents through both messaging and the signing of treaties to simulate localised governance. Agents which were successful in achieving self-organisation did so by gaining knowledge of both the tower and other agents, and applying this knowledge appropriately. Without this, the most logical behaviour of an agent is to act purely selfishly, giving itself the best chance of survival in a random system.

The codebase for this project can be found at https://github.com/SOMAS2021/SOMAS2021.

Appendix A

Project Management

A.1 Organisation

In order to allow for more focused work, the cohort was split into seven teams for this project:

- Team 1 was responsible for designing the infrastructure, MVP, and frontend and visualisation tools, with a group size of 9.
- Teams 2 to 7 were responsible for designing agent strategies, with group sizes of either 6 or 7.

Alongside this, some individual members took on additional organisation and planning roles for the cohort.

Each team had a team lead (except for Team 1, which due to its large size and remit, had two team leads):

1. Team 1: Jaafar Rammal and Cyrus Goodarzi
2. Team 2: Tom Eaton
3. Team 3: Sara Fernandez
4. Team 4: Hussain Kurabadwala
5. Team 5: Jason Zheng
6. Team 6: Matthew Scott
7. Team 7: Moin Bukhari

Team leads were responsible for representing the views of their team and communicating information between their team and the wider cohort.

Regular meetings were held between Professor Pitt and the SOMAS cohort to facilitate cohort-wide discussion and feedback.

Any member was free to discuss system features and implement these such as health, message passing, and treaties.

Go was chosen as the programming language with GitHub used for version control.

Communication was done primarily using the communication platform Slack, with some teams choosing to use their own methods of communication such as Discord, Microsoft Teams, and WhatsApp.

A.2 Reflection

No one in the cohort had any experience in writing Go prior to this project; every person who contributed to the codebase learned the language from scratch at the beginning of the project.

Many individuals were also not comfortable with using Git for version control; they have gained a lot more experience in using Git throughout this project.

Our fundamental organisation structure of having agent teams and a separate infrastructure team seemed to work well overall after the MVP was released, with the infrastructure team available to discuss and implement requests for functionality.

There should have been one person who was not assigned to any team designated as the overall project manager. The project management was primarily done by two people who were also team leads and who heavily contributed to discussion and implementation of key features; this meant that they were stretched very thinly across the project and had a workload significantly higher than most other members.

The system design architecture should have been one of the first decisions that we made as a cohort; instead this was done after deciding on programming language and splitting into teams. This turned out to be a mistake as the simulation architecture then became the sole responsibility of the infrastructure team. This is a high-stakes responsibility and as such should have had more involvement from the wider cohort.

There was no enforced requirement for teams to document features and functionality; this made conveying changes to the cohort difficult and would have made writing the report easier.

Furthermore, information about incoming features could have been communicated better; the primary way this was done for this project was by asking people to look at pull requests. This could have instead been done using a document summarising the changes and new APIs.

This report should have been written continuously as the project progressed rather than in the final week.

The role of a team lead should have been more explicitly defined: team leads did not have the same understanding of the role as the project managers did. This contributed to the unbalanced workload among the cohort. Moreover, the introduction of the role of team lead led to a loss of accountability among many members of the cohort who were not team leads and therefore many people did not keep up with updates in the codebase and Slack. This was not an expected outcome, but had the responsibilities of team leads and individual members been more explicitly defined, this could have been prevented. An example of a responsibility which should have been more explicitly defined is that team leads should have stayed updated with changes from the infrastructure team, and communicated these changes to their agent teams.

Those individuals who were stronger software engineers at the beginning of the project should have been designated to specifically act as code reviewers; these reviews can frequently take over an hour to complete and were primarily done by the project managers and a minority of team leads. Designating specific code reviewers would have helped to balance the workload more

fairly.

We did not properly formalise the dilemma that we were trying to solve with this project at the beginning; at times, it felt like we were designing a piece of software with no specific goal in mind.

Appendix B

List of Team Members

Team 1: Infrastructure and Frontend

- **Jaafar Rammal**
- **Cyrus Goodarzi**
- Gowoon Kim
- Udai Arneja
- Priya Chhaya
- Victor Florea
- Iason Vasileios Papadopoulos
- Gordon Thompson
- Seth Omoke-Enyi

Team 2: Agent Design

- **Tom Eaton**
- Wenqiang Lai
- Sebastian Aegidius
- Raaid Mahbub
- Chun Iao Tai
- Jiangnan Ye

Team 3: Agent Design

- **Sara Fernandez**
- Bjorn Hemberg

- Edward Schamp
- Eoin Quigley
- Kurt Martin-Brown
- Francisco Salgado

Team 4: Agent Design

- **Hussain Kurabadwala**
- Aditya Gupta
- Jason Keung
- Tejas Dandawate
- Jeetendra Joshi
- Rishil Patel

Team 5: Agent Design

- **Jason Zheng**
- Rhys Johnson
- Sebastiano Zane
- Daryl Lim
- Ben Stobbs
- Osman Ozevin
- Jozef Mastalarz

Team 6: Agent Design

- **Matthew Scott**
- Mathieu Dubied
- Gabrielle Rubin
- Arunansu Patra
- Christoph Renschler
- Adrian Koch
- Vasileios Manginas

Team 7: Agent Design

- **Moin Bukhari**
- Nayan Kad
- Abdullah Ehsan
- Zaid Jafarey
- Nasim Miah
- Shaheer Mapara

Bibliography

[1] Jasper Grashuis. An exploratory study of cooperative survival: Strategic adaptation to external developments. *Sustainability*, 10(3), 2018. ISSN 2071-1050. doi: 10.3390/su10030652. URL https://www.mdpi.com/2071-1050/10/3/652.

[2] E. Ostrom. *Governing the Commons*. Cambridge, 1990. URL https://books.google.co.uk/books?hl=en&lr=&id=4xg6oUobMz4C&oi=fnd&pg=PR11&ots=aP6oALnKZk&sig=CG6CS3C7YuUMhCJ1rwCAapxgMH0&redir_esc=y#v=onepage&q&f=false.

[3] J. Pitt, D. Busquets, Bourazeri A., and P Petruzzi. *Collective intelligence and algorithmic governance of socio-technical systems*, page 31–50. Springer International Publishing, 2014.

[4] Elinor Ostrom. Collective action and the evolution of social norms. *Journal of Economic Perspectives*, 14(3):137–158, September 2000. doi: 10.1257/jep.14.3.137. URL https://www.aeaweb.org/articles?id=10.1257/jep.14.3.137.

[5] Tomer Perry and Josiah Ober. Thucydides as a prospect theorist. *Polis The Journal for Ancient Greek and Roman Political Thought*, 31:206–232, 08 2014. doi: 10.1163/20512996-12340015.

[6] Wolfgang Reif, Gerrit Anders, Hella Ponsar, Jan-Philipp Steghöfer, Elisabeth Andre, Jörg Hähner, Christian Müller-Schloer, and Theo Ungerer. *Trustworthy Open Self-Organising Systems*. Springer International Publishing, 01 2016. ISBN 978-3-319-29199-4. doi: 10.1007/978-3-319-29201-4.

[7] Paul Reuter. *Introduction to the Law of Treaties*. Routledge, 1995.

[8] El hoyo, 2019.

[9] Jeremy Pitt. *Self-Organising Multi-Agent Systems*, pages 149–151. WORLD SCIENTIFIC (EUROPE), 2021. doi: 10.1142/q0307. URL https://www.worldscientific.com/doi/abs/10.1142/q0307.

[10] N Nilsson. Teleo-reactive programs for agent control. *The Journal of artificial intelligence research*, 1:139–158, 1994. ISSN 1076-9757.

[11] Garrett Hardin. The competitive exclusion principle. *Science (American Association for the Advancement of Science)*, 131(3409):1292–1297, 1960. ISSN 0036-8075.

[12] David A Whetten. Organizations: Rational, natural and open systems.w. richard scott. *The American journal of sociology*, 88(4):818–820, 1983. ISSN 0002-9602.

[13] Richard S. Sutton. *Reinforcement learning : an introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, second edition. edition, 2018 - 2018. ISBN 9780262039246.

[14] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial intelligence*, 136(2):215–250, 2002. ISSN 0004-3702.

[15] Icek Ajzen. The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, 50(2):179–211, 1991. ISSN 0749-5978. doi: https://doi.org/10.1016/0749-5978(91)90020-T. URL https://www.sciencedirect.com/science/article/pii/074959789190020T. Theories of Cognitive Self-Regulation.

[16] Victor Ocaya. Causal explanation in relation to human behaviour and moral responsibility. *Unknown*, 1985. URL https://heinonline.org/HOL/LandingPage?handle=hein.journals/zambia17&div=5&id=&page=.

[17] Martin Fishbein. *Reasoned Action, Theory of*. John Wiley and Sons, Ltd, 2008. ISBN 9781405186407. doi: https://doi.org/10.1002/9781405186407.wbiecr017. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781405186407.wbiecr017.

[18] C. Daniel Batson, Judy G. Batson, R. Matthew Todd, Beverly H. Brummett, Laura L. Shaw, and Carlo M. R. Aldeguer. Empathy and the collective good: Caring for one of the others in a social dilemma. *Journal of Personality and Social Psychology*, 68(4):619–631, 1995. doi: 10.1037/0022-3514.68.4.619. URL https://psycnet.apa.org/record/1995-25033-001.

[19] Ruodan Shao, Karl Aquino, and Dan Freeman. Beyond moral reasoning: A review of moral identity research and its implications for business ethics. *Business Ethics Quarterly*, 18(4):513–540, 2008. doi: 10.5840/beq200818436.

[20] Augusto Blasi. Moral cognition and moral action: A theoretical perspective. *Developmental Review*, 3(2):178–210, 1983. doi: 10.1016/0273-2297(83)90029-1.

[21] Ferdinanda Elfrida Hubertina Wijermans. *Understanding crowd behaviour: simulating situated individuals*. PhD thesis, 2011. Relation: http://www.rug.nl/ Rights: University of Groningen.

[22] Robert Sokolowski. Phenomenology of friendship. *The Review of Metaphysics*, 55(3):451–470, 2002. ISSN 00346632. URL http://www.jstor.org/stable/20131748.

[23] Michael E. McCullough, Marcia B. Kimeldorf, and Adam D. Cohen. An adaptation for altruism: The social causes, social effects, and social evolution of gratitude. *Current Directions in Psychological Science*, 17(4):281–285, 2008. doi: 10.1111/j.1467-8721.2008.00590.x. URL https://doi.org/10.1111/j.1467-8721.2008.00590.x.

[24] Paul Ekman. An argument for basic emotions. *Cognition and Emotion*, 6(3-4):169–200, 1992. doi: 10.1080/02699939208411068.

[25] Daniel Kahneman and Amos Tversky. *Choices, Values, and Frames*, page 1–16, 2000. doi: 10.1017/cbo9780511803475.002.

[26] Keith Frankish. Dual-process and dual-system theories of reasoning. *Philosophy Compass*, 5(10):914–926, 2010. doi: 10.1111/j.1747-9991.2010.00330.x.

[27] Prem Kamble. What is subconscious mind? how does it impact our behaviour? *SSRN Electronic Journal*, 2021. doi: 10.2139/ssrn.3806525.

[28] 1864-1944 Park, Robert Ezra. *The crowd and the public, and other essays.* Heritage of sociology. University of Chicago Press, Chicago,, 1972. "The crowd and the public [Masse und Publikum] translated by Charlotte Elsner. Note on The crowd and the public by Donald N. Levine.".

[29] Jeremy Pitt. Lecture notes on the trust framework, November 2021.

[30] Josiah Ober. Democracy and knowledge: Innovation and learning in classical athens. *Democracy and Knowledge: Innovation and Learning in Classical Athens*, 09 2008. doi: 10.1515/9781400828807.

[31] Christopher Reinders Folmer. *Social Motives*, pages 886–890. SAGE, 06 2016. doi: 10.4135/9781483346274.n303.

[32] Thomas C. Schelling. The strategy of conflict prospectus for a reorientation of game theory. *The Journal of Conflict Resolution*, 2(3):203–264, 1958. ISSN 00220027, 15528766. URL http://www.jstor.org/stable/172793.

[33] Augustine Kong, Gudmar Thorleifsson, Michael L. Frigge, Bjarni J. Vilhjalmsson, Alexander I. Young, Thorgeir E. Thorgeirsson, Stefania Benonisdottir, Asmundur Oddsson, Bjarni V. Halldorsson, Gisli Masson, Daniel F. Gudbjartsson, Agnar Helgason, Gyda Bjornsdottir, Unnur Thorsteinsdottir, and Kari Stefansson. The nature of nurture: Effects of parental genotypes. *Science*, 359(6374):424–428, 2018. doi: 10.1126/science.aan6877. URL https://www.science.org/doi/abs/10.1126/science.aan6877.

[34] Sibel Adali, Robert Escriva, Mark K. Goldberg, Mykola Hayvanovych, Malik Magdon-Ismail, Boleslaw K. Szymanski, William A. Wallace, and Gregory Williams. Measuring behavioral trust in social networks. In *2010 IEEE International Conference on Intelligence and Security Informatics*, pages 150–152, 2010. doi: 10.1109/ISI.2010.5484757.

[35] Elinor Ostrom and James Walker. *Trust and reciprocity: Interdisciplinary lessons for experimental research*, pages 24, 209. Russell Sage Foundation, 2003.

[36] Dennis Norris. Short-term memory and long-term memory are still different. *Psychological bulletin*, 143(9):992, 2017.

[37] Peter C Fishburn. *Nonlinear preference and utility theory*, page 2. Johns Hopkins University Press, 1988.

[38] Peter C Fishburn. Utility theory for decision making. Technical report, Research analysis corp McLean VA, 1970.

[39] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*, pages 99–127. World Scientific, 2013.

[40] W Keith Campbell, Adam S Goodie, and Joshua D Foster. Narcissism, confidence, and risk attitude. *Journal of behavioral decision making*, 17(4):297–311, 2004.

[41] W.R. Ashby. *Design for a Brain; The Origin of Adaptive Behavior - Scholar's Choice Edition*, chapter 4-5. Creative Media Partners, LLC, 2015. ISBN 9781296026226. URL https://books.google.co.uk/books?id=bX9HrgEACAAJ.

[42] J-P Vacher, Thierry Galinho, Franck Lesage, and Alain Cardon. Genetic algorithms in a multi-agent system. In *Proceedings. IEEE International Joint Symposia on Intelligence and Systems (Cat. No. 98EX174)*, pages 17–26. IEEE, 1998.

[43] Tom Froese. Steps toward the evolution of communication in a multi-agent system. In *Symposium for Cybernetics Annual Research Projects, SCARP*, volume 3, 2003.

[44] W. Ross Ashby. *Design for a Brain: The origin of adaptive behaviour*, pages 44–57. Springer Netherlands, Dordrecht, 1960. ISBN 978-94-015-1320-3. doi: 10.1007/978-94-015-1320-3_4. URL https://doi.org/10.1007/978-94-015-1320-3_4.

[45] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Transactions on Graphics*, 33(4):1–11, 2014. ISSN 15577333. doi: 10.1145/2601097.2601116.

[46] Dr. Mad Ithnin Salleh Norazlan Hasbullah, Prof. Dr. Abdul Jumaat Mahajar. A conceptual framework of extending the theory of planned behavior: The role of service quality and trust in the consumer cooperatives. *International Journal of Business and Social Science*, 2014.

[47] S. Danziger, J. Levav, and L. Avnaim-Pesso. Extraneous factors in judicial decisions. *Proceedings of the National Academy of Sciences*, 108(17):6889–6892, 2011. doi: 10.1073/pnas.1018033108.

[48] Tanya Heikkila. Institutional boundaries and common-pool resource management: A comparative analysis of water management programs in california. *Journal of Policy Analysis and Management*, 23(1):97–117, 2004. doi: https://doi.org/10.1002/pam.10181. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/pam.10181.

[49] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: theory and practice. *Knowledge engineering review*, 10(2):115–152, 1995. ISSN 0269-8889.